МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Відокремлений структурний підрозділ
«Тернопільський фаховий коледж
Тернопільського національного технічного університету імені Івана Пулюя»

Відділення телекомунікацій та електронних систем

Циклова комісія комп'ютерних наук

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи

фахового молодшого бакалавра

на тему: Розробка вебсайту магазину книг «BookFlow»

Виконала: студентка IV курсу, групи КН-421
спеціальності: 122 «Комп'ютерні науки»
Софія КУГАЇВСЬКА

Керівник Руслан СЛОБОДЯН

Рецензент
(ім'я та прізвище)

м. Тернопіль – 2026

ВІДОКРЕМЛЕНИЙ СТРУКТУРНИЙ ПІДРОЗДІЛ
«ТЕРНОПІЛЬСЬКИЙ ФАХОВИЙ КОЛЕДЖ
ТЕРНОПІЛЬСЬКОГО НАЦІОНАЛЬНОГО ТЕХНІЧНОГО УНІВЕРСИТЕТУ
ІМЕНІ ІВАНА ПУЛЮЯ»

Відділення телекомунікацій та електронних систем
Циклова комісія комп'ютерних наук
Освітньо-професійний ступінь «фаховий молодший бакалавр»
Спеціальність 122 Комп'ютерні науки
Галузь знань 12 Інформаційні технології

ЗАТВЕРДЖУЮ
Голова циклової комісії комп'ютерних наук
_____ Галина МАРЦІЯШ
« 02 » березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Кугаївській Софії Віталіївні
(прізвище, ім'я, по батькові)

1. Тема роботи: Розробка вебсайту магазину книг «BookFlow»

керівник роботи Слободян Руслан Олесійович,
(прізвище, ім'я, по батькові)
затверджені наказом вищого навчального закладу № 4/9-132 від 27.02.2026 р.
2. Строк подання студенткою роботи: 19.06.2026 р.
3. Вихідні дані до роботи: технічне завдання на розробку програмного забезпечення, мови програмування: Java, JavaScript; фреймворки: Spring Boot, Hibernate; бібліотека React, Vite, SCSS, стандарти IEEE 29148-2018, IEEE 29119, ДСТУ 8302:2015.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

1 Загальний розділ

1.1 Аналітичний огляд існуючих рішень

1.2 Технічне завдання

1.2.1 Найменування та область застосування

1.2.2 Призначення розробки

1.2.3 Вимоги до функціоналу вебзастосунку

1.2.4 Вимоги до програмної документації

1.2.5 Техніко-економічні показники

1.2.6 Стадії та етапи розробки

1.2.7 Порядок тестування та прийому

2 Розробка технічного та робочого проєкту

2.1 Розробка структури сайту і вебсторінок

2.2 Створення та верстка сторінок сайту

2.3 Розробка структури бази даних сайту

2.4 Програмування сайту

2.4.1 Написання клієнтської частини

2.4.2 Написання серверної частини

2.5 Тестування вебсайту

3 Спеціальний розділ

3.1 Інструкція з розміщення сайту в Інтернеті

3.2 Інструкція з обслуговування та наповнення сайту

3.3 Інструкція з популяризації та підтримки сайту

4 Економічний розділ

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

4.3 Розрахунок витрат на електроенергію

4.4 Розрахунок суми амортизаційних відрахувань

4.5 Обчислення накладних витрат

4.6 Складання кошторису витрат та визначення собівартості вебсайту

4.7 Розрахунок ціни вебсайту магазину книг «BookFlow»

4.8 Визначення економічної ефективності і терміну окупності капітальних вкладень

5 Охорона праці, техніка безпеки та екологічні вимоги

5.1 Укладання колективного договору. Його зміст

5.2 Вплив параметрів мікроклімату на організм людини

6 Висновки

Додаткові вказівки: виконання кваліфікаційної роботи із розробкою програмного продукту – вебсайту магазину книг «BookFlow».

5. Перелік графічного матеріалу:

1. Схема структурна клієнтської частини
2. UML-діаграма станів
3. ER-діаграма бази даних вебсайту
4. Таблиця техніко-економічних показників

6. Консультанти розділів роботи

Розділ	Ім'я, прізвище та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний розділ	Любов КАЛУШКА		
Охорона праці, техніка безпеки та екологічні вимоги	Генадій ГОРЯЧЕК		

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Отримання і аналіз технічного завдання	20.03.2026	
2	Збір і узагальнення інформації	01.05.2026	
3	Написання першого розділу	15.05.2026	
4	Розробка технічного та робочого проєкту	29.05.2026	
5	Написання спеціального розділу	05.06.2026	
6	Розрахунок економічної частини	08.06.2026	
7	Написання розділу охорони праці	09.06.2026	
8	Виконання графічної частини	10.06.2026	
9	Оформлення пояснювальної записки	11.06.2026	
10	Погодження нормоконтролю	12.06.2026	
11	Попередній захист кваліфікаційної роботи	.06.2026	
12	Захист кваліфікаційної роботи	.06.2026	

7. Дата видачі завдання: 05 березня 2026 р.

Студентка _____ Софія КУГАЇВСЬКА
(підпис)

Керівник кваліфікаційної роботи _____ Руслан СЛОБОДЯН
(підпис)

ЗМІСТ

АНОТАЦІЯ

Тема кваліфікаційної роботи: Розробка вебсайту магазину книг «BookFlow».

Метою кваліфікаційної роботи є реалізація вебсайту магазину книг «BookFlow» на основі застосування сучасних інформаційних технологій та методів веброзробки.

Пояснювальна записка складається з 5 розділів.

У загальній частині описуються аналітичний огляд існуючих рішень та аналіз технічного завдання.

У другому розділі представлено процес створення вебсайту та верстка його сторінок, опис та обґрунтування вибору розроблюваної структури бази даних вебсайту, а також описи процесів написання клієнтської та серверної частин магазину.

У спеціальній частині описані процес розміщення вебсайту в Інтернеті, інструкція з наповнення та правилами обслуговування вебсайту, а також інструкція з популяризації та підтримки програмного продукту.

Розрахунок вартості розробки та економічної ефективності приведено в економічній частині.

Основні питання охорони праці та техніки безпеки розглянуто в п’ятому розділі.

Обсяг пояснювальної записки 79 сторінок.

До складу кваліфікаційної роботи входить графічна частина, яка складається зі структурної схеми клієнтської частини, ER-діаграми бази даних, UML-діаграми станів замовлення та таблиці техніко-економічних показників, що виконані на окремих аркушах формату А1.

ABSTRACT

Topic of the qualification paper: Development of the «BookFlow» bookstore website.

The objective of the qualification paper is to implement the «BookFlow» bookstore website by applying modern information technologies and web development methods.

The explanatory note consists of 5 sections.

The general section provides an analytical overview of existing solutions and an analysis of the technical specifications.

The second section presents the process of website creation and its page layout development, describes and justifies the chosen database structure for the website, and outlines the development processes of the client and server sides of the store.

The special section describes the process of website deployment on the Internet, provides content management instructions and website maintenance rules, along with guidelines for the promotion and support of the software product.

The calculation of development costs and economic efficiency is presented in the economic section.

The fundamental issues of occupational health and safety are addressed in the fifth section.

The total volume of the explanatory note is 79 pages.

The qualification paper includes a graphic part, which consists of a structural diagram of the client side, a database ER diagram, an order state UML diagram, and a table of technical and economic indicators, executed on separate A1 format sheets.

ВСТУП

Стан розвитку суспільства зараз характеризується глобальними процесами цифровізації та стрімким переходом традиційних бізнес-моделей у площину електронної комерції. Для України це питання набуло особливої гостроти внаслідок повномасштабного вторгнення, що спричинило критичні пошкодження виробничих потужностей друкарень, дестабілізацію логістичних ланцюгів та збитки для мереж фізичних книжкових магазинів. У таких умовах книжковий сектор, який є фундаментом культурного розвитку та збереження національної ідентичності, потребує впровадження інноваційних цифрових рішень для забезпечення своєї життєздатності та доступності для читача.

У свою чергу завдяки високому рівню адаптивності українського IT-сектору та активному розвитку інструментів веброзробки, створення спеціалізованих вебплатформ виступає ефективним засобом підтримки видавничої галузі. Це дозволяє не лише нівелювати фізичні бар'єри у доступі до книг, а й значно розширити ринок збуту, автоматизувати взаємодію з читачами та підвищити конкурентоспроможність українського книговидавництва у глобальному інформаційному просторі.

Актуальність кваліфікаційної роботи зумовлена необхідністю розробки сучасних та функціональних вебресурсів для продажу книжкової продукції, які здатні забезпечити безперебійний доступ користувачів до освітніх та культурних надбань у складних соціально-економічних умовах.

Вибір реалізації саме вебзастосунку базується на принципах кросплатформленості, адже він забезпечує максимальне охоплення аудиторії, доступ до ресурсу можливий з будь-якого пристрою через браузер без потреби в попередній інсталяції. Ще однією причиною є можливість інтеграції сучасних методів пошукової оптимізації, що критично важливо для залучення нових клієнтів в умовах високої конкуренції.

Саме завдяки веброзробці можна створити унікальний, інтуїтивно зрозумілий інтерфейс, що впливає на візуальне сприйняття продукту клієнтом та

отриманий в результаті користування позитивний досвід.

Також вебресурси дозволяють реалізувати ряд функцій, що покращують комфорт і зручність покупок. Прикладом таких функцій є можливість реєстрації користувачів, оформлення кошика покупок, забезпечення процесу оплати, аналітика та збір даних про клієнтів, а також кооперативна взаємодія з іншими секторами бізнесу. Вебплатформа також забезпечує гнучкість у оновленні контенту та асортименту в режимі реального часу.

Метою кваліфікаційної роботи є розв'язання прикладної задачі з проєктування та реалізації вебсайту магазину книг «BookFlow» на основі застосування сучасних інформаційних технологій та методів frontend та backend-розробки.

Передбачається, що в результаті даної кваліфікаційної роботи буде реалізовано повноцінний вебсайт, що відповідатиме всім поставленим технічним та користувацьким вимогам. Для досягнення мети визначено наступні завдання:

– провести аналітичний огляд існуючих рішень у сфері веброзробки для книжкової галузі;

– сформувати технічне завдання та вимоги до функціоналу сайту;

– спроєктувати структуру вебсторінок та архітектуру бази даних;

– реалізувати програмний комплекс, включаючи клієнтську та адміністративну частини;

– провести тестування розробленого вебсайту та підтвердити його працездатність.

Об'єктом дослідження є процес автоматизації роздрібної торгівлі та взаємодії з користувачами в галузі електронної комерції.

Предметом дослідження є програмно-технічні засоби створення вебсайту книжкового магазину.

Галузь можливого використання розробленої системи охоплює книжкові видавництва та мережі магазинів, що потребують автоматизації онлайн-продажів та розширення присутності на цифровому ринку.

1 ЗАГАЛЬНИЙ РОЗДІЛ

1.1 Аналітичний огляд існуючих рішень

Проведення процесу аналітичного огляду вже наявних конкурентних рішень є важливим етапом розробки власного продукту, адже саме аналіз, проведений на базі прототипування та розумінні ринку, здатен забезпечити нас необхідною інформацією про функціональні можливості та архітектурні особливості власної майбутньої системи.

Аналітичний огляд вирішено поділити на два етапи, спершу це огляд прямого конкурентного ринку, а другий етап – це аналіз підходів для створення електронного магазину.

На теперішній час в українському ринку електронної комерції книжкова галузь представлена як великими маркетплейсами, так і спеціалізованими онлайн-майданчиками окремих видавництв.

Для аналізу обрано три типи рішень, що є найбільш релевантними для предметної галузі вебсайту магазину книг «BookFlow»:

1. Yakaboo[1]. Дана платформа є прикладом глобального лідера ринку. Вона характеризується надзвичайно широким асортиментом та складною системою фільтрації. Водночас перевагою є інтеграція з багатьма платіжними системами та логістичними сервісами. Проте велика кількість контенту здебільшого перевантажує інтерфейс користувача, котрий не має достатньо технічних навиків.

2. Vivat[2]. Дане видавництво є прикладом спеціалізоваої онлайн-крамниці. Схожі ресурси такого типу зосереджені на власному продукті, мають високий рівень візуалізації та зручну навігацію, проте часто обмежені функціоналом лише одного бренду.

3. Книгарня Є[3]. Наочний приклад втілення локальної книгарні. Платформа поєднує в собі переваги онлайн-каталогу та мережі фізичних точок видачі. Ключовою особливістю є акцент на україномовній літературі та системі

лояльності для постійних читачів.

Для отримання достовірної інформації про архітектурні особливості та програмно-технічну базу обраних об'єктів у роботі використано сучасні методи автоматизованого аналізу вебресурсів. Зокрема, за допомогою спеціалізованого інструменту Wappalyzer[4] здійснювалась ідентифікація використаних мов програмування, систем управління вмістом, JS-фреймворків, бібліотек та інтегрованих аналітичних сервісів. Даний підхід дозволив провести глибокий технічний аналіз прототипів без доступу до їх вихідного коду.

Аналіз показав що технологічний стек платформи «Yakaboo» побудований з використанням JavaScript-фреймворку – Vue.js, він забезпечує високу інтерактивність користувацького інтерфейсу. Для збірки та оптимізації коду застосовуються Webpack та Babel. Також продуктивність роботи підвищується за допомогою механізму Priority Hints та CDN-сервісу Cloudflare.

У свою чергу інтернет-магазин видавництва «Vivat» використовує стек на базі React та фреймворку Next.js. Це дозволяє реалізувати генерацію статичних сторінок, що є критично важливим для SEO-оптимізації та швидкості завантаження.

А от онлайн-платформа «Книгарня Є» на відміну від попередніх систем, використовує сучасний мережевий протокол HTTP/3, який забезпечує швидшу передачу даних. Безпека та доставка контенту також реалізовані через Cloudflare.

Щодо нефункціональних характеристик, огляд описаних систем показав потребу у створенні ресурсу, який поєднуватиме високу швидкість роботи, лаконічний дизайн та інтуїтивно зрозумілу адміністративну частину для оперативного управління контентом. На основі проведеного аналізу було визначено, що проєкт «BookFlow» має бути спрямований на спрощення шляху користувача від вибору книги до оформлення замовлення, а також забезпечений мінімалістичним інтерфейсом без візуального перевантаження.

Виконавши вище описаний огляд конкурентного ринку, другим етапом аналітичного дослідження є порівняння методів реалізації вебсайту. Загалом

сучасна веброзробка пропонує два основні підходи до створення онлайн-платформ:

– розробка з використанням конструкторів;

– індивідуальна розробка з використанням мов програмування та фреймворків.

Перший підхід, використання конструкторів, більш орієнтований на швидке розгортання ресурсу за допомогою готових шаблонів та візуальних редакторів. Перевагами такого вибору є мінімальний час запуску проєкту, наявність вбудованих платіжних та логістичних модулів, а також відсутність потреби у глибоких технічних знаннях. Однак, недоліки у свою чергу включають обмеженість у кастомізації унікальних функцій, надмірність коду, а це негативно впливає на швидкість завантаження, ну і також це є залежність від стороннього сервісу та регулярна абонплата, що може випливати зайвими витратами на етапі початкового запуску магазину [5].

Прикладом засобу розробки з використанням конструкторів можна представити платформу Wix (див. рис. 1.1).

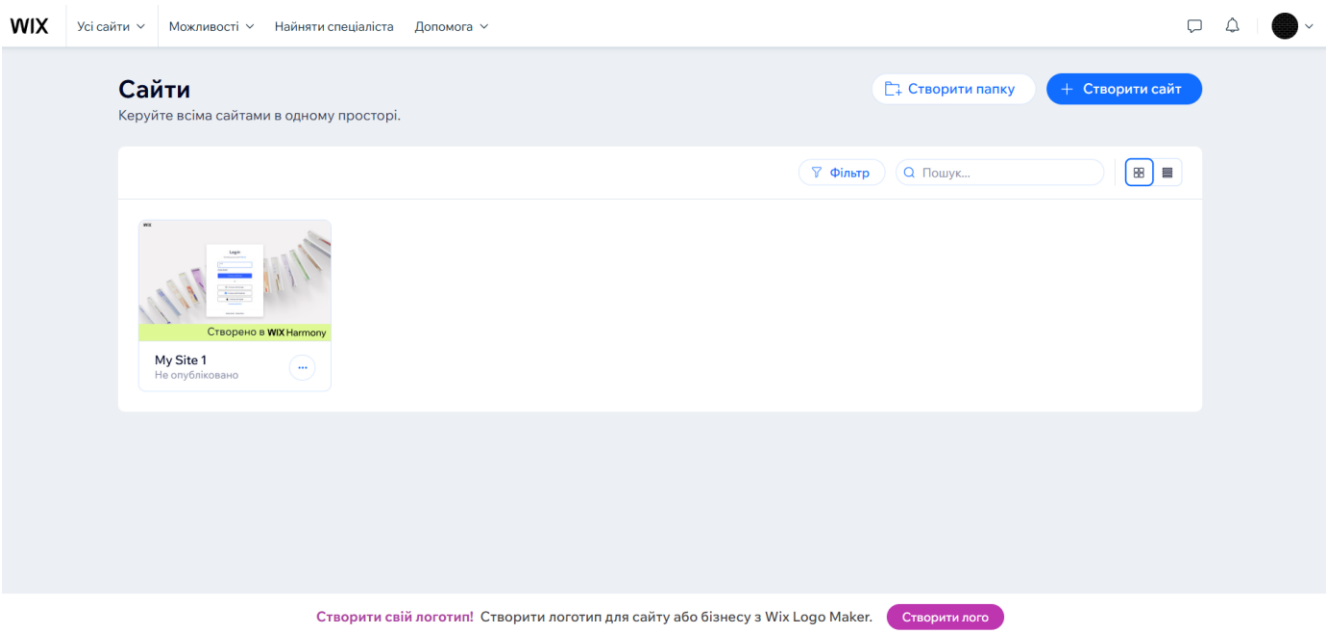


Рисунок 1.1 – Інтерфейс платформи Wix

Зазвичай інтерфейс таких систем інтуїтивно зрозумілий та простий у

користуванні. Ще однією перевагою використання схожих платформ є їхня можливість легко розширюватись і адаптовуватись до потреб бізнесу.

Індивідуальна розробка у свою чергу передбачає повністю самостійне написання коду з використанням сучасних стекових рішень та проєктування власної архітектури бази даних. Перевагами цього вибору є можливість реалізації будь-якої специфічної бізнес-логіки, висока продуктивність, завдяки відсутності зайвого функціоналу, а також можливість вибору та налаштування власних протоколів безпеки та шифрування даних.

Власна розробка магазину також дозволяє уникнути додаткових витрат через скорочення залежності від зовнішніх платформ і таким чином зберегти прибуток на початковій стадії запуску.

Звичайно вибір шляху самостійної розробки зазвичай вимагає більше зусиль, ширший діапазон технічних знань та часу на його створення, налаштування та підтримку. Однак, жертовність перерахованих потреб може бути цілком вигідною в довгостроковій перспективі, забезпечивши повний контроль над власним продуктом.

1.2 Технічне завдання

1.2.1 Найменування та область застосування

Найменування кваліфікаційної роботи – Розробка вебсайту магазину книг «BookFlow».

Стисла назва – «BookFlow».

Область застосування вебсайту – сектор електронної комерції, зокрема книжкова галузь.

Об’єкт використання – книжкові видавництва, як фізичних, так і онлайн-магазини, мережі книгарень та приватні підприємці, що займаються продажем літератури.

1.2.2 Призначення розробки

Основною метою розроблюваного програмного виробу є реалізація зручного та візуально приємного вебсайту магазину книг «BookFlow».

Експлуатаційне призначення вебсайту включає:

– автоматизацію процесів продажу книг;

– забезпечення безперебійної можливості потенційних покупців пошуку та придбання книг онлайн.

Функціональне призначення:

– реалізація зручного інтерфейсу навігації;

– створення легкого та зрозумілого механізму керування кошиком обраних товарів та процесу оформлення замовлень;

– забезпечення користувачів вебсайту особистим кабінетом клієнта та адміністративною панеллю для управління асортиментом;

– надання детального опису товару, візуального представлення товару та його характеристик для повноцінного ознайомлення майбутніх покупців;

– впровадження легкої але повноцінної системи фільтрації та пошуку книг;

– забезпечення надійної системи захисту вебсайту та конфіденційних даних клієнтів. Даний пункт можна реалізувати шляхом налаштування безпекових протоколів даних та використанням шифрування даних.

Узагальнене бачення мети розроблюваного вебсайту полягає у наданні користувачу можливості купівлі книги віддалено та зручно, а бізнесу – гнучкість керування товарами, без прив’язки до конкретного місця розташування.

1.2.3 Вимоги до функціоналу вебсайту

Під час розробки вебсайту магазину книг «BookFlow» визначено такі основні вимоги до функціоналу:

– система повинна забезпечити надійність зберігання конфіденційних

даних клієнтів у базі даних, використовуючи хешування паролів та шифрування вразливих полів;

– вхідні дані користувача при реєстрації та авторизації повинні проходити валідацію;

– система повинна мати окремо структурований каталог книг зі зручною навігацією та функцію фільтрації з декількома параметрами;

– повинен бути реалізований функціонал вподобаних товарів користувачів та історії попередніх купівель, але лише клієнтів з попередньо створеним акаунтом;

– система повинна реалізувати базовий функціонал можливостей адміністратора (CRUD) із можливістю завантаження зображень товару;

– пошукова система повинна підтримувати основні характеристики книги, а саме за назвою та автором, без необхідності закінчення ключових слів;

– система повинна забезпечити адаптивний дизайн, який зможе підтримувати інтерфейс вебсайту, як у форматі мобільної версії, так і в режимі десктопних платформ;

– функції реєстрації та авторизації користувачів повинні підтримувати зовнішні системи верифікації за допомогою Google OAuth;

– вебсайт повинен забезпечити зворотній зв'язок з клієнтами для вирішення проблем та налагодження питань, які можуть виникнути в процесі. Зворотній зв'язок може передбачати телефон гарячої лінії, електронну пошту або ж соціальні мережі.

1.2.4 Вимоги до програмної документації

По завершенню розробки вебсайту має бути підготовлений такий пакет документації:

– загальний перелік відомостей про можливості вебсайту та його ключових характеристик;

– інструкція з розміщення програмного продукту в Інтернеті. Вона складається з опису покрокового розміщення вебсайту на хостинговій платформі, налаштування характеристик вебсервісу та подальша підтримка хостингової платформи;

– інструкція з обслуговування та наповнення вебсайту, а саме опис процесів підтримки безперебійності роботи сайту та впровадження оновлень;

– інструкція з популяризації та підтримки сайту. Дана інструкція міститиме опис стратегій оптимізації пошукової видимості сайту та інтеграцію з різноманітними соціальними мережами;

– інструкція щодо налаштування бази даних та з’єднання з нею;

– перелік усіх ендпоінтів вебсайту, їх функціонал та параметри виклику;

– інструкція щодо комунікації з клієнтами, обробки замовлень, опис правил додавання нових товарів та зміни наявних.

1.2.5 Техніко-економічні показники

Розрахунок техніко-економічних показників включають ресурси, які були витрачені на розробку вебсайту. Основні параметри розрахунків:

– тривалість робочого часу;

– трудові витрати;

– машинний час.

Детальніше техніко-економічні показники подано в розділі 4 та представлено окремим плакатом у графічній частині кваліфікаційної роботи 2026.КВР.122.421.12.00.00 ТБ.

1.2.6 Стадії та етапи розробки

Процес розробки вебсайту магазину книг «BookFlow» передбачає проходження таких етапів:

– аналіз та планування. Початковий етап розробки, який передбачає

дослідження предметної області, формування бажаного функціоналу та вимог щодо нього, визначення технологічного стеку роботи, а також даний етап включає планування архітектури вебсайту та його дизайн;

– моделювання. На цьому етапі визначається архітектура вебсайту, проєктуються схеми взаємодії між компонентами клієнтської та серверної частин, а також структури бази даних. Використовуючи спеціалізовані платформи для проектування такі як Figma[6] та Canva будується користувацький інтерфейс разом з графічними елементами та бажаною палітрою кольорів;

– реалізація. Найважливіший етап розробки, який поділяють на дві частини: серверну та клієнтську.

Спершу розробляється серверна частина (Backend), вона включає налаштування середовища розробки, підключення бази даних, формування бізнес-логіки, розробка ендпоінтів для комунікації з клієнтською частиною, налаштування обробки вхідних даних та їхня передача в БД.

Після розробки серверної частини йде реалізація клієнтської (Frontend), тобто створення інтерфейсу користувача, верстка статичних та динамічних об’єктів, інтеграція із серверною частиною, а також написання адаптивного дизайну під різні пристрої задля приємного користувацького досвіду.

– тестування та налагодження. Не менш важливим етапом є проведення тестів, які визначають рівень задоволення вимог до вебсайту, перевірку коректності роботи протоколів безпеки, тестування сумісності. У випадках невідповідності очікуваним результатам виконується налагодження проблем та усунення недоліків.

– впровадження (реліз). Цей етап передбачає фінальний огляд роботи сайту, підготовку до розгортання на хостинговій платформі та перевірку коректності роботи вебсайту після релізу.

– підтримка. Фінальна стадія розробки, яка відповідає за супровід роботи вебсайту після його релізу. Має на меті усунення проблем, які можуть виникати в майбутньому, а також відповідає за впровадження оновлень та розширення

функціоналу, за потреб бізнесу.

1.2.7 Порядок тестування та прийому

Перевірка працездатності системи здійснюється у кілька етапів:

1. Функціональне тестування, тобто перевірка на коректність роботи всіх функціональних можливостей вебсайту, а саме пошук, фільтрація, реєстрація, авторизація тощо.

2. Нефункціональне тестування, яке перевірятиме відповідність визначеним вимогам рівня безпеки, швидкодії виконання функціональних параметрів, часу завантаження ресурсів та їх рівень оптимізації.

3. Тестування на сумісність. Воно включає огляд роботи вебсайту в різноманітних веббраузерах, таких як Google Chrome, Safari, Microsoft Edge тощо. А також перевірку адаптивності вебсторінок на пристроях різних розмірів.

4. Користувацьке тестування. Дане тестування передбачає залучення сторонніх людей в ролі користувачів вебсайту для незалежної оцінки користувацького досвіду та отримання таким чином зворотного відгуку.

5. Приймальне тестування. Саме це є вирішальним етапом тестування, яке виконується безпосередньо замовником. На даному етапі перевіряється готовність розроблюваного продукту та його відповідність поставленим вимогам.

Прийом реалізованого вебсайту проводиться комісією у складі двох осіб, один з яких є представником Замовника, а інший – Виконавця. Прийом проводиться у такій послідовності:

- презентація реалізованого проєкту Виконавцем;
- демонстрація роботи вебсайту в реальному часі;
- проведення контрольних сценаріїв тестування;
- відповіді на запитання та зауваження комісії.

2 РОЗРОБКА ТЕХНІЧНОГО ТА РОБОЧОГО ПРОЄКТУ

2.1 Розробка структури сайту і вебсторінок

Проєктування архітектури вебсайту магазину книг «BookFlow» базується на аналізі сценаріїв взаємодії кінцевих користувачів із системою. Головним завданням етапу є створення інтуїтивно зрозумілого та функціонального інтерфейсу, що забезпечить безперешкодний доступ до ключових сервісів платформи.

При відкритті вебсайту, відвідувача повинні зустрічати наступні базові компоненти:

– навігаційна панель: логотип вебсайту, перелік категорій книг, пошукова стрічка та іконки швидкого доступу (вподобані товари, особистий кабінет, кошик);

– інформаційний блок, який включатиме центральний банер для висвітлювання актуальних пропозицій та новинок;

– каталог продукції, що являє собою структурований перелік товарних одиниць із можливістю швидкої взаємодії через картку товару.

Для мінімізації кількості кроків у процесі купівлі, механізм додавання книг до списку вподобань або кошика реалізовано безпосередньо на рівні картки товару. Модуль «Кошик» повинен бути спроєктованим як модальне вікно з функціоналом динамічного перерахунку загальної вартості, редагуванням кількості позицій та видаленням товарів.

У випадку додаткового бажання користувачем ознайомитися з товаром має бути реалізована персональна сторінка продукту, яка міститиме:

– назву книжкової продукції;

– зображення обкладинки книги;

– описові характеристики товару;

– елемент ініціації замовлення (кнопка придбання).

Ключову роль у функціонуванні вебсайту повинен відігравати механізм розділення доступного функціоналу для користувача та адміністратора. Для відслідковування цієї логіки передбачається процес реєстрації та авторизації користувачів.

Процес ідентифікації реалізовуватиметься через багаторівневий механізм реєстрації, що гарантує верифікацію особи. Користувачам пропонуватиметься два методи реєстрації:

– класичний метод, який матиме на меті внесення персональних даних (прізвище, ім’я, пароль) із обов’язковою верифікацією через код підтвердження, що буде надіслано на електронну пошту;

– спрощений метод, тобто інтеграція протоколу Google OAuth[7], що дозволяє прискорити процес створення акаунту та покращити користувацький досвід.

Також елемент реєстрації/авторизації важливий тому що лише після ідентифікування особи клієнт зможе перейти на сторінку оформлення замовлення. Дана сторінка повинна включати наступні функціональні блоки:

– перелік обраних книг;

– фінальна сума замовлення;

– блок введення реквізитів отримувача;

– інтегрований модуль вибору локації доставки із використанням Public API логістичного оператора «Нова Пошта» для автоматизації вибору відділень;

– інформаційне застереження щодо правил оплати та повернення.

Функціональний простір адміністратора, у свою чергу, орієнтований на управління контентом та складськими залишками, тож основні операційні можливості повинні включати:

– створення нового товару;

– редагування властивостей наявного товару;

– архівування неактуальних товарів.

Відповідно, вікно для створення книги повинно містити:

– поле вводу назви книги;

– поле вибору автора;

– поле вибору категорії;

– поле вибору видавництва;

– поля для вводу додаткової інформації (ISBN, кількість сторінок, рік видання, ціна, опис, кількість залишку на складі);

– скринька завантаження зображення книги;

– кнопка підтвердження створення.

Однак, для забезпечення чистоти у роботі адміністратора, система повинна обмежити дозвіл на здійснення покупок та використання клієнтських сервісів (список вподобань).

На основі викладеного переліку вимог, було спроєктовано структурну схему вебсайту «BookFlow», яка подана окремим плакатом у графічній частині 2026.КВР.122.421.12.00.00 СС.

2.2 Створення та верстка сторінок сайту

Процес розробки клієнтської частини вебсайту магазину книг «BookFlow» реалізується на основі бібліотеки React, яка дозволяє створювати динамічні односторінкові застосунки (SPA) та поділяти інтерфейс на незалежні компоненти, котрі можуть повторно використовуватися у різних частинах системи. Завдяки такому підходу значно спрощується підтримка програмного коду, забезпечується модульність архітектури та дозволяється швидше масштабувати проєкт у майбутньому.

Для забезпечення високої продуктивності розробки та оптимізації фінальної збірки обрано інструмент Vite[8]. Його особливістю є використання нативних модулів браузера для швидкого запуску середовища розробки, тим самим забезпечується моментальне оновлення компонентів без повного перезавантаження сторінки. Завдяки цьому значно прискорюється процес верстки

інтерфейсу та тестування змін у режимі реального часу.

Для реалізації клієнтської частини проєкту потрібна наступна послідовність дій:

1. Підготовка робочого середовища шляхом встановлення платформи Node.js та менеджера пакетів npm. Конфігурація інтегрованого середовища розробки WebStorm та впровадження інструментів статичного аналізу коду для дотримання єдиного стандарту написання коду та запобігання синтаксичним помилкам [9].

2. Ініціалізація базової структури застосунку, використовуючи інструмент збірки Vite. Налаштування конфігураційного файлу vite.config.js для підтримки версії React, забезпечення коректної обробки активів та інтегрування препроцесор Sass[10] для розширення можливостей стилізації інтерфейсу.

3. Виконання декомпозиції інтерфейсу на незалежні блоки. Розроблення глобальних структурних компонентів, таких як навігаційна панель (Header) та нижня частина сайту (Footer). Створення ключових елементів інтерфейсу:

- картки товарів;
- інтерактивних банерів;
- системи фільтрації каталогу.

4. Налаштування взаємодії між компонентами. Для управління локальними станами окремих компонентів використовуватимуться хуки useState та useReducer. Для роботи з глобальними даними, які потребують доступу з різних частин дерева компонентів (стан авторизації користувача, вміст кошика тощо), впровадиться React Context API[11]. Це дозволить створити централізовані сховища даних, котрі забезпечать синхронізацію інтерфейсу в реальному часі при зміні станів.

5. Створення спеціалізованих форм для збору даних від користувача (реєстрація, авторизація, оформлення замовлення). Для цього необхідно інтегрувати бібліотеки React Input Mask та React Number Format для контролю формату введення.

Також важливим є впровадження алгоритмів валідації на стороні клієнта для перевірки коректності e-mail, міцності паролів та обов'язковості заповнення полів.

6. Реалізація механізмів керування станом обраних товарів, кошика і збереження ідентифікаторів сесії. Для цього доцільно використовувати Cookies[12], адже на відміну від локального сховища вони дозволяють:

– зберігати стан кошика та списку улюблених навіть після закриття браузера;

– синхронізувати стан клієнтської частини з серверними налаштуваннями сесії.

7. Формування сервісного шару застосунку для комунікації з сервером. Використовуючи бібліотеку Axios[13], створюється базовий екземпляр клієнта, налаштовується маршрутизація запитів до REST API та підготовка механізму обробки HTTP-заголовків для передачі JWT-токенів у процесі майбутньої інтеграції з серверною частиною.

Додатково для покращення взаємодії користувача з інтерфейсом варто впровадити систему миттєвих сповіщень React Hot Toast та механізм діалогових вікон SweetAlert2. Це дозволить оперативно інформувати користувача про результати виконаних дій, помилки або успішне завершення операцій.

Для створення початкової заготовки проєкту необхідно відкрити інтегрований термінал у WebStorm та виконати наступну команду:

npm create vite@latest bookflow-frontend

Під час виконання команди у діалоговому вікні потрібно обрати фреймворк React та варіант JavaScript + SWC для забезпечення максимально швидкої компіляції та роботи застосунку.

Після цього Vite автоматично згенерує початкову структуру проєкту (див. рис. 2.1), яка міститиме базові каталоги, конфігураційні файли та стартові компоненти. Варто зазначити, що сформована архітектура є модульною і чітко розмежовує вихідний код застосунку, статичні ресурси та файли конфігурації

збірки, що є критично важливим для подальшого масштабування вебсайту магазину книг.

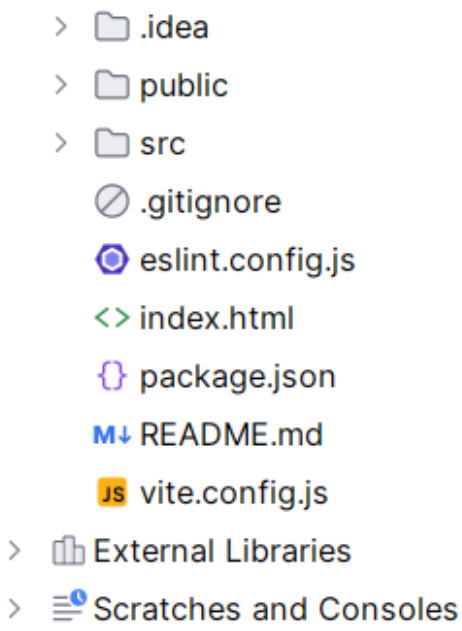


Рисунок 2.1 – Початкова структура проєкту на базі Vite

Далі необхідно перейти до папки проєкту та встановити всі залежності, описані у файлі package.json, виконавши покроково наступні команди:

cd bookflow-frontend

npm install

Після успішного встановлення пакетів запускається локальний сервер розробки командою:

npm run dev

Таким чином, Vite запускає локальний сервер, який забезпечує миттєве відображення змін у вікні веббраузера без необхідності повного перезавантаження сторінки. Це значно спрощує процес тестування та налагодження інтерфейсу.

Кореневою точкою входу є файл index.html (див. рис. 2.2), який у проєктах Vite розміщується безпосередньо у кореневій папці проєкту, що добре помітно на рисунку 2.1.

<> index.html ×

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="UTF-8" />
    <link rel="icon" type="image/svg+xml" href="/favicon.svg" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <title>BookFlow</title>
  </head>
  <body>
    <div id="root"></div>
    <script type="module" src="/src/main.jsx"></script>
  </body>
</html>
```

Рисунок 2.2 – Структура файлу index.html

Як подано вище, файл index.html містить єдиний елемент-контейнер:

Саме у цей блок за допомогою JavaScript-файлу main.jsx буде здійснюватися відмальовування всього інтерфейсу вебсайту.

Далі для зручності роботи налаштовуємо запуск проєкту безпосередньо через інтерфейс IDE. Для цього створюємо конфігурацію запуску npm, вказавши команду dev. Це дає дасть можливість запускати вебзастосунок не через термінал, а натисканням однієї клавіші у WebStorm. IDE автоматично виконає потрібну команду та дозволить зручно керувати запуском і зупинкою проєкту через графічний інтерфейс.

Варто зазначити, що Vite працює швидше за старі інструменти для розробки вебзастосунків. Після першого встановлення всіх необхідних пакетів повторний запуск проєкту відбувається значно швидше, оскільки система зберігає залежності у кеші та повторно їх використовує. Завдяки цьому не потрібно щоразу заново завантажувати ресурси з мережі, що робить роботу з проєктом більш швидкою та зручною.

2.3 Розробка структури бази даних сайту

Для забезпечення надійного зберігання та обробки даних вебсайту

магазину книг «BookFlow» обрано реляційну систему управління базами даних (СКБД) PostgreSQL[14]. Вибір обґрунтований підтримкою стандартів SQL, високою продуктивністю при роботі зі складними запитами та можливістю масштабування системи.

База даних проєкту будується за реляційним принципом, де інформація організовується у вигляді взаємопов'язаних таблиць. Кожна таблиця відповідає окремій сутності предметної області та містить набір полів із визначеними типами даних. Для забезпечення зв'язків між сутностями використовуються первинні та зовнішні ключі, що дозволяє підтримувати цілісність інформації та уникати дублювання даних.

Особливістю розробленої структури є модульний підхід до організації даних, завдяки якому окремі сутності можуть незалежно розширюватися без суттєвих змін загальної архітектури системи. Крім цього, база даних забезпечує підтримку механізмів авторизації користувачів, обробки замовлень, збереження інформації про книги та їх категоризацію.

При проєктуванні бази даних було сформовано такі етапи:

1. Визначення основних сутностей предметної області;
2. Розробка структури кожної таблиці та визначення типів даних для полів;
3. Встановлення зв'язків між таблицями;
4. Нормалізація таблиць для усунення надлишковості даних.

Спершу формується структура бази даних вебсайту магазину книг «BookFlow», яка складатиметься із сутностей:

- User (збереження записів користувачів);
- Book (збереження записів книг);
- Author (збереження записів авторів);
- Category (збереження записів категорій);
- Publisher (збереження записів видавництв);
- Order (збереження записів замовлень);

– OrderItem (збереження записів позицій замовлення).

Далі на основі аналізу всіх сутностей – формується наповнення кожної з них. Таблиця User міститиме записи з такими полями:

– username (повне ім’я користувача). Тип даних String;

– email (електронна адреса користувача). Тип даних String;

– password (хеш-пароль користувача). Тип даних String;

– role (роль користувача). Тип даних Enum;

– orders (замовлення користувача). Тип даних Object;

– isVerified (статус активізації акаунту користувачем). Тип даних Boolean;

– authProvider (метод верифікації користувача). Тип даних Enum.

Таблиця Book складатиметься із записів з такими полями:

– title (назва книги). Тип даних String;

– author (автор книги). Тип даних Object;

– price (ціна книги). Тип даних Integer;

– publisher (видавництво). Тип даних Object;

– category (категорія книги). Тип даних Object;

– isbn (міжнародний стандартний номер книги). Тип даних String;

– numberOfPages (кількість сторінок). Тип даних Integer;

– yearPublished (рік видання). Тип даних Integer;

– quantity (кількість на складі). Тип даних Integer;

– description (опис книги). Тип даних String;

– imageUrl (гіперпокликання на зображення книги). Тип даних String;

– cloudinaryPublicId (ідентифікаційний ключ зображення книги). Тип даних String;

– isArchived (статус архівації книги). Тип даних Boolean;

– inStockStatus (статус книги «в наявності»). Тип даних Integer.

Таблиця Author міститиме записи з такими полями:

– fullName (повне ім’я автора). Тип даних String;

– bio (коротка біографія автора). Тип даних String;

– books (список книг, що належать даному автору). Тип даних Object.

Таблиця Category складатиметься із записів з такими полями:

– categoryName (назва категорії книг). Тип даних String;

– books (список книг, що відносяться до цієї категорії). Тип даних Object.

Таблиця Publisher міститиме записи з такими полями:

– publisherName (назва видавництва). Тип даних String;

– books (список книг, випущених цим видавництвом). Тип даних Object.

Таблиця Order складатиметься із записів з такими полями:

– user (користувач, який здійснив замовлення). Тип даних Object;

– orderDate (дата та час створення замовлення). Тип даних LocalDateTime;

– totalAmount (загальна сума замовлення). Тип даних Integer;

– recipientName (ім’я отримувача замовлення). Тип даних String;

– recipientPhone (номер телефону отримувача). Тип даних String;

– shippingAddress (адреса доставки). Тип даних String;

– items (перелік позицій у замовленні). Тип даних Object;

– status (поточний статус замовлення). Тип даних Enum.

Так як таблиця Orders містить інформацію про поточний статус кожної покупки. Життєвий цикл замовлення та дозволені алгоритми зміни його статусів детально проілюстровані за допомогою UML-діаграми станів, що подана у графічній частині кваліфікаційної роботи 2026.КВР.122.421.12.00.00 ДС.

Таблиця OrderItem міститиме записи з такими полями:

– book (книга, що входить до позиції замовлення). Тип даних Object;

– quantity (кількість примірників книги у даній позиції). Тип даних Integer;

– price (ціна книги на момент здійснення замовлення). Тип даних Integer;

– order (ідентифікатор головного замовлення, до якого належить позиція). Тип даних Object.

У полях role, authProvider та status використовуються перелічувані типи даних Enum, це дозволяє обмежити набір допустимих значень та забезпечити цілісність даних.

Для візуалізації спроєктованої структури та логічних зв’язків між сутностями було розроблено ER-діаграму бази даних вебсайту магазину книг «BookFlow», яку подано окремим плакатом у графічній частині 2026.КВР.122.421.12.00.00 БД.

2.4 Програмування сайту

2.4.1 Написання клієнтської частини

Попередньо створена папка проєкту (див. рис. 2.1) для подальшої розробки клієнтської частини вебсайту магазину книг «BookFlow» базується на використанні бібліотеки React у поєднанні з інструментом збірки Vite, що забезпечує високу швидкість розробки та оптимізацію фінального коду.

Варто вказати, що React являє собою декларативну JavaScript-бібліотеку, використовувану для побудови користувацьких інтерфейсів. В основі архітектури розроблюваного вебсайту лежить компонентний підхід, тобто такий, який передбачає розбиття складного користувацького інтерфейсу на набір незалежних, придатних для багаторазового використання блоків – компонентів. Кожен компонент містить власну логіку, структуру відображення та стилі.

Така організація структури допомагає уникнути дублювання коду, помітно спрощує пошук помилок, тестування і полегшує подальше розширення функціоналу вебсайту.

Важливим аспектом побудови користувацького інтерфейсу на React є використання JSX. JavaScript XML – це синтаксис, який дозволяє описувати інтерфейс користувача за допомогою тегів, схожих на HTML, прямо всередині JavaScript-коду. Але на відміну від статичного HTML, JSX має повну функціональність мови JavaScript, саме це дає можливість не лише створювати

структуру сторінки, а й динамічно підставляти дані, використовувати умови та виводити списки елементів.

Також важливою особливістю у реалізації вебсайту є використання концепції односторінкового застосунку – SPA[15] (Single Page Application). Завдяки цьому підходу завантаження базової HTML-сторінки відбувається лише один раз, а подальша маршрутизація та оновлення контенту виконуються динамічно на стороні клієнта за допомогою JavaScript. Такий підхід не потребує повного перезавантаження сторінки при переходах між ними, і тим самим забезпечує плавність та швидкість роботи інтерфейсу, що робить роботу сайту більш зручною для користувача.

Висока продуктивність React також забезпечується завдяки Virtual DOM[16]. Це спеціальний механізм, який дозволяє зменшити навантаження на браузер і покращує продуктивність застосунку. А досягається дане покращення шляхом роботи React, де він обчислює різницю між попереднім та новим станом інтерфейсу в пам'яті, після чого точково оновлює лише ті елементи, які дійсно зазнали змін.

Архітектуру застосунку, тобто папку src (див. рис. 2.1) було розділено на логічні модулі (див. рис. 2.3) задля забезпечення чіткої ієрархії, читабельності коду та зручності подальшої підтримки і масштабування проєкту.

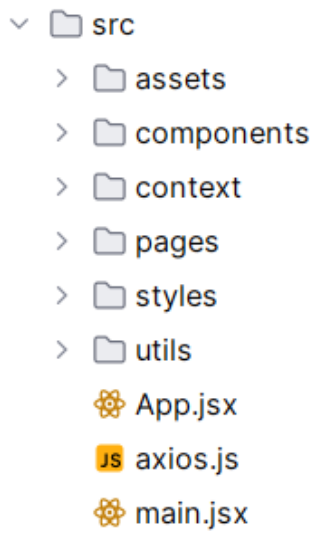


Рисунок 2.3 – Структура папки «src»

– папка «assets» міститиме статичні ресурси проєкту, такі як зображення, іконки та локальні шрифти;

– у папці «components» зберігатимуться UI-компоненти (кнопки, модальні вікна, картки товарів, Header, Footer), які не прив'язані до конкретного маршруту. В майбутньому їхня особливість буде в тому, що їх можна буде використовувати повторно;

– папка «pages» відповідатиме за компоненти сторінок, які відображаються залежно від поточного URL (головна сторінка, профіль користувача, оформлення замовлення тощо);

– у папці «context» міститимуться файли для налаштування глобального стану застосунку за допомогою React Context API[11], наприклад керування станом авторизації, кошика, категорій;

– папка «styles» зберігатиме глобальні файли стилів, SCSS-змінні та налаштування шрифтів, щоб забезпечити єдиний стиль вебсайту;

– папка «utils» служитиме директорією для допоміжних функцій, наприклад конфігурації роботи з файлами cookie для безпечного збереження даних сесії користувача.

Для організації взаємодії клієнтської та серверної частин вебсайту було створено окремий модуль для виконання HTTP-запитів – axios.js. Лістинг файлу «axios.js» подано в Додатку А кваліфікаційної роботи.

Точкою входу в застосунок є файл main.jsx (див. рис. 2.4). У ньому відбувається ініціалізація кореневого компонента React-застосунку – <App/> та його підключення до DOM-елемента за допомогою методу createRoot.

```
createRoot(document.getElementById( elementId: 'root')).render(
  <StrictMode>
    <App />
  </StrictMode>,
)
```

Рисунок 2.4 – Вміст вхідного файлу main.jsx

Задля підвищення стабільності та швидкого виявлення проблем – застосуток обгорнуто в компонент <StrictMode/>, який активовує додаткові перевірки React.

Компонент <App/> у свою чергу містить головну логіку навігації між сторінками вебсайту (див. рис. 2.5). У файлі визначено основні маршрути для всіх ключових сторінок, наприклад головної сторінки, каталогу, пошуку, особистого кабінету користувача тощо.

```
return (
    <CartProvider>
        <CartModalProvider>
            <AuthProvider>
                <AuthModalProvider>
                    <CategoryProvider>
                        <ScrollToTop />
                        <Header/>
                        <Toaster position="bottom-right" reverseOrder={false} />
                        <Routes>
                            <Route path="/" element={<MainPage/>}/>
                            <Route path="/category/:id" element={<CategoryPage/>}/>
                            <Route path="/search" element={<SearchResults/>}/>
                            <Route path="/profile" element={<ProfilePage/>}/>
                            <Route path="/register" element={<RegisterPage/>}/>
                            <Route path="/wishlist" element={<WishlistPage/>}/>
                            <Route path="/checkout" element={<CheckoutPage/>} />
                            <Route path="/book/:id" element={<ProductPage/>}/>
                            <Route path="/delivery" element={<DeliveryPage />} />
                            <Route path="/admin/orders" element={<AdminOrdersPage/>} />
                            <Route path="/archived-books" element={<ArchivedBooksPage/>}/>
                            <Route path="*" element={<NotFoundPage/>}/>
                        </Routes>
                        <Footer/>
```

Рисунок 2.5 – Вміст компоненту <App/>

Також у компоненті <App/> передбачено обробку неіснуючих маршрутів шляхом перенаправлення користувача на сторінку «404 Not Found».

Для відслідковування рівня доступності користувача до певного маршруту – підключено набір провайдерів, які дозволяють зберігати спільний стан (наприклад ролі авторизованого користувача, кошика, категорій) і надавати

доступ до цих станів будь-якому компоненту без необхідності передавати дані через props.

Окремо в App.jsx підключено компонент <Toaster/>, який відповідає за показ спливаючих вікон з повідомленнями. Він використовується для інформування користувача про виконані події, наприклад успішне створення нового товару або зміна статусу замовлення.

Базовим та основним маршрутом вебсайту є кореневий шлях, який перенаправляє користувача на головну сторінку сайту, реалізовану в компоненті <MainPage/> (src/pages/MainPage.jsx). Дана сторінка відображає користувачу головний інформаційний банер новин, верхнє навігаційне меню, перелік декількох книг для ознайомлення та нижнє поле для комунікації з додатковою інформацією (див. рис. 2.6).

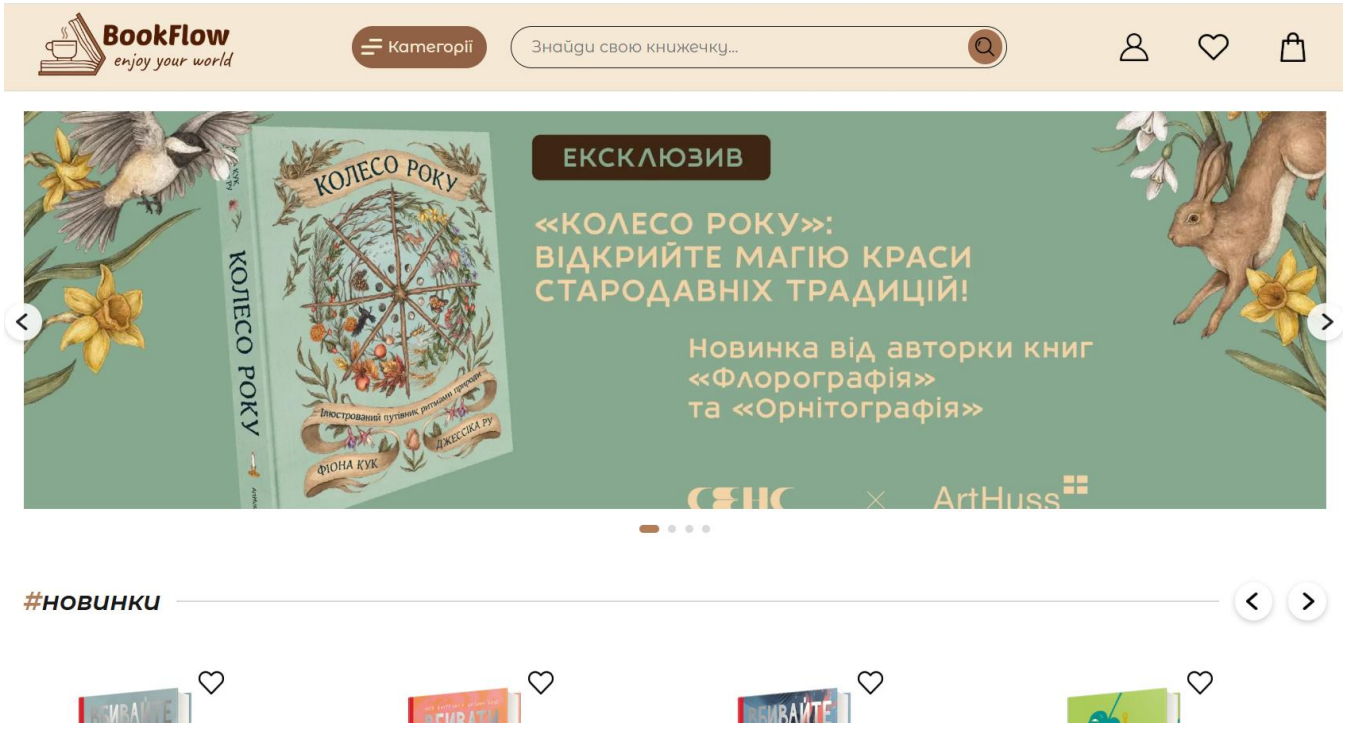


Рисунок 2.6 – Головна сторінка вебсайту «BookFlow»

Для того щоб виконати процес авторизації користувачу необхідно натиснути на відповідну іконку у правій верхній частині навігаційного меню, після цього відкриється модальне вікно входу (див. рис. 2.7).

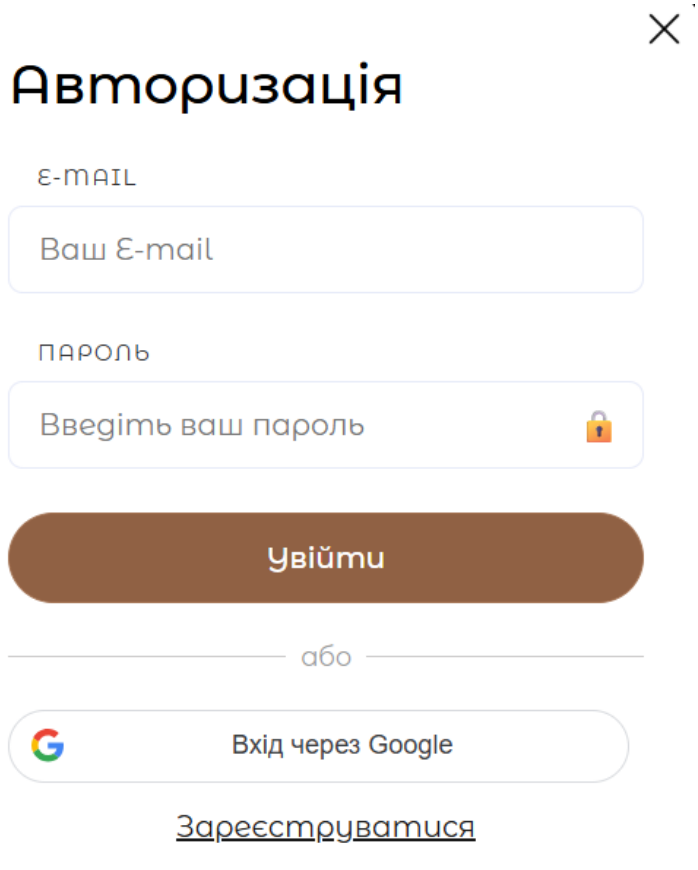


Рисунок 2.7 – Модальне вікно авторизації

Логіка компоненту <AuthModal/> передбачає дві можливості авторизації – класичним методом (введення електронної адреси користувача і пароль) та через Google OAuth. Другий метод представляє концепцію швидкого безпарольного входу через сторонній сервіс автентифікації Google. Це реалізовується взаємодією клієнтського компонента зі спеціалізованою бібліотекою React та сервером Google API.

У випадках відсутності акаунту – модальне вікно містить інтерактивне покликання для його створення. Після натискання на покликання вебсайт перенаправляє користувача на сторінку реєстрації зі шляхом «/register» (див. рис. 2.8), який відповідає компоненту <RegisterPage/> (src/components/RegisterPage.jsx).

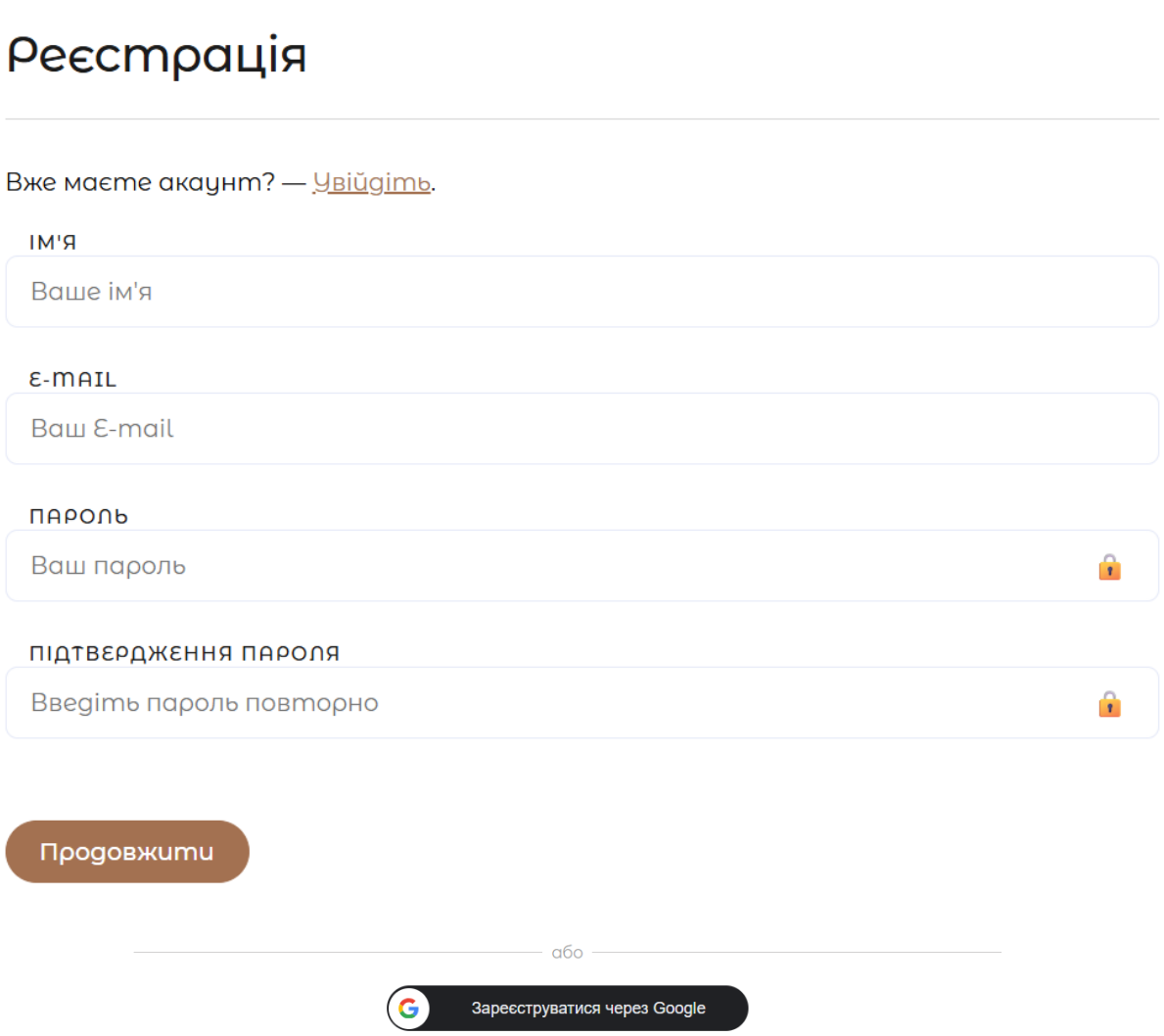


Рисунок 2.8 – Сторінка реєстрації

Частково логіку реєстрації було описано в пункті 2.1, але якщо деталізувати процес створення облікового запису, то він складається з двох варіантів підтвердження особи:

1. Верифікація за допомогою одноразового коду підтвердження пошти.

Для захисту від автоматизованих реєстрацій (ботів) та збереження автентичності даних користувачів, у системі реалізовано механізм двофакторної верифікації через код підтвердження, котрий надсилається на вказану користувачем електронну пошту.

Отже після заповнення форми реєстрації на сторінці <RegisterPage/> та проходження перевірки введених даних, обліковий запис користувача не створюється одразу. Натомість інтерфейс змінює свій стан, який відслідковується

змінною «step» і переходить в режим підтвердження електронної пошти, де додатково відмальовуються комірки для введення шестизначного коду підтвердження. Паралельно у цей момент серверна частина генерує випадковий код із обмеженим терміном дії та надсилає його на вказану електронну адресу користувача. Після отримання листа користувач вводить код у відповідні комірки, а клієнтська частина у свою чергу надсилає POST-запит на сервер для перевірки введених даних.

У разі успішного підтвердження коду – користувач автоматично перенаправляється на головну сторінку вебсайту. Якщо код введено неправильно або термін його дії вичерпався, то інтерфейс відобразить помилку, що означатиме необхідність повернути на попередній етап реєстрації та знову подати запит на отримання коду підтвердження.

2. Інтеграція протоколу Google OAuth.

Цей механізм швидкої реєстрації впроваджено завдяки використанню технології OAuth через Google та бібліотеки @react-oauth/google. Такий підхід дозволяє користувачам створювати обліковий запис за допомогою власного Google-акаунта.

Для реалізації даного механізму, основний файл main.jsx (див. рис. 2.4) було обгорнуто в компонент <GoogleOAuthProvider/>, який ініціалізовується за допомогою унікального клієнтського ключа «clientId». Потім після виконання процесу реєстрації Google повертає спеціальний токен, який клієнтська частина надсилає на серверний ендпоінт.

При успішній авторизації або ж реєстрації користувача перенаправляє на кореневий маршрут сайту – структура головної сторінки не відрізнятиметься для користувачів будовою та наповненням, лише компонент <Header/> (src/components/Header.jsx) матиме два різні стани відображення:

– для звичайного користувача (див. рис. 2.9);
– для адміністратора (див. рис. 2.10).

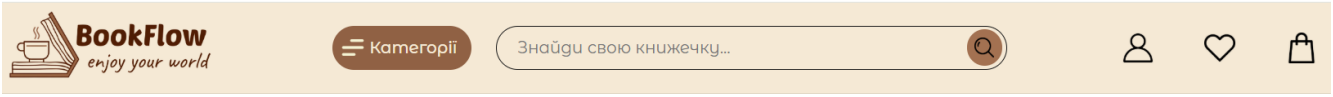


Рисунок 2.9 – Компонент <Header/> для звичайного користувача

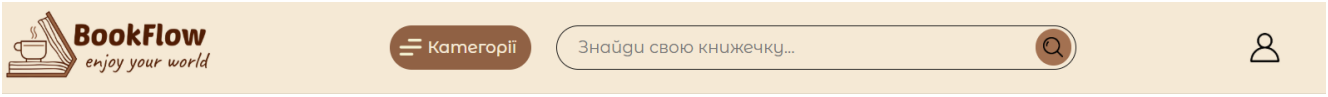


Рисунок 2.10 – Компонент <Header/> для адміністратора

Список доступних категорій книг відображається при наведені миші на відповідну кнопку компонента <Header/> (див. рис. 2.11).

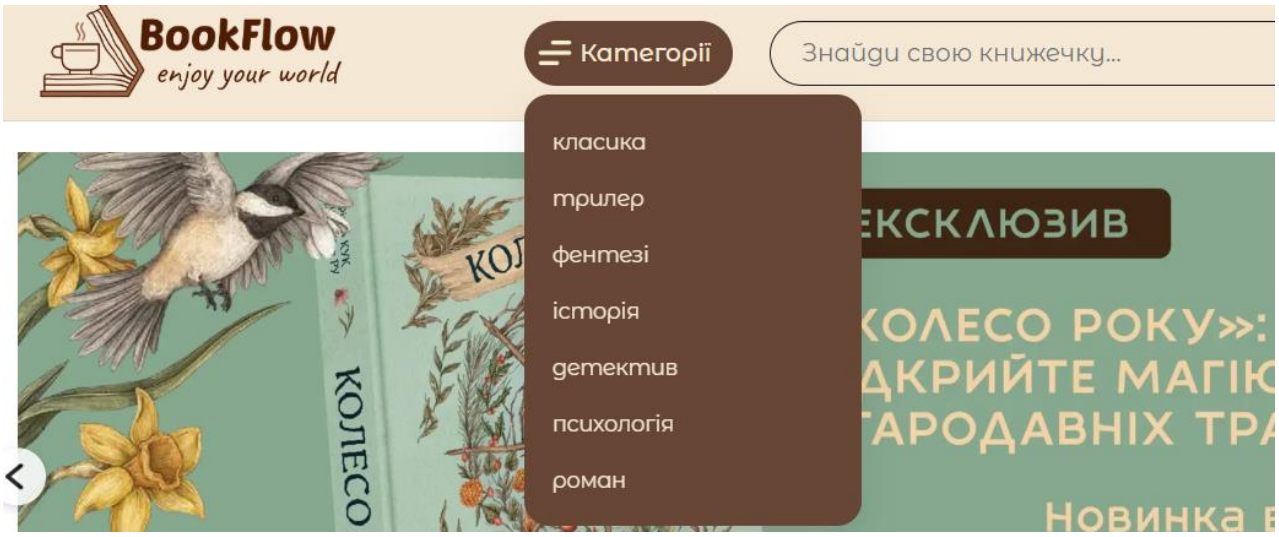


Рисунок 2.11 – Список категорій книг

Задля зменшення кількості запитів на серверну частину та покращення продуктивності вебсайту – взаємодію зі списком категорій винесено в окремий глобальний стан <CategoryContext/> (src/context/CategoryContext.jsx) за допомогою React Context API. Таким чином список категорій підтягується зі сторони серверного API для відображення лише раз – при запуску проєкту, після цього він зберігається на клієнтській стороні. Саме завдяки такій реалізації всі компоненти в подальшому можуть отримувати доступ до категорій без повторних HTTP-запитів.

При виборі категорії користувач переходить на динамічний маршрут

«/category/:id», де id – це унікальний ідентифікатором категорії. Компонент <CategoryPage/> (src/pages/CategoryPage.jsx) реалізовує перегляд книг за обраною категорією. Для цього він отримує параметр id з адресного рядка та завантажує відповідний список книг для подальшого відображення на сторінці (див. рис. 2.12).

Рисунок 2.12 – Компонент <CategoryPage/>

Наповнення компоненту <CategoryPage/> передбачає також взаємодію користувача з книгами за допомогою фільтрації та сортування задля покращення клієнтського досвіду користування.

У випадку натискання на картку товару спрацьовує компонент <ProductPage/> (src/pages/ProductPage.jsx), відреагувавши на динамічний елемент маршруту «/book/:id», де id – це унікальний ідентифікатором книги, перенаправляє користувача на персоналізовану сторінку товару.

Візуально інтерфейс сторінки поділено на дві частини:

– блок візуалізації та взаємодії (зображення книги, назва, автор, ціна, категорія, статус наявності, кнопка купити та вподобати);

– блок метаданих (додаткова інформація про книгу, анотація книги,

біографія автора).

В залежності від ролі користувача компонент <ProductPage/> відображатиме блок візуалізації та взаємодії по-різному для звичайного клієнта (див. рис. 2.13) та адміністратора (див. рис. 2.14).

Рисунок 2.13 – Компонент <ProductPage/> для звичайного користувача

Рисунок 2.14 – Компонент <ProductPage/> для адміністратора

Щодо блоку метаданих, то він є незмінним у відображенні незалежно від ролі користувача (див. рис. 2.15).

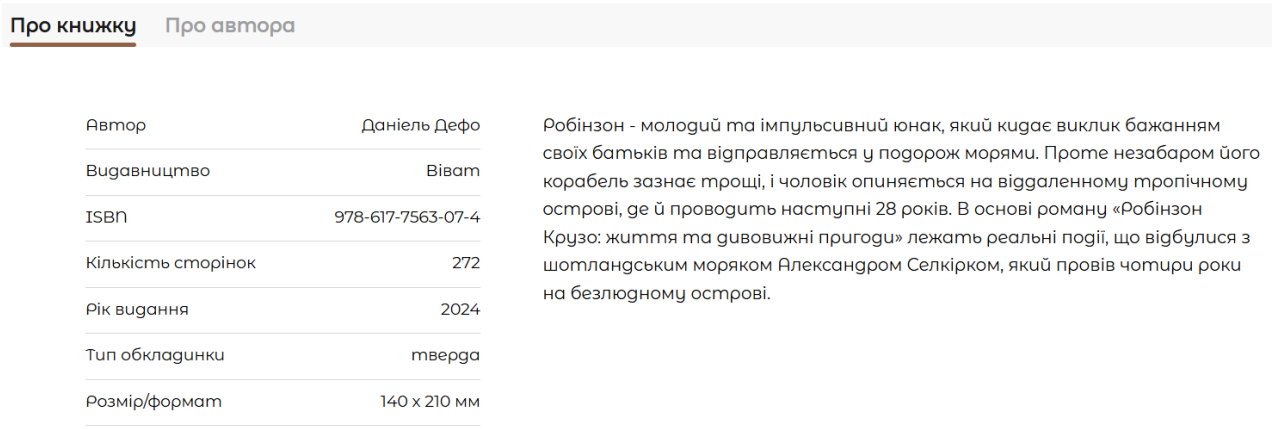


Рисунок 2.15 – Другий блок компонента <ProductPage/>

Для звичайних користувачів передбачено базовий функціонал взаємодії з каталогом:

1. Додавання товару до кошика.

При натисканні на кнопку «Купити» товар поміщається в корзину покупок та на іконці корзини в компоненті <Header/> (src/components/Header.jsx) з’являється ітеративний елемент, який інформує покупця про кількість позицій у кошику (див. рис. 2.16).

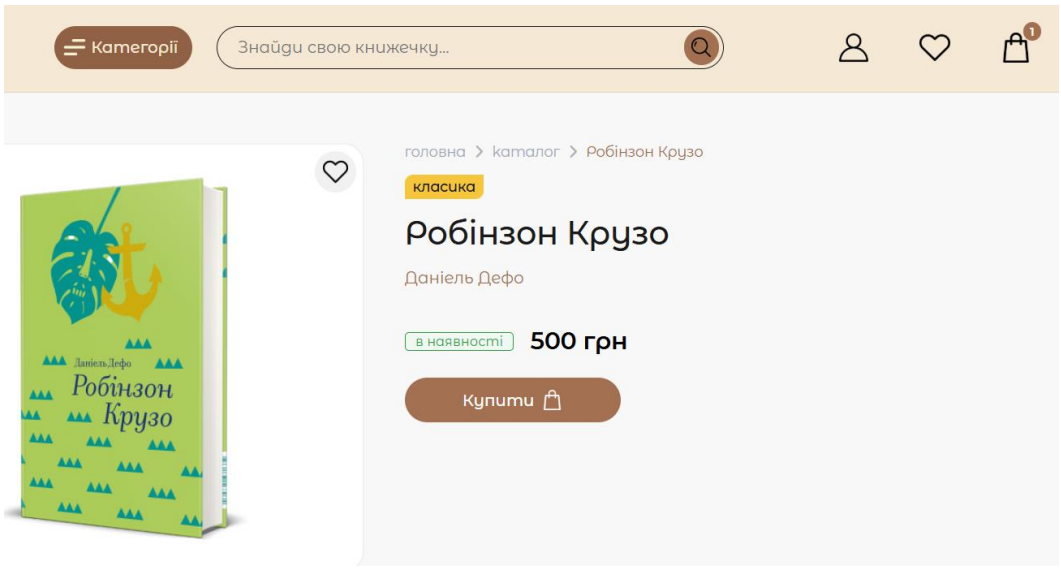


Рисунок 2.16 – Додавання товару в корзину

В той же час, щоб забезпечити доступ до оформлення замовлень тільки авторизованим користувачам – візуальна складова кошика відрізнятиметься лише

однією логічною кнопкою:

– «Авторизуйтесь, щоб купити» – для не авторизованого користувача;

– «Оформити замовлення» – для авторизованого користувача.

Як вже згадувалось вище, користувач з правами адміністратора не має можливості здійснювати покупки, тому і кнопка «Купити» на персональній сторінці книги відображається для нього недоступною (див. рис. 2.14).

2. Додавання товару в список улюблених.

За допомогою іконки сердечка на картці товару в користувача є можливість формувати власний список вподобаних книг. За реалізацію сторінки вподобаних книг відповідає компонент <WishlistPage/> (src/pages/WishlistPage.jsx). Загальна структура сторінки однакова як для авторизованих, так і не авторизованих користувачів. Відмінність полягає лише у боковій панелі, яка змінюється залежно від статусу користувача.

Якщо користувач не авторизований, масив вподобаних книг, а саме їхні ідентифікатори зберігаються на стороні клієнта за допомогою Cookies[12]. Таким чином, коли користувач відкриває сторінку вподобаних, застосунок зчитує збережені ідентифікатори книг і надсилає їх через запит на сервер щоб отримати повну інформацію про них (див. рис. 2.17).

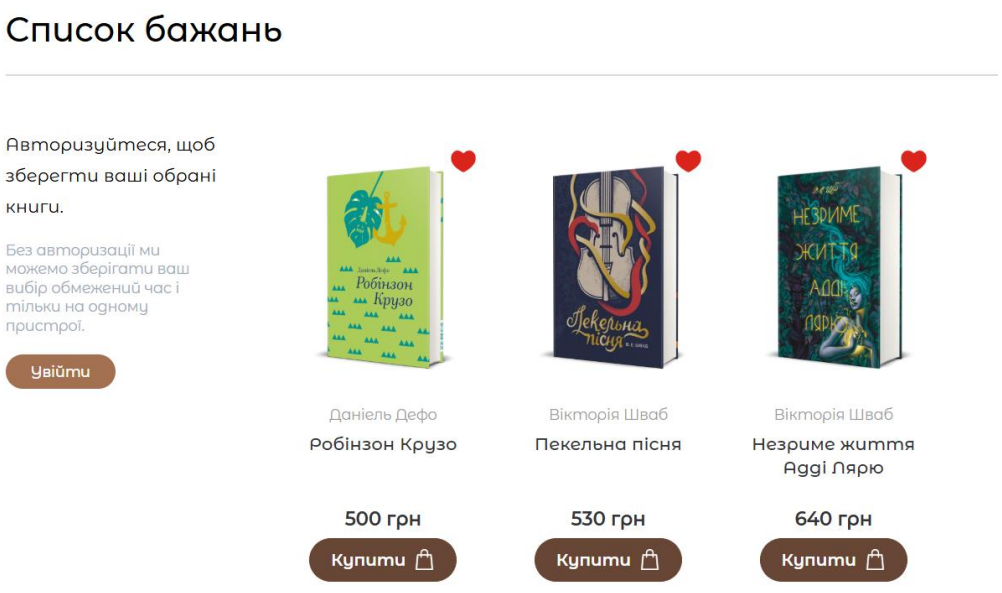


Рисунок 2.17 – Список бажань не авторизованого користувача

Для зареєстрованих користувачів синхронізація списку бажань відбувається через базу даних та захищений API-маршрут «/wishlist». При натисканні на іконку вподобання, клієнтський застосунок надсилає запит на сервер та одночасно оновлює глобальний стан користувача за допомогою Context API (див. рис. 2.18).

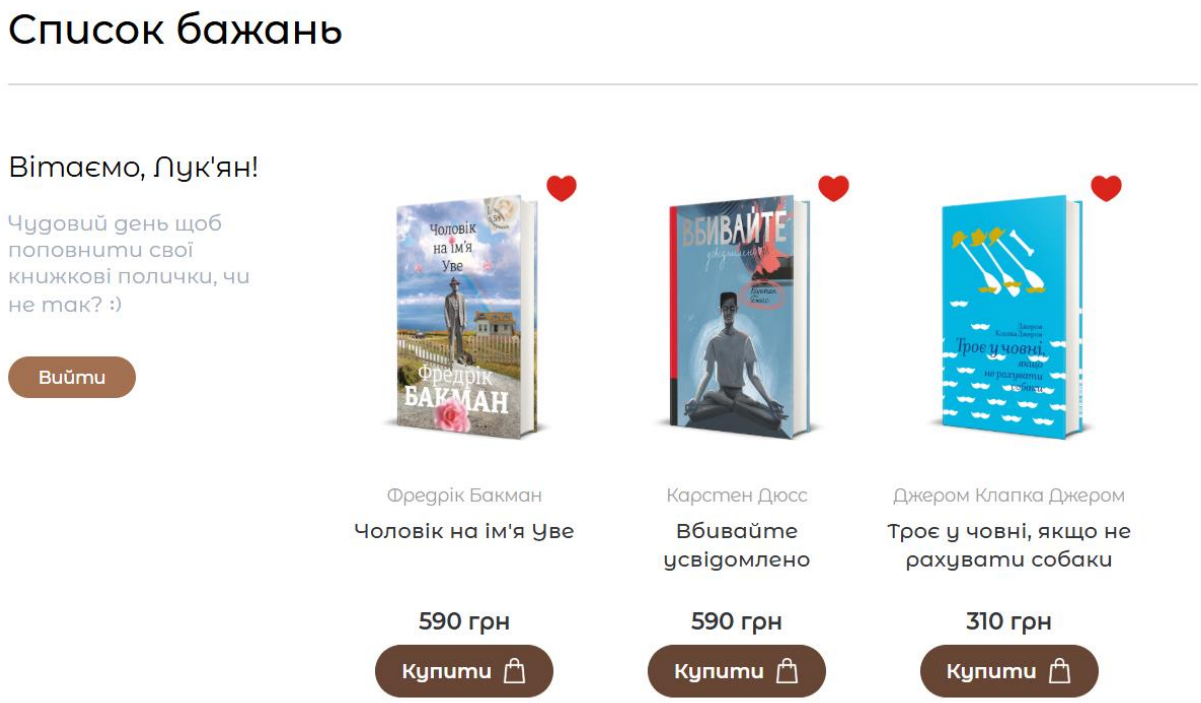


Рисунок 2.18 – Список бажань авторизованого користувача

Також інтерфейс сторінки «Список бажань» підтримує пагінацію, тобто книги автоматично розподіляються по сторінках по 10 елементів на кожній. Якщо користувач видаляє останню книгу з поточної сторінки, вебсайт автоматично перенаправляє його на попередню сторінку без небхідності у перезавантаженні вікна браузера.

Для користувачів з правами адміністратора клієнтська частина на персональній сторінці книги розширюється додатковими інструментами керування (див. рис. 2.14):

1. Редагування параметрів товару.

Адміністратор має можливість внести корективи в інформацію про книгу

викликавши модальне вікно з формою редагування (див. рис. 2.19). При відкритті модального вікна поля форми автоматично заповнюються поточними даними товару. Перед відправкою оновлених даних на сервер, клієнтська частина валідує введену інформацію, запобігаючи відправленню некоректних значень.

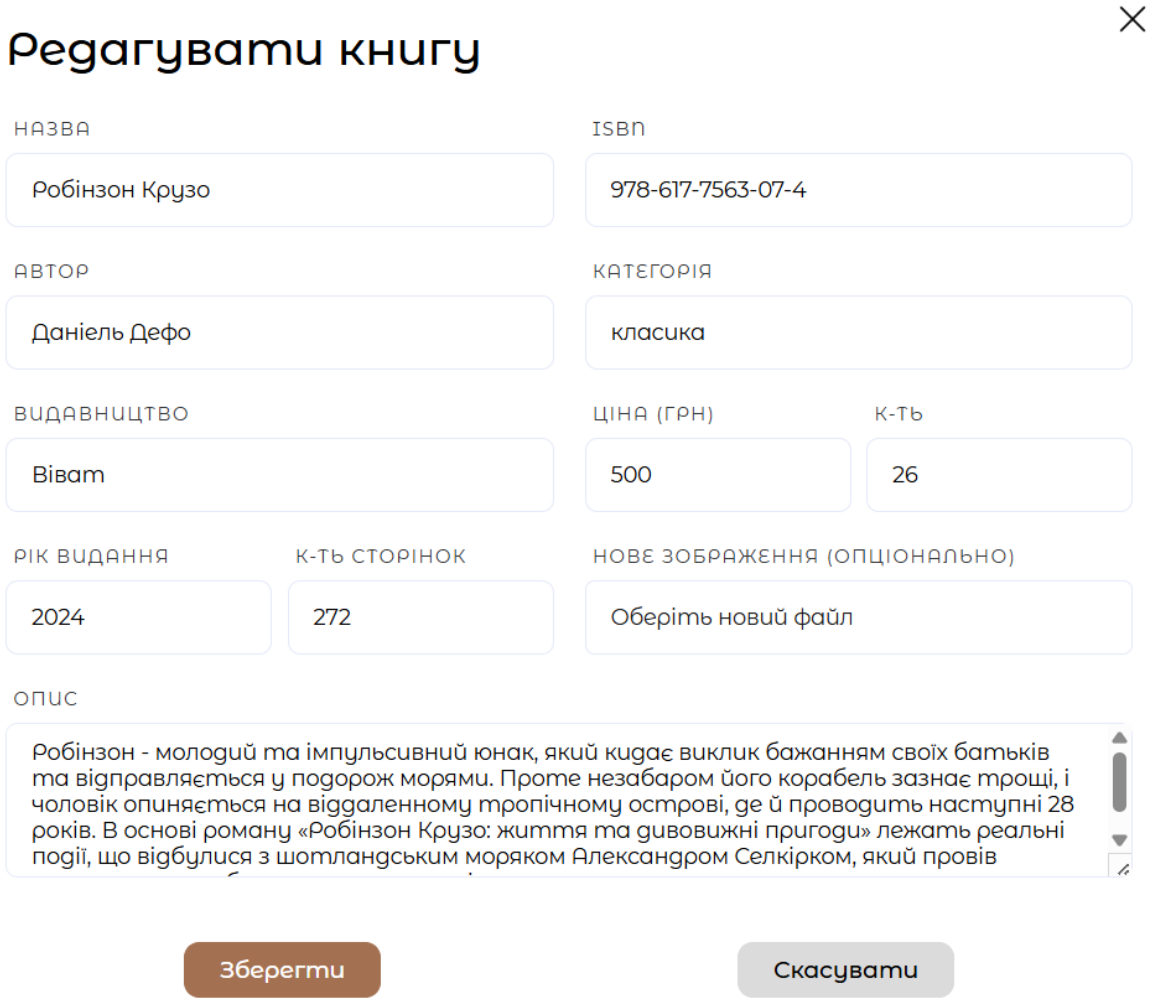


Рисунок 2.19 – Модальне вікно редагування книги

2. Архівування товару.

Задля збереження цілісності бази даних та структури зв’язків між сутностями, замість фізичного видалення книги реалізовано механізм «м'якого видалення» – архівування. Коли адміністратор натискає на кнопку «Архівувати» клієнтська частина надсилає на сервер запит на зміну статусу даної книги. Після успішної відповіді сервера компонент автоматично оновлюється, заархівована книга зникає із загального списку для користувачів, але все ще залишається

доступною в адміністративній панелі.

Такий підхід також допомагає відслідковувати історію покупок клієнтів, що забезпечує коректну роботу системи та запобігає втраті важливих даних.

Щоб забезпечити повне керуванням товарами вебсайт повинен надати адміністратору ключову можливість – створювати товар. Оскільки сутність «Книга» має зовнішні зв’язки з іншими таблицями бази даних, то процес її створення вимагає наявності вже існуючих записів про авторів, категорії та видавництва. Для цього було створено окремі модальні вікна: CreateBookModal, CreateAuthorModal, CreateCategoryModal та CreatePublisherModal (src/components/CreateModals).

Компоненти для створення авторів, категорій і видавництв містять прості форми для введення текстових даних (див. рис. 2.20).

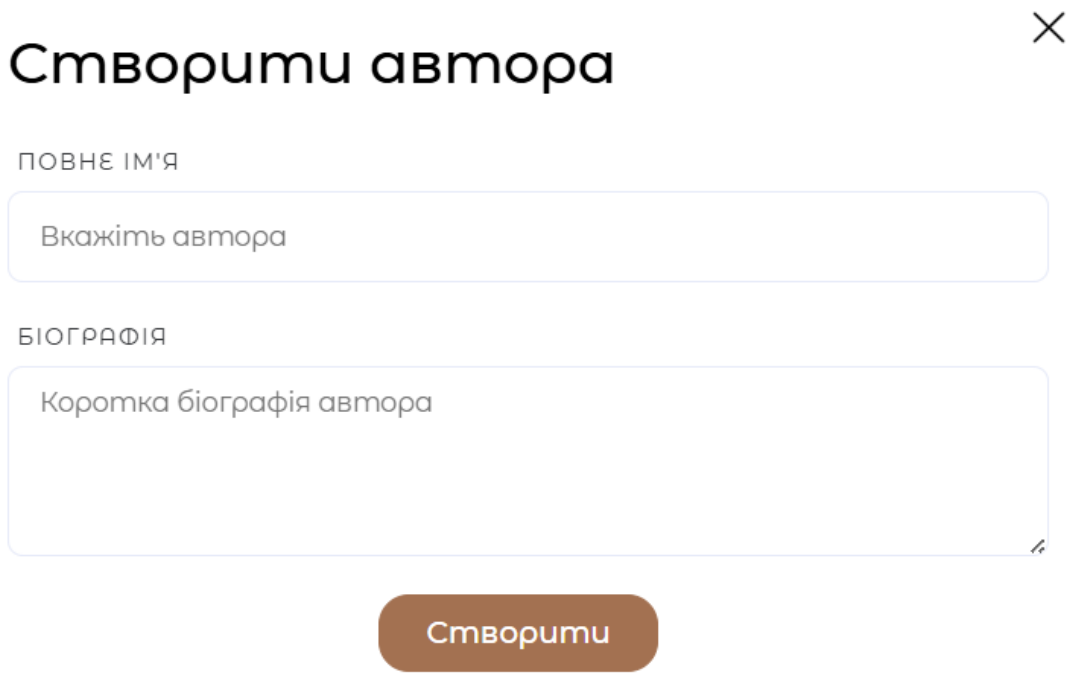


Рисунок 2.20 – Модальне вікно створення автора

Після заповнення форми клієнтська частина вебсайту відправляє запит до серверної частини, а після успішного збереження нові дані одразу оновлюються в інтерфейсі без потреби у перезавантаженні сторінки.

У свою чергу, основна форма створення книги є складнішою. При її відкритті автоматично завантажуються списки авторів, категорій та видавництв з бази даних. Для зручності адміністратора реалізовано також пошук у випадаючих списках, що дозволяє швидко знаходити потрібний елемент навіть при великій кількості записів у базі даних (див. рис. 2.21).

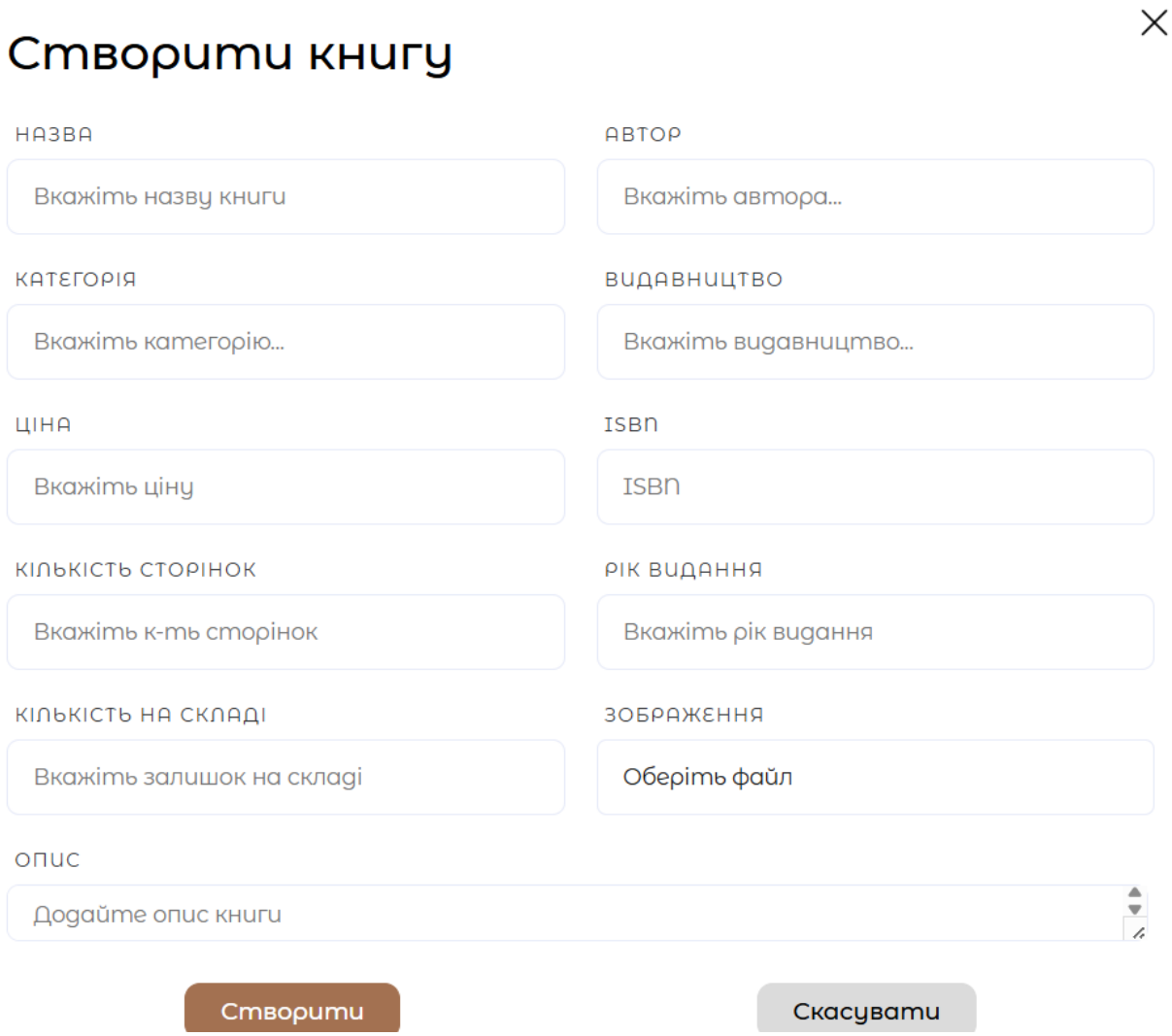


Рисунок 2.21 – Модальне вікно створення книги

Перед відправленням даних застосунок перевіряє правильність заповнення полів, тобто наявність автора, категорії, видавництва, а також коректність числових значень, таких як ціна, рік видання чи залишкова кількість книг на складі.

Логічним завершенням взаємодії користувача з клієнтською частиною

вебсайту «BookFlow» є процес оформлення замовлення, який реалізований у компоненті <CheckoutPage/> (src/pages/CheckoutPage.jsx). Цей етап відповідає за збір даних отримувача замовлення, інформування правил оплати та вибір місця доставки.

Ключовою особливістю даного компонента є інтеграція зі стороннім сервісом логістики – API «Нової Пошти». Для забезпечення оптимальної продуктивності та зменшити кількість зайвих запитів до сервера інтерфейс вибору доставки побудовано за ланцюговим принципом, тобто перелік доступних відділень або поштоматів завантажується лише після того як користувач обере місто і пройде перевірка унікального ідентифікатора обраного міста.

Також для зручності користувача поле введення номера телефону має автоматичну маску відповідного формату, що допомагає вводити дані у правильному форматі та запобігає помилкам (див. рис. 2.22).

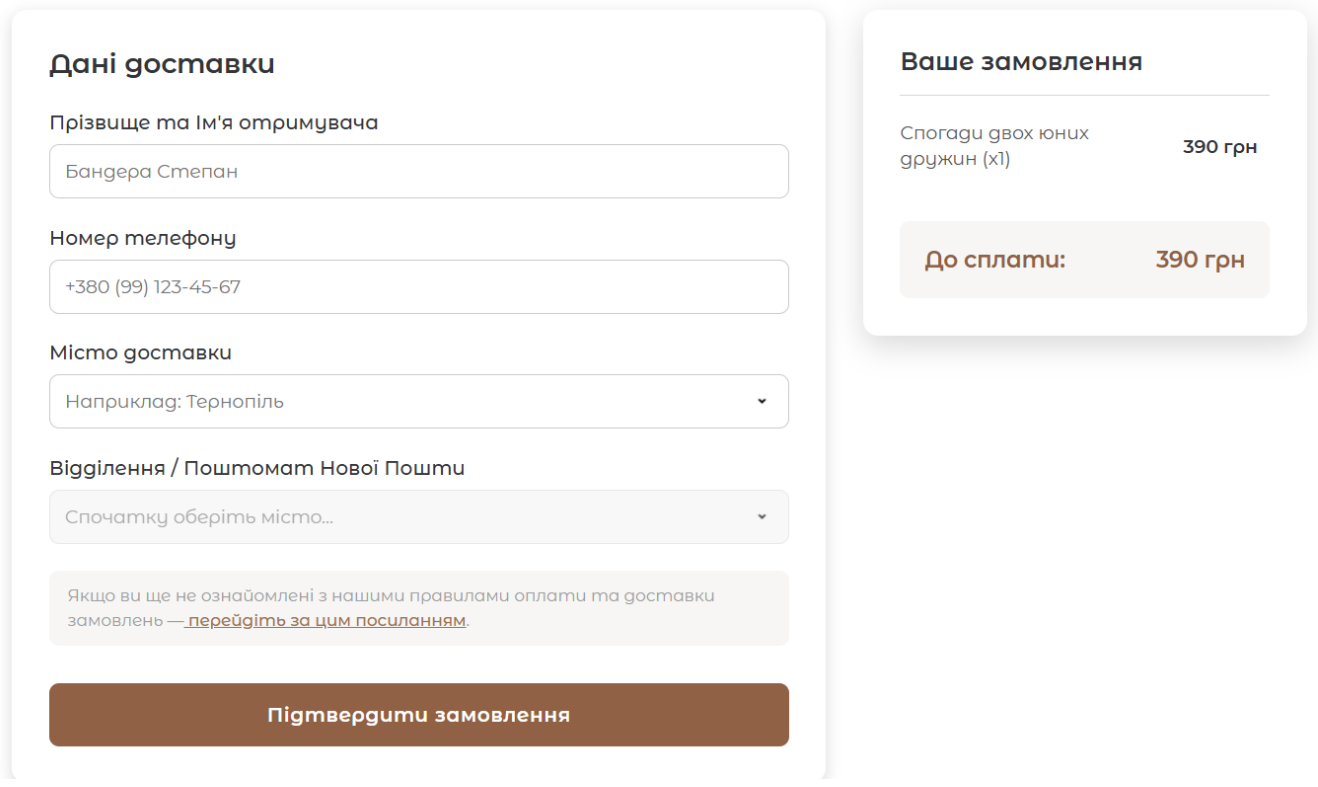


Рисунок 2.22 – Сторінка оформлення замовлення

2.4.2 Написання серверної частини

Базою серверної частини вебсайту магазину книг «BookFlow» обрано мову програмування Java та фреймворк Spring Boot[17].

Вибір Java, як мови програмування, пов'язаний з декількома причинами. Спершу через надійність та строгість типізації, адже саме завдяки ООП (Об'єктно-орієнтована парадигма) та строгій перевірці типів на етапі компіляції, що тим самим зменшує кількість критичних помилок під час виконання програми. Java має вбудовані механізми безпеки, а це є критично важливим фактором для застосунку у сфері електронної комерції.

Причинами вибору фреймворку Spring Boot є його:

– автоконфігурація. Spring Boot автоматично налаштовує компоненти на основі підключених бібліотек;

– вбудований вебсервер. Фреймворк з початкових налаштувань інтегрується із сервером Apache Tomcat. Завдяки цьому не потрібно встановлювати та налаштовувати зовнішній сервер;

– сумісність. Компанія Spring створила потужну екосистему розробки, тому всі додаткові модулі (Spring Data JPA для роботи з базою даних, Spring Security для захисту, Spring Web для створення REST API) ідеально взаємодіють між собою без складних налаштувань. Такий підхід робить Spring Boot ідеальним комплексним рішенням для розробки проєкту.

Також для управління залежностями та автоматизації збірки проєкту використовується система Maven.

Ініціалізація базової структури проєкту здійснюється за допомогою інтегрованого середовища розробки IntelliJ IDE, виконуючи такі кроки:

1. Обираємо Spring Boot у лівій панелі, що під капотом активує офіційний генератор Spring Initializr;

2. Вибір мови програмування Java та системи збірки Maven;

3. Встановлення метаданих проєкту (назва проєкту та пакету);

4. Додавання базових залежностей (Spring Web, Spring Data JPA,

PostgreSQL Driver).

Цей підхід дозволив автоматично згенерувати базву структуру проєкту, файл pom.xml та головний клас запуску застосунку.

Для реалізації повноцінного функціоналу вебсайту «BookFlow» до проєкту було інтегровано декілька додаткових бібліотек:

– Spring Boot Web. Основне призначення – створення REST API, обробка HTTP-запитів та маршрутизація;

– Spring Data JPA & PostgreSQL. Завдання – організація взаємодії з базою даних PostgreSQL за допомогою технології ORM (Hibernate);

– Spring Boot Security. Призначення – забезпечення безпеки застосунку, налаштування правил доступу до кінцевих точок API;

– JJWT (Java JWT) – генерація та валідація JSON Web Tokens[18] для реалізації коректної stateless-авторизації користувачів;

– Google API Client – реалізація можливості авторизації користувачів через сервіси Google OAuth;

– Spring Data Redis – підключення сховища Redis для оптимізації роботи системи;

– Cloudinary – інтеграція з хмарним сервісом для завантаження, зберігання та оптимізації медіафайлів, а саме обкладинок книг;

– Spring Boot Mail – налаштування SMTP-клієнта для відправки системних електронних листів користувачам;

– Lombok – генерація шаблонного коду на етапі компіляції для зменшення обсягу однотипних частин;

– Spring Boot Validation – забезпечення перевірки та валідації вхідних даних на стороні сервера, перед їх обробкою.

Архітектуру кореневого пакету проєкту було розділено на логічні папки, щоб забезпечити масштабованість та зручність підтримки (див. рис. 2.23).

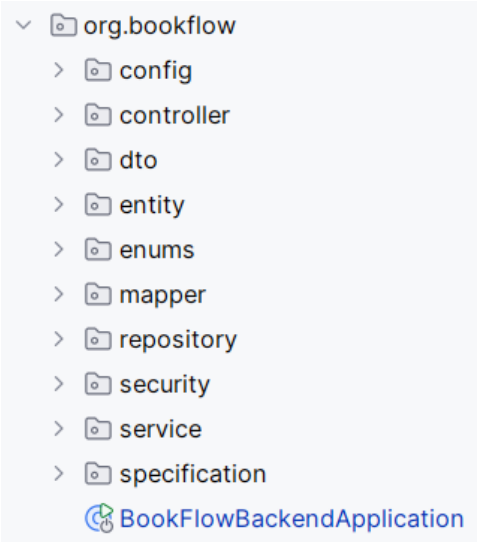


Рисунок 2.23 – Структура кореневого пакету

Кожна папка (див. рис. 2.23) виконує певну роль у системі:

– «config» – це основа налаштувань всіх додаткових конфігурацій. Даний каталог містить класи для роботи CORS, Swagger для документації, Cloudinary[19] для зберігання зображень та Redis[20] для сховища даних. Також тут за допомогою Spring Security виконується налаштування фільтрів безпеки та доступу до ендпоінтів вебсайту;

– «controller» – це каталог, який відповідає за приймання та обробку вхідних HTTP-запитів з клієнтської частини;

– «dto» – це об'єкти передачі даних. Даний міжшаровий рівень було створено для підвищення безпеки та зручності валідації вхідних даних. Тобто об’єкти цього рівня використовуються для зберігання та передачі даних між шаром контролерів і сервісів;

– «entity» – це ключовий каталог сутностей вебсайту «BookFlow», він містить класи-моделі, які відображають структуру таблиць у базі даних PostgreSQL;

– «enums» – це папка, де зберігаються набори констант, які використовуються в системі (статуси замовлень, роль користувача тощо);

– «mapper» – це аналогічно з папкою «dto» є міжшаровим каталогом, де міститься логіка для перетворення об'єктів бази даних (entity) в об'єкти передачі

даних (dto) і навпаки;

– «repository» – це шар доступу до даних, тобто саме тут відбувається взаємодія з БД. Дана папка містить інтерфейси, які забезпечують виконання базових CRUD-операцій над сутностями бази даних без необхідності написання SQL-запиті самостійно;

– «security» – це пакет, котрий містить логіку захисту серверної частини. В ньому розміщені класи для налаштування обробки JWT-токенів;

– «service» – це основний шар, який відповідає за бізнес-логіку вебсайту. Саме тут виконуються перевірки даних, необхідні обчислення та взаємодія між різними компонентами системи;

– «specification» – це каталог, який містить класи, які за допомогою Spring Data JPA Specifications створюють динамічні запити до бази даних. Це важливий компонент для реалізації складних систем фільтрації за кількома параметрами одночасно.

Окрім описаних вище пакетів, дуже важливу роль у функціонуванні та розгортанні серверної частини відіграють конфігураційні файли. Основними з них є pom.xml та application.properties.

– «pom.xml» – це головний файл конфігурації Maven, який розташований у кореневій папці проєкту. Він використовується для:

а) підключення необхідних бібліотек і залежностей;

б) налаштування збірки проєкту;

в) зберігання інформації про метадані проєкту.

– «application.properties» – це головний файл налаштувань самого фреймворку Spring Boot, який стандартно розміщується у директорії src/main/resources. В основному application.properties відповідає за етап виконання застосунку. Його призначення:

а) налаштування середовища;

б) параметри підключення до баз даних;

в) інтеграція змінних середовища;

г) кастомні властивості застосунку.

2.5 Тестування вебсайту

Заключною складовою розробки вебсайту магазину книг «BookFlow» є проведення його тестування. Цей процес є необхідним для розуміння рівня відповідності реалізованого продукту поставленим технічним вимогам, пошуку помилок, оцінки стабільності та безпеки вебзастосунку.

Для перевірки зазначених аспектів було використано декілька видів тестування, а саме:

- unit testing (тестування окремих модулів);
- integration testing (перевірку взаємодії між компонентами);
- ручне тестування інтерфейсу та функціоналу вебсайту.

З огляду на те, що клієнтська частина вебзастосунку реалізована на базі бібліотеки React та інструменту збірки Vite, є такі інструменти проведення тестувань:

1. Vitest. Це спеціалізоване середовище для проведення та аналізу тестів, яке добре інтегрується з Vite;

2. React Testing Library. Це бібліотека спеціально призначена для тестування React-компонентів. Її особливість у перевірці інтерфейсу так, як це робить звичайний користувач, тобто натискання кнопок, введення тексту, взаємодія з елементами сторінки тощо;

3. MSW. Це бібліотечний інструмент, який перехоплює HTTP-запити. Тобто ця бібліотека дає можливість створювати фіктивні відповіді від сервера, таким чином це корисний спосіб для проведення ізольованого тестування клієнтської частини вебсайту, не беручи до уваги поточний стан серверної частини і бази даних.

Серверна частина у свою чергу розроблена за допомогою фреймворку Spring Boot і мови програмування Java. Він надає хорошу реалізацію архітектурної екосистеми для тестування, інструменти якої входять до складу

Spring Boot Test. Взявши до уваги ці аспекти, було виділено такі інструменти тестування:

1. JUnit 5. Це базовий фреймворк для написання тестів на Java. Він надає систему анотації та використовується для перевірки правильності роботи методів контролерів і сервісів;

2. Mockito. Це бібліотека для створення тестових об’єктів, по-іншому ще об’єктів-заглушок. У трирівневій архітектурі серверної частини вона дозволяє ізолювати логіку сервісів без реального підключення до бази даних. Наприклад, під час тестування шару бізнес-логіки можна імітувати роботу репозиторію та повертати наперед запрограмовані дані;

3. MockMvc. Це спеціалізований інструмент Spring Test для тестування шару контролерів без запуску реального сервера. Завдяки MockMvc здійснюється симуляція надсилання HTTP-запитів на ендпоінти вебсайту, перевірка статусів відповідей та структури JSON-документів, що повертають користувачу;

4. @SpringBootTest. Це анотація для комплексного тестування застосунку. Вона запускає всі основні компоненти системи та дозволяє перевірити взаємодію всіх компонентів системи разом.

Отже, спершу було проведено базове модульне тестування серверної частини на базі фреймворку JUnit 5, який поданий у Додатку З кваліфікаційної роботи.

Мета тесту заключає перевірку базової сутності Book, а тестовий сценарій перевіряє правильність ініціалізації об'єкта, запису даних та їх зчитування через геттери та сеттери (див. рис 2.24).

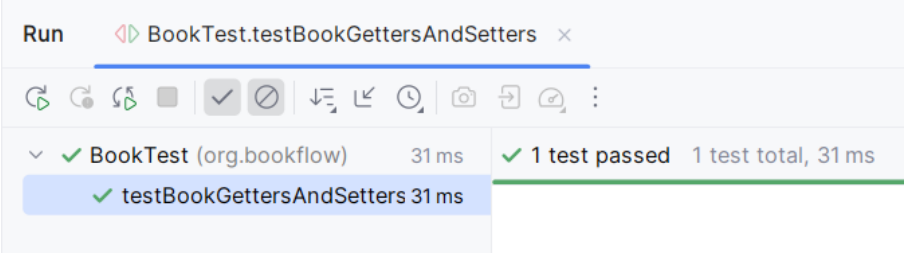


Рисунок 2.24 – Результат успішного проходження автоматизованого Unit-тесту у середовищі розробки

Далі було проведено інтеграційне тестування серверної частини API-ендпоінтів за допомогою інструмента MockMvc інтегрованого у Spring Boot Test. Повний код тесту наведено у Додатку И кваліфікаційної роботи.

Основна мета тесту перевірити, чи правильно працюють REST-контролери та чи коректно обробляються HTTP-запити без запуску реального сервера Tomcat. Тест імітує відправку GET-запиту на ендпоінт «/api/books», який повертає список книг. За допомогою перевірки status().isOk() було підтверджено, що сервер успішно обробляє запит і повертає статус 200 OK (див. рис. 2.25).

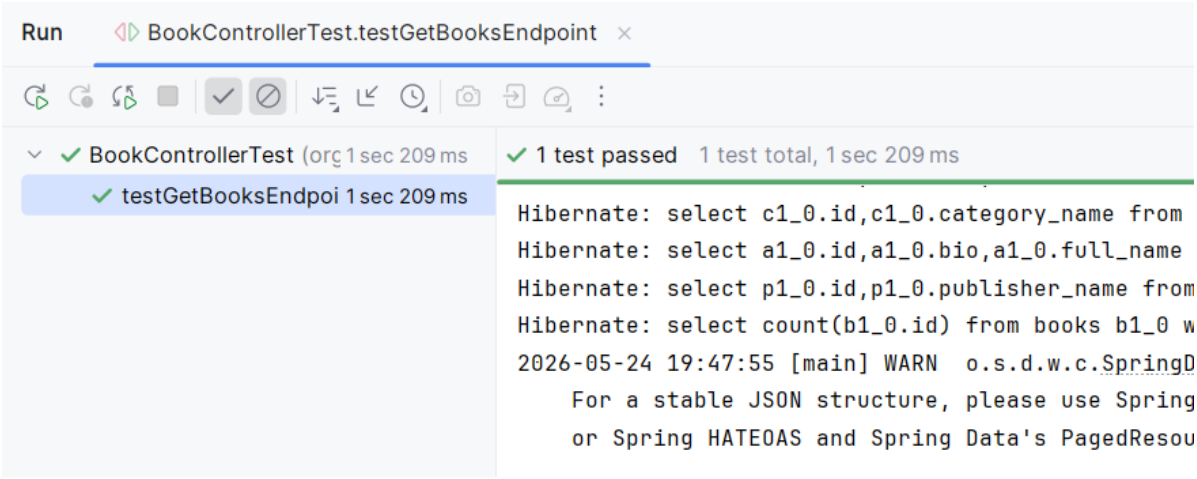


Рисунок 2.25 – Результат виконання інтеграційного тесту REST-контролера за допомогою MockMvc

Останнім кроком було проведене ручне функціональне тестування клієнтської частини, а саме адаптивності вебсайту магазину книг «BookFlow». Перевірка графічного інтерфейсу виконувалася за допомогою вбудованих інструментів розробника Google Chrome DevTools шляхом імітації екранів мобільних пристроїв. Тестування було поділено на два ключові етапи: перевірку користувацького інтерфейсу на сторінках каталогу та перевірку панелі управління.

Зі сторони звичайного користувача було перевірено коректність відображення головних сторінок та каталогу товарів (див. рис. 2.26).

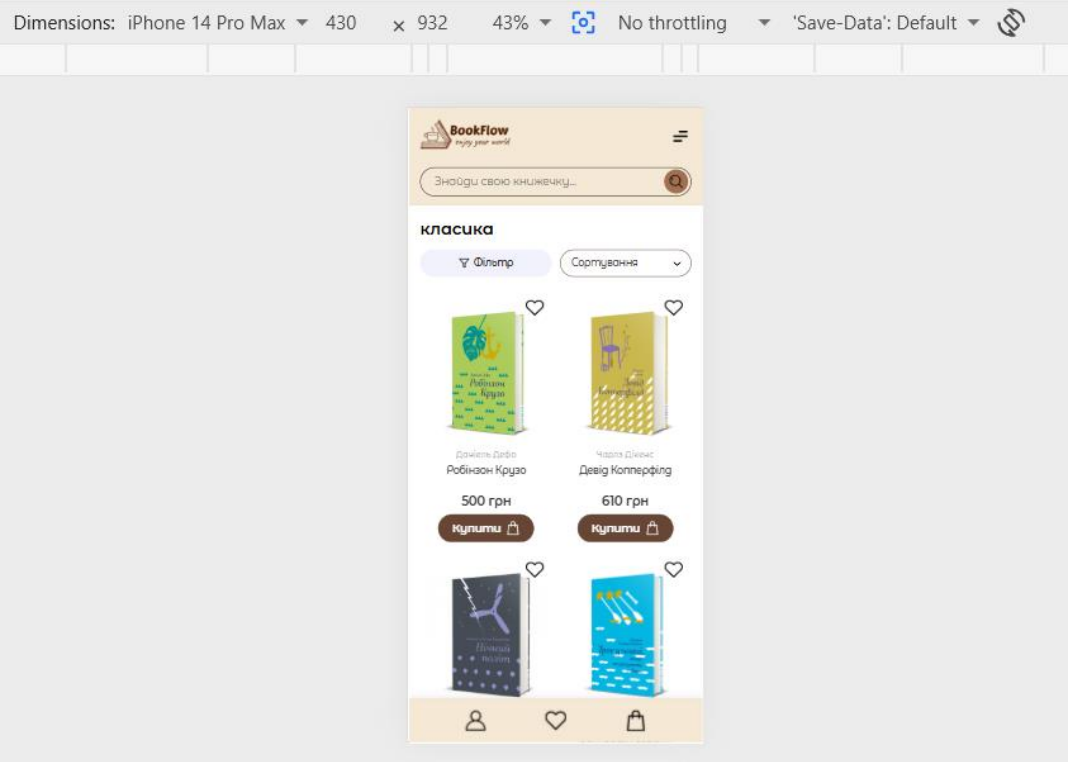


Рисунок 2.26 – Результат тестування адаптивності для звичайного користувача

Другим етапом стала перевірка закритої частини вебсайту, призначеної для адміністраторів, де структура сторінок є більш складною (див. рис. 2.27).

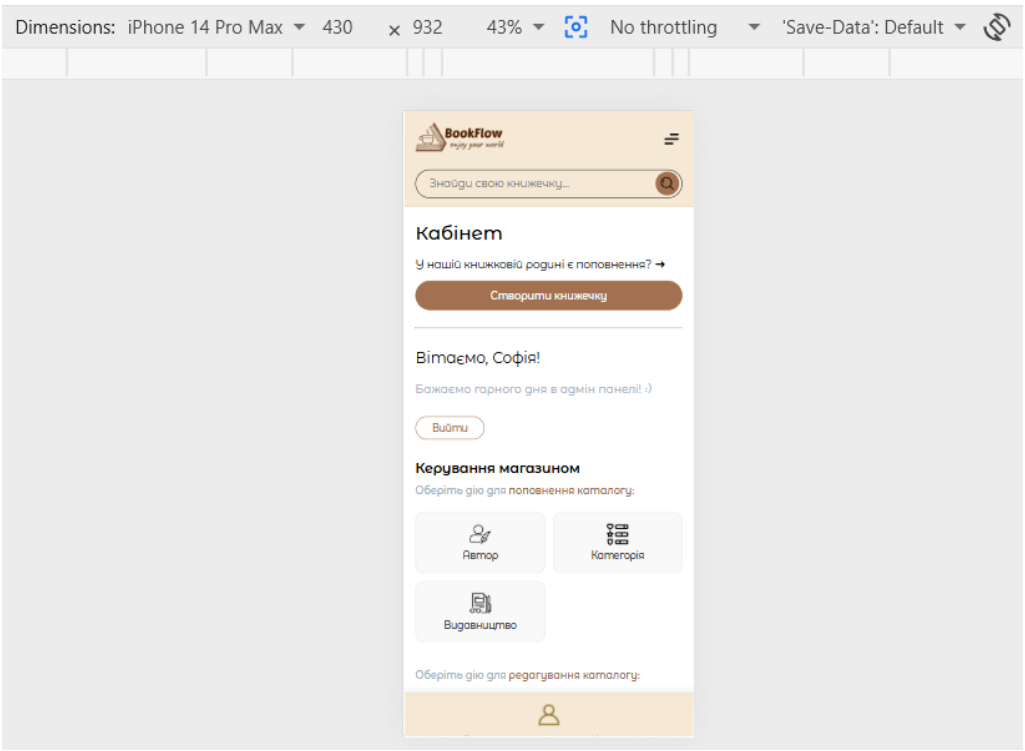


Рисунок 2.27 – Результат тестування адаптивності для адміністратора

3 СПЕЦІАЛЬНИЙ РОЗДІЛ

3.1 Інструкція з розміщення сайту в Інтернеті

Розміщення вебсайту магазину книг «BookFlow» в мережі Інтернет здійснюється з використанням сучасних хмарних платформ.

З огляду на те, що вебсайт поділяється на серверну частину та клієнтську – це буквально означає два незалежні проєкти. Саме тому для зручності розгортання та хорошої продуктивності було обрано сервіси Railway[21] та Vercel[22].

Хостингова патформа Railway передбачатиме розміщення серверної частини (Spring Boot), бази даних (PostgreSQL) та кешування (Redis), а Vercel, у свою чергу, відповідатиме за клієнтську частину, реалізовану на базі React та Vite. Вибір у використанні цих платформ також обгрунтовано тим, що вони дозволяють автоматизувати процес збірки проєкту безпосередньо з репозиторію GitHub без необхідності ручного налаштування серверів.

Процес розміщення вебсайту поділяється на кілька послідовних етапів:

1. Розгортання інфраструктури баз даних на Railway.

Даний етап передбачає створення нового проєкту на платформі Railway, в який додаються сервіси PostgreSQL та Redis (див. рис. 3.1). Після їх налаштування отримуються параметри підключення до бази даних: хост, порт, назва бази даних, логін і пароль. Оскільки віддалена база даних спочатку порожня, виконується перенесення даних із локального середовища. Для цього за допомогою pgAdmin[23] створюється резервна копія локальної бази даних, після чого здійснюється підключення до бази Railway через External Connection (TCP Proxy) та виконується відновлення даних. При цьому під час відновлення вимикається перенесення власника (Owner) і привілеїв (Privileges), адже це дозволяє уникнути можливих конфліктів прав доступу на сервері.

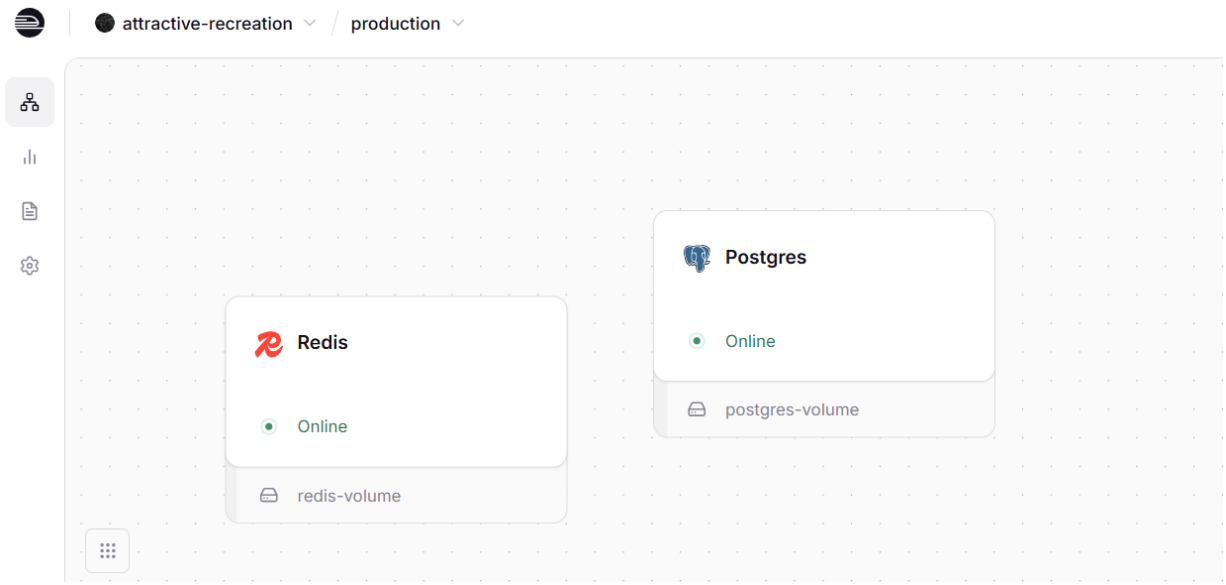


Рисунок 3.1 – Розгортання інфраструктури баз даних на Railway

2. Первинне розгортання клієнтської частини на Vercel.

Цей етап виконує розгортання клієнтської частини застосунку на платформі Vercel. Для цього створюється новий проєкт шляхом імпорту репозиторію фронтенду з GitHub (див. рис. 3.2).

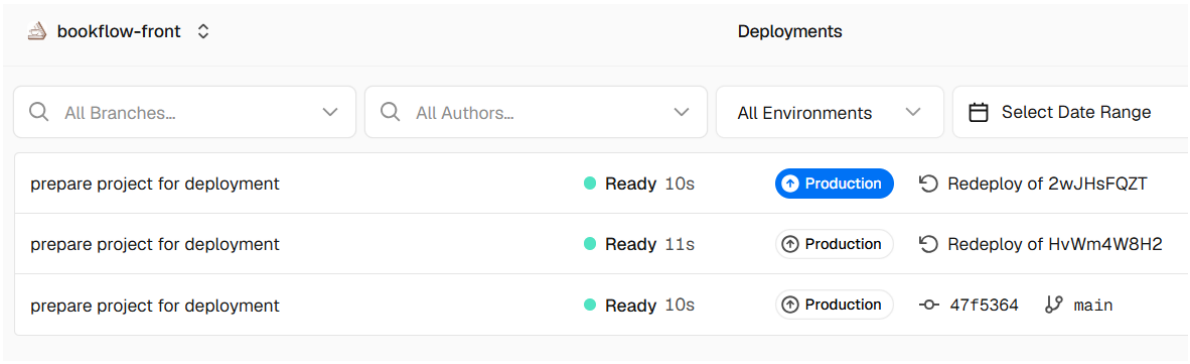


Рисунок 3.2 – Розгортання клієнтської частини на Vercel

У налаштуваннях середовища додаються необхідні змінні, зокрема ключі для інтеграції з API «Нової Пошти» та Google OAuth. Також на цьому етапі змінна базового URL серверного API залишається незаповненою, оскільки серверна частина ще не розгорнута. Після завершення автоматичного процесу збірки та деплою Vercel надає публічне посилання, за яким стає доступною клієнтська частина вебзастосунку.

3. Розгортання серверної частини на Railway.

На цьому етапі ключовим є розгортання серверної частини застосунку на платформі Railway. Для цього, до вже створеного проєкту (див. рис. 3.1) додається новий вебсервіс, який підключається до репозиторію з бекендом. Після цього налаштовуються змінні середовища на основі параметрів із локального конфігураційного файлу. Окрему увагу приділяється налаштуванню підключення до PostgreSQL, де URL бази даних адаптується до формату JDBC, необхідного для роботи Spring Boot. Також додається змінна з адресою клієнтської частини, розгорнутої на Vercel, що забезпечує коректну взаємодію між фронтендом і бекендом. Після завершення процесу збірки Railway автоматично розгортає застосунок і надає публічний домен, за яким стає доступним серверний API.

4. Фінальне налаштування та зв'язування компонентів.

Завершальним етапом є налаштування взаємодії між клієнтською та серверною частинами застосунку. Для цього в проєкті на Vercel (див. рис. 3.2) додається змінна середовища з адресою API серверної частини, розгорнутої на Railway. Після оновлення налаштувань виконується повторне розгортання клієнтського застосунку, щоб нові параметри були враховані під час збірки. Також оновлюються налаштування Google OAuth, де до списку дозволених джерел додається продакшен-домен клієнтської частини. Це забезпечує коректну роботу авторизації через Google та повноцінну взаємодію всіх компонентів системи в робочому середовищі.

Таким чином, у результаті виконання описаних етапів розгортання, вебзастосунок було успішно опубліковано в мережі Інтернет та забезпечено його повноцінне функціонування. Доступ до вебсайту здійснюється за посиланням: https://bookflow-ua.vercel.app

3.2 Інструкція з обслуговування та наповнення сайту

Обслуговування та наповнення вебсайту магазину книг «BookFlow» здійснюється виключно користувачами з роллю адміністратора.

З метою забезпечення безпеки вебзастосунку під час розробки було реалізовано механізм, за яким усі новостворені облікові записи автоматично отримують роль звичайного користувача («USER»). Саме тому, при необхідності надання користувачу адміністративних прав зміна ролі виконується безпосередньо в базі даних.

Оскільки в проєкті використовується реляційна система керування базами даних PostgreSQL, для взаємодії з нею застосовується середовище pgAdmin.

Для зміни ролі користувача необхідно виконати такі дії:

1. Відкрити базу даних проєкту;

2. Перейти до таблиці «users»;

3. Знайти відповідний запис користувача та змінити значення поля «role» з «USER» на «ADMIN» (див. рис. 3.3).

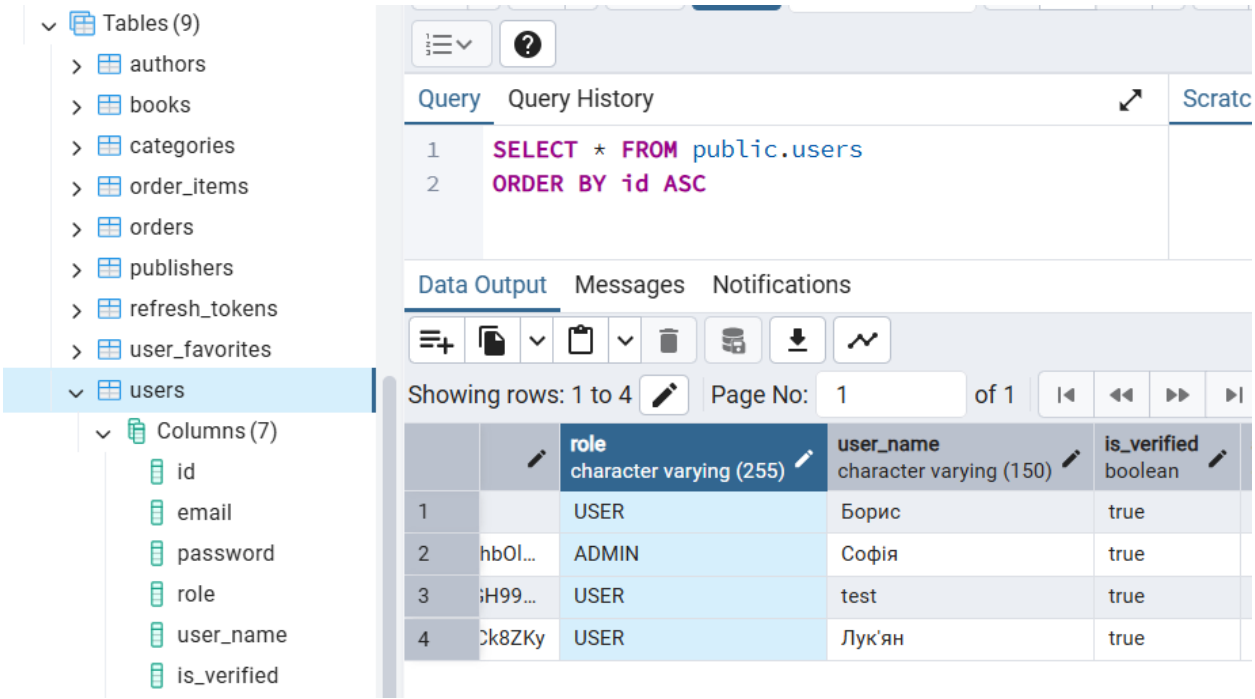


Рисунок 3.3 – Зміна ролі користувача

Після зміни ролі користувача та проходженні авторизації вебсайт автоматично надає доступ до розширеного функціоналу адміністратора.

Наповнення та обслуговування вебсайту здійснюється через особистий кабінет адміністратора, у межах якого реалізовано можливість створення,

редагування та керування товарами. Створення нових товарів виконується за допомогою спеціалізованих компонентів <CreateModals/> (див. рис. 2.20). Редагування вже існуючих записів реалізовано через компоненти <EditModals/> (див. рис. 2.19).

У разі необхідності зняття товару з продажу адміністратор повинен перейти на сторінку відповідної книги та скористатися спеціальною кнопкою керування статусом товару (див. рис. 2.14).

Для зручності адміністрування в особистому кабінеті передбачено окремий розділ, у якому зберігається перелік усіх книг, що були зняті з продажу. Це дозволяє здійснювати подальше керування такими товарами та за потреби повторно повертати їх до каталогу.

Окрім керування товарами, через особистий кабінет адміністратора реалізовано функціонал обробки замовлень. У відповідному розділі адміністратор має можливість переглядати всі оформлені замовлення, контролювати їхній статус та виконувати подальші дії, пов’язані з обслуговуванням покупців (див. рис. 3.4).

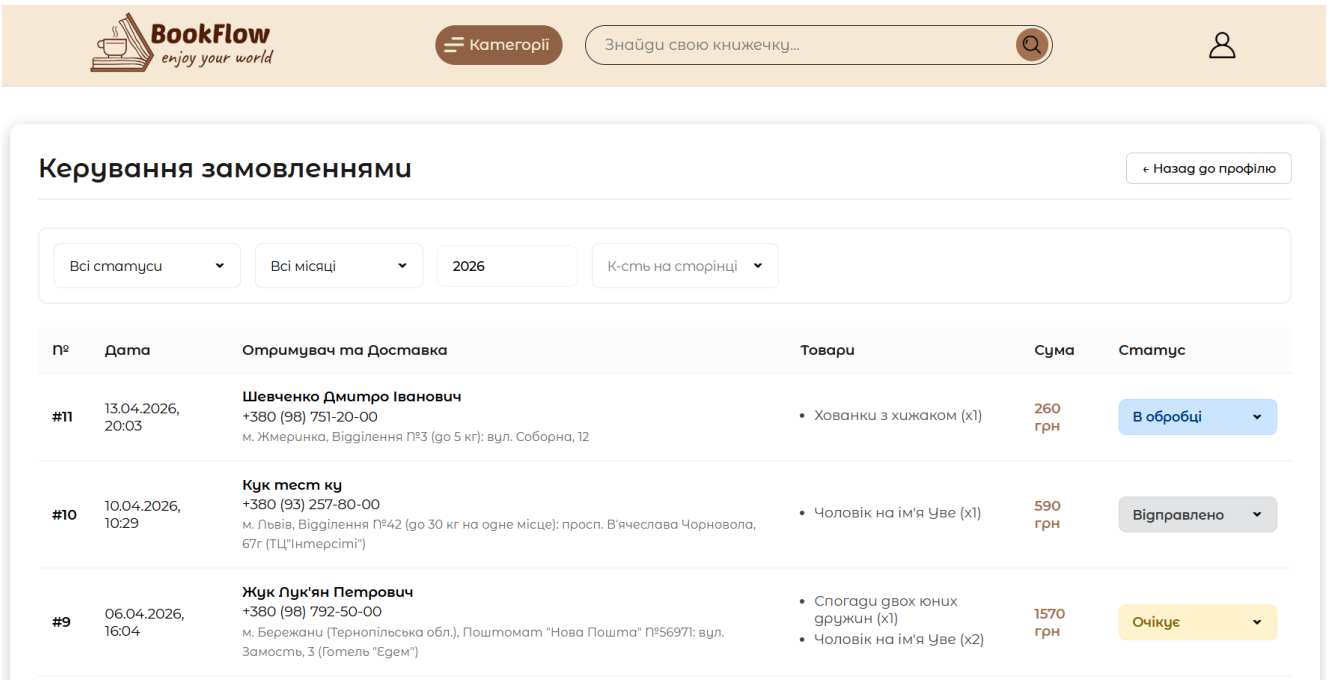


Рисунок 3.4 – Інтерфейс адміністратора для обробки замовлень

3.3 Інструкція з популяризації та підтримки сайту

Для забезпечення стабільного розвитку вебсайту, збільшення кількості відвідувачів та підвищення впізнаваності магазину книг «BookFlow» необхідно постійно застосовувати комплекс маркетингових та організаційних заходів.

Насамперед важливим є аналіз ринку та цільової аудиторії. Це означає що потрібно регулярно досліджувати потреби користувачів, визначати переваги вебресурсу серед конкурентів та формувати стратегію його подальшого просування.

Також одним із ключових аспектів є покращення користувацького інтерфейсу. Для цього необхідно забезпечити просту та зрозумілу навігацію, адаптивність для різних пристроїв, стабільну роботу форм, а також використовувати плавні анімації для інтерактивних елементів, щоб у клієнта створювалось приємне візуальне враження.

Для підвищення позицій сайту в пошукових системах у свою чергу необхідно здійснювати SEO-оптимізацію. Вона включає використання ключових слів, оптимізацію контенту, заповнення мета-тегів та створення унікальних текстових матеріалів. Але оскільки клієнтська частина вебзастосунку реалізована за допомогою React, важливо враховувати специфіку SEO-оптимізації односторінкового застосунку. В таких випадках необхідно використовувати зрозумілі URL-адреси, забезпечувати швидке завантаження сторінок, оптимізувати зображення прописавши у відповідний атрибут ключові слова, а також створити файл sitemap.xml для коректної індексації сайту пошуковими системами [24].

Можливості Vite також дають результати у покращенні продуктивності вебсайту, зокрема через оптимізацію файлів проєкту, стиснення зображень та кешування даних. А ще важливо забезпечити швидку роботу серверної частини, оскільки швидкість відповіді API безпосередньо впливає на швидкість завантаження сторінок.

Важливим інструментом популяризації є просування у соціальних

мережах. Для залучення нової аудиторії доцільно вести сторінки вебресурсу в соціальних мережах, публікувати новини, огляди книг, інформацію про акції та проводити кооперації з іншими брендами або зірками шоу-бізнесу.

Окрему увагу необхідно приділяти підтримці комунікації з користувачами. Для цього потрібно своєчасно реагувати на звернення через форми зворотного зв’язку або електронну пошту.

Також доцільно використовувати системи вебаналітики, зокрема Google Analytics та Google Search Console. Їх використання дозволяє аналізувати поведінку користувачів, відстежувати джерела трафіку, показник відмов та глибину перегляду.

Підсумовуючи можна сказати що важливою складовою підтримки вебсайту є його постійний розвиток та адаптація до потреб користувачів. Це включає вдосконалення інтерфейсу, додавання нового функціоналу та коригування маркетингової стратегії на основі отриманих аналітичних даних і відгуків користувачів.

4 ЕКОНОМІЧНИЙ РОЗДІЛ

Метою економічної частини даної кваліфікаційної роботи є проведення економічних розрахунків, спрямованих на визначення економічної ефективності розробки вебсайту магазину книг «BookFlow», прийняття рішення про подальший розвиток і впровадження або ж недоцільність проведення відповідної розробки.

Об’єктом розробки є вебзастосунок для магазину книг «BookFlow».

Розрахунок вартості розробки виконується в декілька етапів:

– описати технологічний процес розробки із зазначенням трудомісткості кожної операції;

– визначити суму витрат на оплату праці основного і допоміжного персоналу, включаючи відрахування на соціальні заходи;

– обчислити витрати на електроенергію;

– нарахувати суму амортизаційних відрахувань;

– визначити суму накладних витрат;

– скласти кошторис та визначити собівартість робіт;

– розрахувати ціну робіт;

– визначити економічну ефективність та термін окупності.

4.1 Визначення стадій технологічного процесу та загальної тривалості проведення НДР

В цьому підрозділі розглянемо основні етапи технологічного процесу для розробки вебсайту магазину книг «BookFlow».

Для визначення загальної тривалості проведення науково-дослідних робіт доцільно дані витрат часу по окремих операціях технологічного процесу звести у таблицю 4.1.

Таблиця 4.1 – Середній час виконання робіт по обслуговуванню та стадії

(операції) технологічного процесу

№ п/п	Назва операції (стадії)	Виконавець	Середній час виконання операції, год.
1	Планування та аналіз	Керівник проєкту	8
		Веброзробник	6
2	Розробка технічного завдання	Веброзробник	7
3	Проєктування дизайну інтерфейсу	Веброзробник	21
4	Розробка функціоналу вебсайту	Веброзробник	76
5	Тестування та налагодження	Тестувальник	18
6	Документування	Інженер	5
7	Розгортання та підтримка	Веброзробник	5
Разом			146

Сумарний час виконання операцій технологічного процесу становить 146 годин.

4.2 Визначення витрат на оплату праці та відрахувань на соціальні заходи

У даному підрозділі проводиться аналіз і розрахунок витрат, пов'язаних з оплатою праці та відрахуваннями на соціальні заходи, що необхідні для розробки вебсайту магазину книг «BookFlow».

Розмір заробітної плати залежить від складності та умов виконуваної роботи, професійно-ділових якостей працівника, результатів його праці та діяльності підприємства.

Основна заробітна плата розраховується за формулою:

$$З_{осн.} = Т_с \cdot К_г \quad (4.1)$$

де: T_c – тарифна ставка, грн. (приймаємо для керівника проєкту – 450 грн./год, веброзробника – 272 грн./год., інженера – 113 грн./год., тестувальника – 100 грн./год.); $К_г$ – кількість відпрацьованих годин.

Отже, основна заробітна плата для:

Керівника проекту $З_{осн1} = 450 * 8 = 3\ 600$ грн.

Веброзробника $З_{осн2} = 272 * 115 = 31\ 280$ грн.

Інженера $З_{осн3} = 113 * 5 = 565$ грн.

Тестувальника $З_{осн4} = 100 * 18 = 1\ 800$ грн.

Сумарна основна заробітна плата становить:

$$З_{осн} = 3\ 600 + 31\ 280 + 565 + 1\ 800 = 37\ 245 \text{ грн.}$$

Додаткова заробітна плата становить 10 – 15 % від суми основної заробітної плати.

$$З_{дод.} = З_{осн.} \cdot К_{допл.} \quad (4.2)$$

де: $К_{допл.}$ – коефіцієнт додаткових виплат працівникам.

Отже додаткова заробітна плата по категоріях працівників становить:

Керівника проекту $З_{дод1} = 3\ 600 * 0,1 = 360$ грн.

Веброзробник $З_{дод2} = 31\ 280 * 0,1 = 3\ 128$ грн.

Інженера $З_{дод3} = 565 * 0,1 = 56,5$ грн.

Тестувальник $З_{дод4} = 1\ 800 * 0,1 = 180$ грн.

Загальна додаткова заробітна плата становить:

$$З_{дод} = 360 + 3\ 128 + 56,5 + 180 = 3\ 724,5 \text{ грн.}$$

Звідси загальні витрати на оплату праці ($В_{о.п.}$) визначаються за формулою:

$$В_{о.п.} = З_{осн.} + З_{дод.} \quad (4.3)$$

$$В_{о.п} = 37\ 245 + 3\ 724,5 = 40\ 969,5 \text{ грн.}$$

Єдиний соціальний внесок (ЄСВ – 22%) визначається за формулою:

$$В_{єсв} = Воп \cdot 0,22 \quad (4.4)$$

$$В_{єсв} = 40\ 969,5 * 0,22 = 9\ 013,29 \text{ грн.}$$

Проведені розрахунки витрат на оплату праці наведено у таблиці 4.2.

Таблиця 4.2 – Зведені розрахунки витрат на оплату праці

№ п/п	Категорія працівників	Основна заробітна плата, грн.			Додаткова заробітна плата, грн.	ЄСВ, грн.	Всього витрати на оплату праці, грн. 6=3+4+5
		Тарифна ставка, грн.	К-сть годин	Фактично нарах. зарплати, грн.			
		1	2	3	4	5	6
1	Кер. проєкту	450	8	3 600	360		
2	Веброзробник	272	115	31 280	3 128		
3	Інженер	113	5	565	56,5		
4	Тестувальник	100	18	1 800	180		
Разом				37245	3724,5	9013,29	49982,79

Отже, загальні витрати на оплату праці становлять 49 982,79 грн.

4.3 Розрахунок витрат на електроенергію

Розрахуємо вартість електроенергії. Затрати на електроенергію 1-ці обладнання визначаються за формулою:

$$З_в = W \cdot T \cdot S \quad (4.5)$$

де: W – необхідна потужність, кВт; T – кількість годин роботи обладнання; S – вартість кіловат-години електроенергії (приймаємо 15, 94 грн).

В нашій системі є 1 ПК. Витрати на електроенергію для цього комп’ютера обчислимо окремо, взявши за основу, що час роботи обладнання обчислюється в залежності від виконуваних робіт (згідно табл. 4.1) і споживані потужності

наступні: комп’ютер – 0,82 кВт/год.

$$З_{ек} = 0,82 * 146 * 15,94 = 1\ 908,34 \text{ грн.}$$

Витрати на електроенергію становлять 1 908,34 грн.

4.4 Розрахунок суми амортизаційних відрахувань вебсайту магазину книг «BookFlow»

Характерною особливістю застосування основних фондів у процесі виробництва є їх відновлення. Для відновлення засобів праці у натуральному виразі необхідне їх відшкодування у вартісній формі, яке здійснюється шляхом амортизації.

Амортизація – це процес перенесення вартості основних фондів на вартість новоствореної продукції з метою їх повного відновлення

Комп'ютери та оргтехніка належать до четвертої групи основних фондів.

Амортизація на них нараховується лише в випадку, якщо мінімально допустимі строки їх корисного використання 2 роки. Для визначення амортизаційних відрахувань застосовуємо формулу:

$$А = \frac{Б_В \cdot Н_А}{100\%} \cdot Т \qquad (4.6)$$

де: А – амортизаційні відрахування за звітний період, грн.; $Б_В$ – балансова вартість групи основних фондів на початок звітного періоду, грн.; $Н_А$ – норма амортизації, 0,04 %.

Оскільки для написання програми та її тестування використовується один ПК, вартістю 35000,00 грн., то сума амортизаційних відрахувань становитиме:

$$А = \frac{35\ \ 000,00 * 0,04}{150} * 146 = 1\ 362,7 \text{ грн.}$$

4.5 Обчислення накладних витрат

Накладні витрати пов’язані з обслуговуванням виробництва, утриманням

апарату управління підприємства (фірми) та створення необхідних умов праці.
В залежності від організаційно-правової форми діяльності господарюючого суб'єкта, накладні витрати можуть становити 20–60 % від суми основної та додаткової заробітної плати працівників.

$$Н_в = В_{о.п.} \cdot 0{,}2 \dots 0{,}6 \quad (4.7)$$

де: $Н_В$ – накладні витрати.

$$Н_в = 40\,969{,}5 * 0{,}4 = 16\,387{,}8 \text{ грн.}$$

4.6 Складання кошторису витрат та визначення собівартості вебсайту магазину книг «BookFlow»

Для складення кошторису витрат та визначення собівартості, результати проведених вище розрахунків зведемо у таблиці 4.4.

Таблиця 4.4 – Кошторис витрат вебсайту магазину книг «BookFlow»

№	Зміст витрат	Сума, грн.	В % до загальної суми
1.	Витрати на оплату праці	49 982,79	71,8
2.	Витрати на електроенергію	1 908,34	2,7
3.	Амортизаційні відрахування	1 362,7	2
4.	Накладні витрати	16 387,8	23,5
5.	Собівартість	69 641,63	100

Собівартість ($С_В$) НДР розрахуємо за формулою:

$$С_в = В_{о.п.} + В_{с.з.} + З_е + А + Н_в \quad (4.8)$$

Отже, собівартість дорівнює $С_в$ = 69 641,63 грн.

4.7 Розрахунок ціни вебсайту магазину книг «BookFlow»

Розрахунок ціни науково-дослідної роботи включає в себе урахування різноманітних факторів, таких як рівень рентабельності, собівартість та податкова ставка.

Ціну робіт можна визначити за формулою:

$$Ц = С_в \cdot (1 + Р_{рен.}) \cdot (1 + ПДВ) \quad (4.9)$$

де: $С_в$ – собівартість; $Р_{рен.}$ – рівень рентабельності; ПДВ – ставка податку на додану вартість.

Ц = 70 322,93 * (1+0,3) * (1+0,2) = 108 640,94 грн.

4.8 Визначення економічної ефективності і терміну окупності капітальних вкладень

Ефективність виробництва – це узагальнене і повне відображення кінцевих результатів використання робочої сили, засобів та предметів праці на підприємстві за певний проміжок часу.

Для визначення ефективності продукту розраховують чисту теперішню вартість (ЧТВ) і термін окупності ($Т_{ок}$).

$$ЧТВ = -С_в + \sum_{i=1}^{t} \frac{Г_П}{(1+i)^t} \quad (4.10)$$

де: $С_в$ – собівартість розробки; $Г_п$ – грошовий потік за t – ий рік; t – відповідний рік проекту; і – величина дисконтної ставки (10…15%).

$$ЧТВ = -69\,641{,}63 + \frac{38\,999{,}31}{(1+0{,}1)^1} + \frac{38\,999{,}31}{(1+0{,}1)^2} + \frac{38\,999{,}31}{(1+0{,}1)^3}$$

$$= \ 27\,343{,}88 \ \text{грн}$$

Якщо ЧТВ ≥ 0, то проект може бути рекомендований до впровадження.

Термін окупності визначається за формулою:

$$Т_{ок} = Т_{пв} + \frac{Н_в}{Г_{пр}} \quad (4.11)$$

де: $Т_{пв}$ – період до повного відшкодування витрат, років; $Н_в$ –

невідшкодовані витрати на початок року, грн.; $Г_{пр}$ – грошовий потік на початок року, грн.

$$Т_{ок} = 2 + \frac{1\,956{,}88}{38\,999{,}31} \approx 2{,}1 \text{ р.}$$

Всі дані внесемо в зведену таблицю 4.5.

Таблиця 4.5 – Техніко-економічні показники вебсайту магазину книг «BookFlow»

№ п/п	Показник	Значення
1.	Собівартість, грн.	69 641,63
2.	Плановий прибуток або грошовий потік, грн.	38 999,31
3.	Ціна, грн.	108 640,94
4.	Чиста теперішня вартість, грн.	27 343,88
5.	Термін окупності, рік	2,1

Прибутковість проєкту та термін окупності свідчать про його фінансову ефективність та здатність повернути капітальні вкладення протягом 2,1 року. Отже, на основі отриманих показників можна зробити висновок, що розробка вебсайту магазину книг «BookFlow» є доцільною з економічної точки зору.

5 ОХОРОНА ПРАЦІ, ТЕХНІКА БЕЗПЕКИ ТА ЕКОЛОГІЧНІ ВИМОГИ

Умови праці розробника програмного забезпечення характеризуються тривалою роботою за персональним комп'ютером, високим зоровим та нервово-емоційним напруженням, а також гіподинамією. Тому дотримання норм охорони праці є критично важливим для збереження здоров'я працівника та забезпечення високої продуктивності. У даному розділі розглянуто правові аспекти регулювання умов праці через колективний договір та проаналізовано вплив параметрів мікроклімату на організм людини в умовах офісного або віддаленого робочого місця ІТ-спеціаліста.

5.1 Укладання колективного договору. Його зміст

Колективний договір – це основний локальний нормативно-правовий акт, що регулює виробничі, трудові та соціально-економічні відносини між роботодавцем і працівниками на підприємстві, в установі чи організації. Правовою основою для його розробки є Кодекс законів про працю України (КЗпП) та Закон України «Про колективні договори і угоди».

Для ІТ-компаній або відділів розробки укладання колективного договору є важливим інструментом, оскільки він дозволяє закріпити специфічні умови праці програмістів, які не деталізовані в загальному законодавстві (наприклад, компенсації за використання власної техніки при віддаленій роботі, гнучкий графік, додаткові перерви для зняття зорової втоми).

Порядок укладання колективного договору:

1. Ініціювання переговорів. Будь-яка зі сторін (роботодавець або профспілка/представники колективу) може письмово повідомити іншу сторону про початок колективних переговорів.

2. Утворення робочої комісії. Сторони створюють спільну робочу комісію для підготовки проєкту договору.

3. Розробка проєкту. Комісія збирає пропозиції від працівників, формує

проєкт та узгоджує спірні питання.

4. Обговорення та схвалення. Проєкт виноситься на обговорення трудового колективу і схвалюється загальними зборами (конференцією).

5. Підписання. Уповноважені представники сторін підписують договір.

6. Реєстрація. Договір підлягає повідомній реєстрації в місцевих органах виконавчої влади або органах місцевого самоврядування.

Зміст договору визначається сторонами в межах їх компетенції. Зазвичай він містить такі основні розділи:

– організація виробництва і праці: забезпечення продуктивної зайнятості, створення умов для ефективної роботи (наприклад, надання сучасного апаратного та програмного забезпечення для веброзробки);

– нормування і оплата праці: встановлення форм, систем і розмірів заробітної плати, премій, доплат (зокрема за понаднормову роботу або роботу у вихідні дні під час релізів (деплою) проєктів);

– режим роботи і час відпочинку: визначення тривалості робочого дня та тижня, встановлення гнучкого графіка роботи, який є популярним у сфері IT, графіка відпусток;

– охорона праці: чи не найважливіший розділ для розробників. Він включає зобов’язання роботодавця щодо проведення атестації робочих місць, забезпечення ергономічними меблями (крісла, столи для роботи стоячи), організацію регулярних медичних оглядів (особливо перевірка зору та опорно-рухового апарату), а також встановлення регламентованих перерв при роботі з моніторами (зазвичай 10-15 хвилин кожні 2 години);

– соціальне обслуговування та гарантії: медичне страхування працівників (що є стандартом для IT-галузі), компенсація витрат на спортзал, навчання чи курси підвищення кваліфікації.

Укладений колективний договір є гарантом того, що права розробника будуть захищені, а умови праці відповідатимуть сучасним стандартам безпеки та комфорту.

5.2 Вплив параметрів мікроклімату на організм людини

Робота веброзробника належить до категорії легких фізичних робіт Іа відповідно до ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень»[25], оскільки виконується сидячи та не потребує значних фізичних навантажень.

Мікроклімат виробничих приміщень – це клімат внутрішнього середовища, який визначається комплексом фізичних факторів, що діють на людину. Основними параметрами мікроклімату є:

- температура повітря;
- відносна вологість;
- швидкість руху повітря;
- інтенсивність теплового випромінювання.

Вплив температури повітря.

Температура є головним фактором, що впливає на теплообмін між організмом та навколишнім середовищем. Для робіт категорії Іа оптимальною вважається температура 22-24°С у холодний період року та 23-25°С у теплий.

У разі підвищеної температури організм схильний до перегрівання, рясного потовиділення, втрати солей та водорозчинних вітамінів. Таким чином у розробника знижується концентрація уваги, сповільнюються реакції, з'являється млявість, що значно підвищує ризик допущення синтаксичних та логічних помилок при написанні програмного коду.

Знижена температура, у свою чергу, викликає переохолодження організму, спазм кровоносних судин. Тривала робота за клавіатурою в холодному приміщенні призводить до зниження рухливості пальців (скутості), що сповільнює швидкість набору коду. Крім того, виникає ризик застудних захворювань та запалення суглобів.

Вплив відносної вологості повітря.

Оптимальний рівень відносної вологості для приміщень, де експлуатується обчислювальна техніка, становить 40-60% [25].

– Низька вологість (< 40%). Це дуже поширена проблема в офісах під час опалювального сезону або при постійній роботі кондиціонерів. Сухе повітря викликає пересихання слизових оболонок дихальних шляхів та очей. Для програміста, який постійно фокусує погляд на моніторі та рідше кліпає, сухе повітря різко прискорює розвиток «синдрому сухого ока» (різь в очах, почервоніння, швидка зорова втома). Також сухе повітря сприяє накопиченню статичної електрики, що може вплинути на роботу електронної техніки.

– Висока вологість (> 75%). Ускладнює тепловіддачу організму шляхом випаровування поту. У поєднанні з високою температурою це створює ефект «духоти», що викликає швидку втому, відчуття важкості та головний біль, унеможливлюючи продуктивну інтелектуальну працю.

Вплив швидкості руху повітря.

Швидкість руху повітря допомагає підтримувати нормальний теплообмін. Оптимальна швидкість для робочого місця програміста становить не більше 0,1м/с.

– Застоювання повітря. Відсутність вентиляції призводить до накопичення вуглекислого газу (CO_2) у приміщенні. Для розумової праці це критично: підвищений рівень CO_2 викликає сонливість, зниження когнітивних функцій та здатності до вирішення складних алгоритмічних задач.

– Протяги (швидкість > 0,2-0,3 м/с). Спрямовані потоки холодного повітря (наприклад, від неправильно налаштованого кондиціонера або відкритого вікна) спричиняють локальне переохолодження, що може призвести до невралгії, міозиту (запалення м'язів шиї та спини) або радикуліту, зважаючи на малорухливу позу працівника.

Вимоги до організації робочого місця програміста.

Відповідно до ДСанПіН 3.3.2.007-98 та ДСТУ ISO 9241 робоче місце користувача ПК повинно відповідати таким вимогам:

– площа на одне робоче місце — не менше 6 м²;

– об’єм приміщення — не менше 20 м³;

– рівень освітленості — 300–500 лк;

– монітор слід розташовувати боком до вікна для уникнення відблисків;

– верхня межа екрана повинна бути на рівні очей або трохи нижче;

– клавіатура повинна розташовуватись на відстані 100–300 мм від краю столу.

Також важливим є дотримання правильної робочої пози:

– спина повинна мати опору;

– ноги мають стояти на підлозі або спеціальній підставці;

– кут згину рук у ліктях — приблизно 90°.

Заходи щодо нормалізації мікроклімату.

Для забезпечення комфортних та безпечних умов праці веброзробника необхідно здійснювати комплекс організаційних і технічних заходів:

1. Використання систем вентиляції та кондиціонування повітря відповідно до ДБН В.2.5-67:2013;
2. Регулярне очищення фільтрів кондиціонерів та вентиляційних систем;
3. Використання зволожувачів повітря в опалювальний період;
4. Проведення регулярного провітрювання приміщень;
5. Раціональне розміщення робочих місць відносно джерел тепла та потоків повітря;
6. Організація регламентованих перерв для зниження зорового та нервово-емоційного навантаження;
7. Використання ергономічних меблів та сучасної комп’ютерної техніки.

Дотримання перелічених вище вимог дозволить зберегти здоров'я працівника, знизити рівень професійної втоми та забезпечити високу ефективність процесу розробки програмного забезпечення.

ВИСНОВКИ

Фінальним підсумком кваліфікаційної роботи є досягнення поставленої мети – спроєктований та реалізований повноцінний вебсайт магазину книг «BookFlow».

На початковому етапі розробки було проведено глибокий аналіз предметної області, досліджено потреби цільової аудиторії та проаналізовано існуючі ринкові рішення. На основі отриманих даних було сформовано чітке технічне завдання, розроблено оптимальну клієнт-серверну архітектуру та спроєктовано нормалізовану реляційну структуру бази даних PostgreSQL, що гарантує цілісність, надійність та швидкість обробки інформації.

Програмну реалізацію серверної частини вебсайту виконано з використанням сучасного фреймворку Spring Boot, що забезпечило високу стабільність та масштабованість системи. Побудовано ефективний REST API для безперебійного обміну даними між клієнтською частино і серверною. Особливу увагу в роботі було приділено питанням кібербезпеки, а саме реалізовано механізм шифрування конфіденційних даних та впроваджено надійну систему автентифікації і авторизації користувачів на базі JWT-токенів із чітким розмежуванням прав доступу. Проведене комплексне тестування підтвердило працездатність та стійкість усіх модулів системи.

Клієнтську частину вебзастосунку створено на основі бібліотеки React із застосуванням інструменту збірки Vite, тим самим суттєво оптимізувавши процес розробки та підвищивши загальну продуктивність і швидкість завантаження сторінок. Розроблений користувацький інтерфейс є інтуїтивно зрозумілим і повністю адаптивним, гарантуючи коректне відображення як на десктопних, так і на мобільних пристроях. Використання препроцесора SCSS забезпечило гнучку стилізацію, створення унікальних кастомних компонентів та плавних анімацій, що значно покращує користувацький досвід.

Для забезпечення подальшого життєвого циклу продукту було розроблено детальні інструкції з розміщення вебсайту в мережі Інтернет, а сам програмний

продукт успішно розгорнуто та розміщено на хмарній хостинговій платформі, що дозволило підтвердити його повну доступність та стабільність функціонування у реальному мережевому середовищі. Додатково було описано етапи технічного обслуговування, алгоритми наповнення каталогу контентом та запропоновано дієві стратегії маркетингового просування готового вебзастосунку.

В економічному розділі кваліфікаційної роботи було проведено техніко-економічне обґрунтування проєкту, тобто розраховано витрати на створення програмного продукту, обчислено його собівартість та визначено договірну ціну, що підтверджує комерційну життєздатність розробки.

У розділі охорони праці, техніки безпеки та екологічних вимог проаналізовано правові аспекти укладання колективного договору як інструменту регулювання трудових відносин. Також детально розглянуто вплив параметрів мікроклімату на організм інженера-програміста та розроблено комплекс заходів щодо нормалізації умов праці для збереження здоров'я і підвищення працездатності.

Підсумовуючи, можна стверджувати, що мету та всі поставлені завдання кваліфікаційної роботи виконано в повному обсязі. Розроблений програмний продукт «BookFlow» має високу практичну цінність, відповідає стандартам сучасної веброзробки і повністю готовий до впровадження та подальшої експлуатації.

ПЕРЕЛІК ПОСИЛАНЬ

1) Видавництво книг Yakaboo Publishing. URL: https://www.yakaboo.ua/ (дата звернення: 24.04.2026).

2) Інтернет-магазин книг Vivat. Книгарня для серця і розуму. URL: https://vivat.com.ua/ (дата звернення: 24.04.2026).

3) Інтернет-магазин книг – Книгарня Є. Завжди дешевше. URL: https://book-ye.com.ua/ (24.04.2026).

4) Find out what websites are built with. Wappalyzer. URL: https://www.wappalyzer.com/ (дата звернення: 26.04.2026).

5) Конструктор сайтів | HOSTiQ Wiki. URL: https://hostiq.ua/wiki/ukr/about-sitebuilder/ (дата звернення: 28.04.2026).

6) Figma: The Collaborative Interface Design Tool. Figma. URL: https://www.figma.com/ (дата звернення: 01.05.2026).

7) OAuth 2.0 аутентифікація через Google | Комп'ютерна школа Hillel. URL: https://blog.ithillel.ua/articles/oauth-2-0-autentifikaciya-cerez-google-yak-realizuvati-vxid-cerez-google-na-saiti (дата звернення: 06.05.2026).

8) Розгортання React проекту за допомогою Vite, 2024. YouTube. URL: https://www.youtube.com/watch?v=wL1t3FLgZJo (дата звернення: 08.05.2026).

9) Node.js. W3Schools українською онлайн. URL: https://w3schoolsua.github.io/nodejs/index.html#gsc.tab=0 (дата звернення: 09.05.2026).

10) Sass: syntactically awesome style sheets. URL: https://sass-lang.com/ (дата звернення: 09.05.2026).

11) Context – React. React – A JavaScript library for building user interfaces. URL: https://legacy.reactjs.org/docs/context.html (дата звернення: 11.05.2026).

12) Поняття куки, кеш, сесія в браузері. QATestLab | Головна сторінка. URL: https://training.qatestlab.com/blog/technical-articles/cookies-cache-session-in-browser/ (дата звернення: 13.05.2026).

13) Бібліотека Axios у JavaScript. URL: https://www.it-

notes.wiki/javascript/axios-library-in-javascript/ (дата звернення: 14.05.2026).

14) Neon. PostgreSQL Tutorial. Neon. URL: https://neon.com/postgresql/tutorial (дата звернення: 19.05.2026).

15) SPA в програмуванні: розбираємося з одностранічніками. FoxmindEd. URL: https://foxminded.ua/spa-u-prohramuvanni/ (дата звернення: 22.05.2026).

16) Харовюк В. React та Virtual DOM. WebCraft. URL: https://webscraft.org/blog/react-ta-virtualniy-dom-chomu-onovlennya-storinki-stalo-bliskavichno-shvidkim (дата звернення: 25.05.2026).

17) Вірт-універ. Spring Boot. URL: https://virt.ldubgd.edu.ua/pluginfile.php/356522/mod_resource/content/3/2.1.pdf (дата звернення: 26.05.2026).

18) JWT у Spring Boot і як його реалізувати на практиці. FoxmindEd. URL: https://foxminded.ua/jwt-v-spring-boot/ (дата звернення: 29.05.2026).

19) Image and Video Upload, Storage, Optimization and CDN. Cloudinary. URL: https://cloudinary.com/ (дата звернення: 30.05.2026).

20) Redis. Redis - Real-time data for agents & apps. Redis. URL: https://redis.io/ (дата звернення: 31.05.2026).

21) Quick Start Tutorial | Railway Docs. URL: https://docs.railway.com/quick-start (дата звернення: 01.06.2026).

22) K S. Learn How to Deploy with Vercel: A Quick Guide for Beginners. LinkedIn: Log In or Sign Up. URL: https://www.linkedin.com/pulse/learn-how-deploy-vercel-quick-guide-beginners-sumathi-k-9bv4c/ (дата звернення: 01.06.2026).

23) W3Schools українською. pgAdmin. URL: https://w3schoolsua.github.io/postgresql/postgresql_pgadmin4.html#gsc.tab=0 (дата звернення: 07.06.2026).

24) SEO для FrontEnd-розробників. FoxmindEd. URL: https://foxminded.ua/seo-dlia-frontend-rozrobnykiv/ (дата звернення: 02.06.2026).

25) Санітарні норми мікроклімату приміщень ДСН 3.3.6.042-99. URL: https://zakon.rada.gov.ua/rada/show/va042282-99#Text (дата звернення: 02.06.2026).

ДОДАТКИ

Додаток А. Лістинг файлу «axios.js»

```
import axios from 'axios';

const instance = axios.create({
    baseURL: '/api',
    withCredentials: true,
});
let isRefreshing = false;
let failedQueue = [];

const processQueue = (error, token = null) => {
    failedQueue.forEach(prom => {
        if (error) {
            prom.reject(error);
        } else {
            prom.resolve(token);
        }
    });
    failedQueue = [];
};
instance.interceptors.response.use(
    (response) => response,
    async (error) => {
        const originalRequest = error.config;

        if (
            error.response &&
            (error.response.status === 401 || error.response.status ===
403) &&
            !originalRequest._retry &&
```

```
            !originalRequest.url.includes('/auth/refresh')
        ) {
            if (isRefreshing) {
                return new Promise(function(resolve, reject) {
                    failedQueue.push({ resolve, reject });
                }).then(() => {
                    return instance(originalRequest);
                }).catch(err => {
                    return Promise.reject(err);
                });
            }
            originalRequest._retry = true;
            isRefreshing = true;

            try {
                await      axios.post("/api/auth/refresh",      {},      {
withCredentials: true });
                processQueue(null);
                return instance(originalRequest);

            } catch (refreshError) {
                processQueue(refreshError);
                console.error("Refresh token expired. Logging out...");
                return Promise.reject(refreshError);
            } finally {
                isRefreshing = false;
            }
        }
        return Promise.reject(error);
    }
);

export default instance;
```

Додаток Б. Лістинг файлу «App.jsx»

```
import MainPage from "./pages/MainPage/MainPage.jsx";
import {Routes, Route, Router} from "react-router-dom";
import {AuthModalProvider} from "./context/AuthModalContext.jsx";
import Header from "./components/Header/Header.jsx";
import Footer from "./components/Footer/Footer.jsx";
import CategoryPage from "./pages/CategoryPage/CategoryPage.jsx";
import SearchResults from "./pages/SearchPage/SearchResults.jsx";
import {AuthProvider} from "./context/AuthContext.jsx";
import AuthModal from "./components/AuthModal/AuthModal.jsx";
import ProfilePage from "./pages/ProfilePage/ProfilePage.jsx";
import RegisterPage from "./pages/RegisterPage/RegisterPage.jsx";
import WishlistPage from "./pages/WishlistPage/WishlistPage.jsx";
import {CategoryProvider} from "./context/CategoryContext.jsx";
import ProductPage from "./pages/ProductPage/ProductPage.jsx";
import                          ArchivedBooksPage                          from
"./pages/ArchivedBooksPage/ArchivedBooksPage.jsx";
import {CartProvider} from "./context/CartContext.jsx";
import {CartModalProvider} from "./context/CartModalContext.jsx";
import CartModal from "./components/CartModal/CartModal.jsx";
import CheckoutPage from "./pages/CheckoutPage/CheckoutPage.jsx";
import AdminOrdersPage from "./pages/AdminOrdersPage/AdminOrdersPage.jsx";
import { Toaster } from 'react-hot-toast';
import NotFoundPage from "./pages/NotFoundPage/NotFoundPage.jsx";
import DeliveryPage from "./pages/footer/DeliveryPage/DeliveryPage.jsx";
import ScrollToTop from "./context/ScrollToTop.jsx";
import ReturnsPage from "./pages/footer/ReturnsPage.jsx";
import OurStoresPage from "./pages/footer/OurStoresPage.jsx";
import PrivacyPage from "./pages/footer/PrivacyPage.jsx";
import OfferPage from "./pages/footer/OfferPage.jsx";

function App() {
    return (
```

```
        <CartProvider>
            <CartModalProvider>
                <AuthProvider>
                    <AuthModalProvider>
                        <CategoryProvider>

                            <ScrollToTop />

                            <Header/>
                            <Toaster                position="bottom-right"
reverseOrder={false} />

                            <Routes>
                                <Route path="/" element={<MainPage/>}/>
                                <Route                path="/category/:id"
element={<CategoryPage/>}/>
                                <Route                      path="/search"
element={<SearchResults/>}/>
                                <Route                     path="/profile"
element={<ProfilePage/>}/>
                                <Route                    path="/register"
element={<RegisterPage/>}/>
                                <Route                    path="/wishlist"
element={<WishlistPage/>}/>
                                <Route                    path="/checkout"
element={<CheckoutPage/>} />
                                <Route                    path="/book/:id"
element={<ProductPage/>}/>

                                <Route                    path="/delivery"
element={<DeliveryPage />} />
                                <Route                     path="/returns"
element={<ReturnsPage />} />
                                <Route                  path="/our-stores"
```

```
element={<OurStoresPage />} />
                                <Route                          path="/privacy"
element={<PrivacyPage />} />
                                <Route   path="/offer"   element={<OfferPage
/>} />

                                <Route                  path="/admin/orders"
element={<AdminOrdersPage/>} />
                                <Route               path="/archived-books"
element={<ArchivedBooksPage/>}/>

                                    <Route path="*" element={<NotFoundPage/>}/>
                            </Routes>

                            <Footer/>

                            <AuthModal/>
                            <CartModal/>
                        </CategoryProvider>
                    </AuthModalProvider>
                </AuthProvider>
            </CartModalProvider>
        </CartProvider>
    )
}

export default App
```

Додаток В. Лістинг файлу «main.jsx»

```
import {StrictMode} from 'react'
import {createRoot} from 'react-dom/client'
import {BrowserRouter} from "react-router-dom";
import App from './App.jsx'
import './styles/main.scss'
import {GoogleOAuthProvider} from '@react-oauth/google';

const clientId = import.meta.env.VITE_GOOGLE_CLIENT_ID;

createRoot(document.getElementById('root')).render(
    <StrictMode>
        <BrowserRouter>
            <GoogleOAuthProvider clientId={clientId}>
                <App/>
            </GoogleOAuthProvider>
        </BrowserRouter>
    </StrictMode>,
)
```

Додаток Г. Лістинг файлу «MainPage.jsx»

```
import { useEffect, useState } from 'react';
import api from "../../axios.js";
import './MainPage.scss'
import BannerTop from "../../components/BannerTop/BannerTop.jsx";
import BannerLow from "../../components/BannerLow/BannerLow.jsx";
import MainSwiper from "../../components/MainSwiper/MainSwiper.jsx";
import {ProductSlider} from "../../components/ProductSlider/ProductSlider.jsx";

const MainPage = () => {
    const [latestBooks, setLatestBooks] = useState([]);
    const [bestsellerBooks, setBestsellerBooks] = useState([]);

    useEffect(() => {
        const fetchLatestBooks = async () => {
            try {
                const res = await api.get('/books?page=0&size=30&sortBy=id&direction=desc');

                const availableLatest = res.data.content
                    .filter(book => book.quantity > 0)
                    .slice(0, 7);
                setLatestBooks(availableLatest);

                const bestsellersRes = await api.get('/books?page=0&size=30&sortBy=price&direction=desc');

                const availableBestsellers = bestsellersRes.data.content
                    .filter(book => book.quantity > 0)
                    .slice(0, 7);
                setBestsellerBooks(availableBestsellers);
            } catch (err) {
```

```
                console.error("Помилка при завантаженні книг для головної
сторінки:", err);
            }
        };

        fetchLatestBooks();
    }, []);

    return (
        <div className="main-page">
            <MainSwiper />

            {latestBooks.length > 0 && (
                <ProductSlider
                    products={latestBooks}
                    productsPerScreen={4}
                    headline="новинки"
                />
            )}
            <BannerTop />

            {bestsellerBooks.length > 0 && (
                <ProductSlider
                    products={bestsellerBooks}
                    productsPerScreen={4}
                    headline="бестселери"
                />
            )}

            <BannerLow />
        </div>
    )
}
export default MainPage
```

Додаток Д. Лістинг файлу «AuthContext.jsx»

```
import { createContext, useState, useContext, useEffect } from "react";
import api from "../axios.js";

const AuthContext = createContext();

export const AuthProvider = ({ children }) => {
    const [user, setUser] = useState(null);
    const [loading, setLoading] = useState(true);

    const login = (userData) => {
        setUser(userData.user);
    };

    const logout = async () => {
        try {
            await api.post("/auth/logout");
        } catch (err) {
            console.error("Помилка при виході з сервера:", err);
        } finally {
            setUser(null);
        }
    };

    useEffect(() => {
        const checkAuth = async () => {
            try {
                const res = await api.get("/auth/me");
                setUser(res.data);
            } catch (e) {
                setUser(null);
            } finally {
                setLoading(false);
```

```
            }
        };
        checkAuth();
    }, []);

    return (
        <AuthContext.Provider value={{ user, setUser, login, logout,
isAuth: !!user, loading }}>
            {!loading && children}
        </AuthContext.Provider>
    );
};

export const useAuth = () => useContext(AuthContext);
export default AuthContext;
```

Додаток Е. Лістинг файлу «CheckoutPage.jsx»

```
import React, { useState, useEffect } from "react";
import { useNavigate, Link } from "react-router-dom";
import { useCart } from "../../context/CartContext";
import { useAuth } from "../../context/AuthContext";
import api from "../../axios";
import axios from "axios";
import "./CheckoutPage.scss";
import { PatternFormat } from "react-number-format";
import toast from 'react-hot-toast';

const NP_API_KEY = import.meta.env['VITE_NP_API_KEY'];
const NP_API_URL = import.meta.env['VITE_NP_API_URL'];

export default function CheckoutPage() {
    const { cartItems, cartTotal, clearCart } = useCart();
    const { user, setUser } = useAuth();
    const navigate = useNavigate();

    const [formData, setFormData] = useState({
        recipientName: user?.username || "",
        recipientPhone: ""
    });

    const [citySearch, setCitySearch] = useState("");
    const [selectedCityRef, setSelectedCityRef] = useState("");
    const [selectedCityName, setSelectedCityName] = useState("");

    const [warehouseRef, setWarehouseRef] = useState("");
    const [warehouseName, setWarehouseName] = useState("");

    const [cities, setCities] = useState([]);
    const [warehouses, setWarehouses] = useState([]);
```

```
const [isLoading, setIsLoading] = useState(false);
const [error, setError] = useState("");

useEffect(() => {
    if (cartItems.length === 0) {
        navigate("/");
    }
}, [cartItems, navigate]);

useEffect(() => {
    if (citySearch.length < 2) {
        setCities([]);
        return;
    }

    const delayDebounce = setTimeout(async () => {
        try {
            const res = await axios.post(NP_API_URL, {
                apiKey: NP_API_KEY,
                modelName: "Address",
                calledMethod: "getCities",
                methodProperties: {
                    FindByString: citySearch,
                    Limit: "50"
                }
            });
            if (res.data.success) {
                setCities(res.data.data);
            }
        } catch (err) {
            console.error("Помилка завантаження міст з НП:", err);
        }
    }, 500);
```

```
        return () => clearTimeout(delayDebounce);
    }, [citySearch]);

    useEffect(() => {
        if (!selectedCityRef) {
            setWarehouses([]);
            return;
        }

        const fetchWarehouses = async () => {
            try {
                const res = await axios.post(NP_API_URL, {
                    apiKey: NP_API_KEY,
                    modelName: "Address",
                    calledMethod: "getWarehouses",
                    methodProperties: {
                        CityRef: selectedCityRef
                    }
                });
                if (res.data.success) {
                    setWarehouses(res.data.data);
                }
            } catch (err) {
                console.error("Помилка завантаження відділень з НП:", err);
            }
        };

        fetchWarehouses();
    }, [selectedCityRef]);

    const handleChange = (e) => {
        setFormData({ ...formData, [e.target.name]: e.target.value });
    };
```

```
    const handleCityChange = (e) => {
        const val = e.target.value;
        setCitySearch(val);

        const matchedCity = cities.find(c => c.Description === val);
        if (matchedCity) {
            setSelectedCityRef(matchedCity.Ref);
            setSelectedCityName(matchedCity.Description);
            setWarehouseRef("");
            setWarehouseName("");
        } else {
            setSelectedCityRef("");
            setSelectedCityName("");
        }
    };

    const handleWarehouseChange = (e) => {
        const val = e.target.value;
        setWarehouseRef(val);

        const matchedWh = warehouses.find(w => w.Ref === val);
        if (matchedWh) {
            setWarehouseName(matchedWh.Description);
        }
    };

    const submitOrder = async (e) => {
        e.preventDefault();
        setError("");

        if            (formData.recipientPhone.includes('_')           ||
formData.recipientPhone.length < 19) {
            setError("Будь ласка, введіть повний номер телефону.");
            return;
```

```
        }

        if (!selectedCityRef || !warehouseRef) {
            setError("Будь ласка, оберіть місто та відділення зі списку.");
            return;
        }

        setIsLoading(true);

        try {
            const freshUserRes = await api.get("/auth/me");
            const freshUser = freshUserRes.data;
            setUser(freshUser);

            let orderId = null;
            const activeOrder = freshUser?.orders?.find(o => o.orderStatus
=== 'ACTIVE' || o.orderStatus === 'active');

            if (activeOrder) {
                orderId = activeOrder.id;
            } else {
                const orderRes = await api.post("/orders", { userId:
freshUser.id });
                orderId = orderRes.data.id;
            }

            for (const item of cartItems) {
                await api.post(`/orders/${orderId}/items`, {
                    bookId: item.id,
                    quantity: item.quantity
                });
            }

            const finalShippingAddress = `м. ${selectedCityName},
```

```
${warehouseName}`;

            await api.patch(`/orders/${orderId}/checkout`, {
                recipientName: formData.recipientName,
                recipientPhone: formData.recipientPhone,
                shippingAddress: finalShippingAddress
            });

            const finalUserRes = await api.get("/auth/me");
            setUser(finalUserRes.data);

            clearCart();
            toast.success("Замовлення успішно оформлено! Очікуйте на
підтвердження.");
            navigate("/profile");

        } catch (err) {
            console.error("Помилка при оформленні:", err);
            const serverMsg = err.response?.data?.message ||
err.response?.data?.error;
            setError(`Виникла помилка: ${serverMsg || 'Перевірте дані та
спробуйте ще раз'}.`);
        } finally {
            setIsLoading(false);
        }
    };

    if (cartItems.length === 0) return null;

    return (
        <div className="checkout-page">
            <div className="checkout-container">
                <h1 className="checkout-title">Оформлення замовлення</h1>
```

```
                <div className="checkout-content">
                    <form className="checkout-form" onSubmit={submitOrder}>
                        <h3>Дані доставки</h3>

                        {error       &&       <div       className="error-
message">{error}</div>}

                        <div className="form-group">
                            <label className="label-make-order">Прізвище та
Ім'я отримувача</label>
                            <input
                                className="input-make-order"
                                type="text"
                                name="recipientName"
                                onChange={handleChange}
                                required
                                placeholder="Бандера Степан"
                                pattern="^[А-ЯІЇЄа-яіїє']+\s[А-ЯІЇЄа-
яіїє']+(\s[А-ЯІЇЄа-яіїє']+)?$"
                                title="Введіть прізвище та ім'я українською
(мінімум 2 слова)"
                            />
                        </div>

                        <div className="form-group">
                            <label       className="label-make-order">Номер
телефону</label>
                            <PatternFormat
                                className="input-make-order"
                                format="+380 (##) ###-##-##"
                                mask="_"
                                value={formData.recipientPhone}
                                onValueChange={(values) => {
                                    setFormData({
```

```
                                    ...formData,
                                    recipientPhone:
values.formattedValue
                                });
                            }}
                            type="tel"
                            name="recipientPhone"
                            required
                            placeholder="+380 (99) 123-45-67"
                        />
                    </div>

                    <div className="form-group">
                        <label       className="label-make-order">Місто
доставки</label>
                        <input
                            className="input-make-order select-city"
                            type="text"
                            list="cities-list"
                            value={citySearch}
                            onChange={handleCityChange}
                            placeholder="Наприклад: Тернопіль"
                            required
                        />
                        <datalist id="cities-list">
                            {cities.map(city => (
                                <option                  key={city.Ref}
value={city.Description}>
                                    {city.AreaDescription} обл.
                                </option>
                            ))}
                        </datalist>
                    </div>
```

```
                        <div className="form-group">
                            <label  className="label-make-order">Відділення
/ Поштомат Нової Пошти</label>
                            <select
                                className="select-city"
                                value={warehouseRef}
                                onChange={handleWarehouseChange}
                                required
                                disabled={!selectedCityRef                ||
warehouses.length === 0}
                            >
                                <option value="" disabled>
                                    {!selectedCityRef  ?  "Спочатку  оберіть
місто..." : "Оберіть відділення..."}
                                </option>
                                {warehouses.map(wh => (
                                    <option key={wh.Ref} value={wh.Ref}>
                                        {wh.Description}
                                    </option>
                                ))}
                            </select>
                        </div>

                        <div className="delivery-info-link">
                            Якщо ви ще не ознайомлені з нашими правилами
оплати та доставки замовлень —
                            <Link   to="/delivery">   перейдіть   за   цим
посиланням</Link>.
                        </div>

                        <button     type="submit"     className="submit-btn"
disabled={isLoading}>
                            {isLoading  ?  "Оформлюємо..."  :  "Підтвердити
замовлення"}
```

```
                        </button>
                    </form>

                    <div className="checkout-summary">
                        <h3>Ваше замовлення</h3>
                        <div className="summary-items">
                            {cartItems.map(item => (
                                <div    key={item.id}    className="summary-
item">
                                    <span>{item.title}
(x{item.quantity})</span>
                                    <span>{item.price    *    item.quantity}
грн</span>
                                </div>
                            ))}
                        </div>
                        <div className="summary-total">
                            <span>До сплати:</span>
                            <span>{cartTotal} грн</span>
                        </div>
                    </div>
                </div>
            </div>
        );
}
```

Додаток Є. Лістинг файлу «application.properties»

```
spring.application.name=BookFlow backend
spring.datasource.url=${DB_URL}
spring.datasource.username=${DB_USERNAME}
spring.datasource.password=${DB_PASSWORD}
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.PostgreSQLDia
lect
logging.level.org.springframework.web=INFO
logging.level.org.springframework.web.filter.CommonsRequestLoggingFilter=IN
FO
server.servlet.context-path=/api
cloudinary.cloud-name=${CLOUDINARY_CLOUD_NAME}
cloudinary.api-key=${CLOUDINARY_API_KEY}
cloudinary.api-secret=${CLOUDINARY_API_SECRET}
jwt.secret=${JWT_SECRET}
jwt.access.expiration=900000
jwt.refresh.expiration=604800000
app.cors.allowed-
origins=${FRONTEND_URL:http://localhost:3000,http://localhost:5173}
google.client.id=${GOOGLE_CLIENT_ID}
spring.mail.host=smtp.gmail.com
spring.mail.port=${MAIL_PORT}
spring.mail.username=${MAIL_USERNAME}
spring.mail.password=${MAIL_PASSWORD}
spring.mail.properties.mail.smtp.auth=true
spring.mail.properties.mail.smtp.starttls.enable=true
spring.data.redis.host=${REDIS_HOST}
spring.data.redis.port=${REDIS_PORT}
spring.data.redis.password=${REDIS_PASSWORD}
```

Додаток Ж. Лістинг файлу «SecurityConfig.java»

```
package org.bookflow.config;

import org.bookflow.security.JwtFilter;
import org.bookflow.service.CustomUserDetailsService;
import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.ProviderManager;
import
org.springframework.security.authentication.dao.DaoAuthenticationProvider;
import
org.springframework.security.config.annotation.method.configuration.EnableM
ethodSecurity;
import
org.springframework.security.config.annotation.web.builders.HttpSecurity;
import
org.springframework.security.config.annotation.web.configuration.EnableWebS
ecurity;
import org.springframework.security.config.http.SessionCreationPolicy;
import org.springframework.security.core.userdetails.UserDetailsService;
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder;
import org.springframework.security.crypto.password.PasswordEncoder;
import org.springframework.security.web.SecurityFilterChain;
import
org.springframework.security.web.authentication.UsernamePasswordAuthenticat
ionFilter;
import org.springframework.web.cors.CorsConfiguration;
import org.springframework.web.cors.CorsConfigurationSource;
import org.springframework.web.cors.UrlBasedCorsConfigurationSource;

import java.util.List;
```

```
@Configuration
@EnableWebSecurity
@EnableMethodSecurity
public class SecurityConfig {
    private final JwtFilter jwtFilter;
    private final CustomUserDetailsService userDetailsService;

    public SecurityConfig(JwtFilter jwtFilter, CustomUserDetailsService
userDetailsService) {
        this.jwtFilter = jwtFilter;
        this.userDetailsService = userDetailsService;
    }
    @Value("${app.cors.allowed-origins}")
    private List<String> allowedOrigins;
    @Bean
    public SecurityFilterChain securityFilterChain(HttpSecurity http)
throws Exception {
        return http
                .cors(cors                                              ->
cors.configurationSource(corsConfigurationSource()))
                .csrf(csrf -> csrf.disable())
                .authorizeHttpRequests(auth -> auth
                        .requestMatchers("/auth/**","/registration",
"/books/**",
                                "/authors/**",          "/categories/**",
"/publishers/**").permitAll()
                        .requestMatchers("/orders/**",        "/users/**",
"/auth/me", "/wishlist/**").authenticated()
                        .anyRequest().permitAll()
                )
                .sessionManagement(session ->

session.sessionCreationPolicy(SessionCreationPolicy.STATELESS)
                )
```

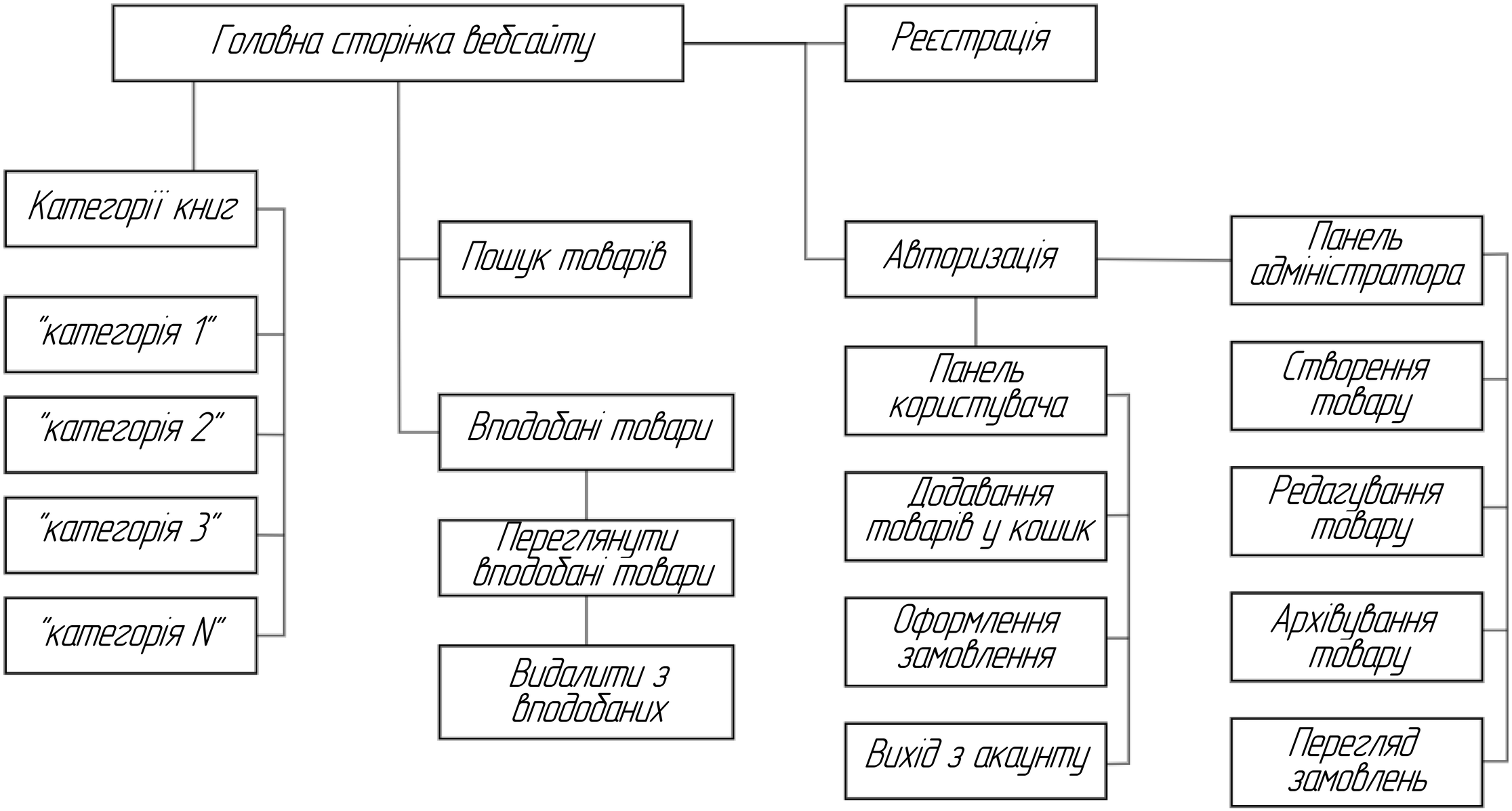
```
                .addFilterBefore(jwtFilter,
UsernamePasswordAuthenticationFilter.class)
                .build();
    }
    @Bean
    public BCryptPasswordEncoder bCryptPasswordEncoder() {
        return new BCryptPasswordEncoder();
    }
    @Bean
    public AuthenticationManager authenticationManager(UserDetailsService
userDetailsService,
                                                       PasswordEncoder
passwordEncoder) {
        DaoAuthenticationProvider authProvider = new
DaoAuthenticationProvider();
        authProvider.setUserDetailsService(userDetailsService);
        authProvider.setPasswordEncoder(passwordEncoder);

        return new ProviderManager(authProvider);
    }
    @Bean
    public CorsConfigurationSource corsConfigurationSource() {
        CorsConfiguration configuration = new CorsConfiguration();
        configuration.setAllowedOrigins(allowedOrigins);
        configuration.setAllowedMethods(List.of("GET", "POST", "PATCH",
"PUT", "DELETE", "OPTIONS"));
        configuration.setAllowedHeaders(List.of("*"));
        configuration.setAllowCredentials(true);
        UrlBasedCorsConfigurationSource source = new
UrlBasedCorsConfigurationSource();
        source.registerCorsConfiguration("/**", configuration);
        return source;
    }
}
```

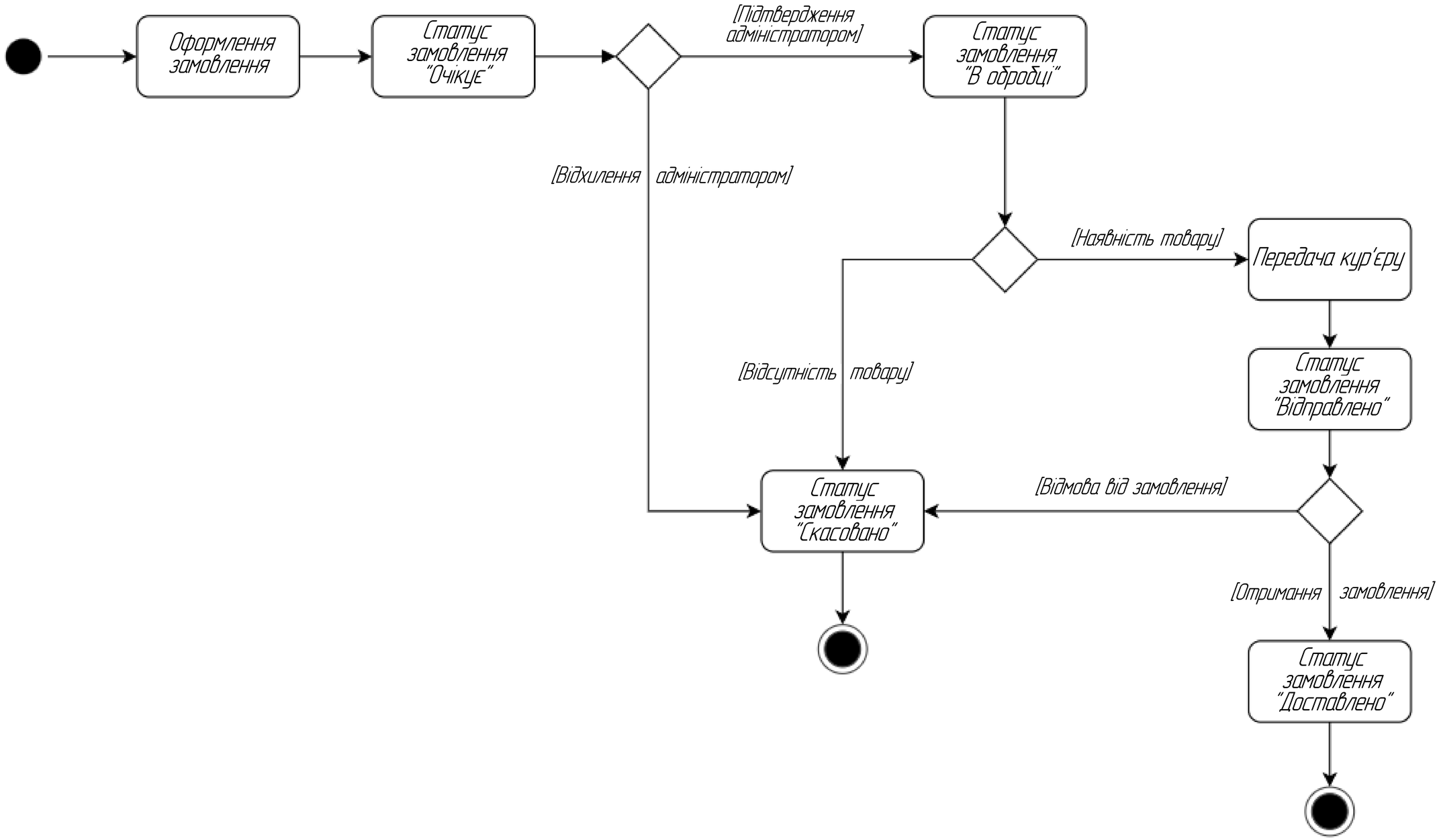
Додаток З. Лістинг модульного тестування серверної частини

```
package org.bookflow;

import org.bookflow.entity.Book;
import org.junit.jupiter.api.Test;

import static org.junit.jupiter.api.Assertions.assertEquals;

public class BookTest {
    @Test
    void testBookGettersAndSetters() {
        Book book = new Book();

        book.setId(1L);
        book.setTitle("Кобзар");
        book.setPrice(350);

        assertEquals(1L, book.getId(), "Ідентифікатор книги має
збігатися");
        assertEquals("Кобзар", book.getTitle(), "Назва книги має
збігатися");
        assertEquals(350, book.getPrice(), "Ціна книги має збігатися");
    }
}
```

Додаток И. Лістинг інтеграційного тестування серверної частини

```
package org.bookflow;

import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import
org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigureMockMv
c;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.http.MediaType;

import                                                          static
org.springframework.test.web.servlet.request.MockMvcRequestBuilders.get;
import                                                          static
org.springframework.test.web.servlet.result.MockMvcResultMatchers.status;

@SpringBootTest
@AutoConfigureMockMvc
public class BookControllerTest {
    @Autowired
    private MockMvc mockMvc;

    @Test
    void testGetBooksEndpoint() throws Exception {
        mockMvc.perform(get("/books")
                        .contentType(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk());
    }
}
```


Головна сторінка вебсайту
Реєстрація
Категорії книг
"категорія 1"
"категорія 2"
"категорія 3"
"категорія N"
Пошук товарів
Вподобані товари
Переглянути вподобані товари
Видалити з вподобаних
Авторизація
Панель користувача
Додавання товарів у кошик
Оформлення замовлення
Вихід з акаунту
Панель адміністратора
Створення товару
Редагування товару
Архівування товару
Перегляд замовлень

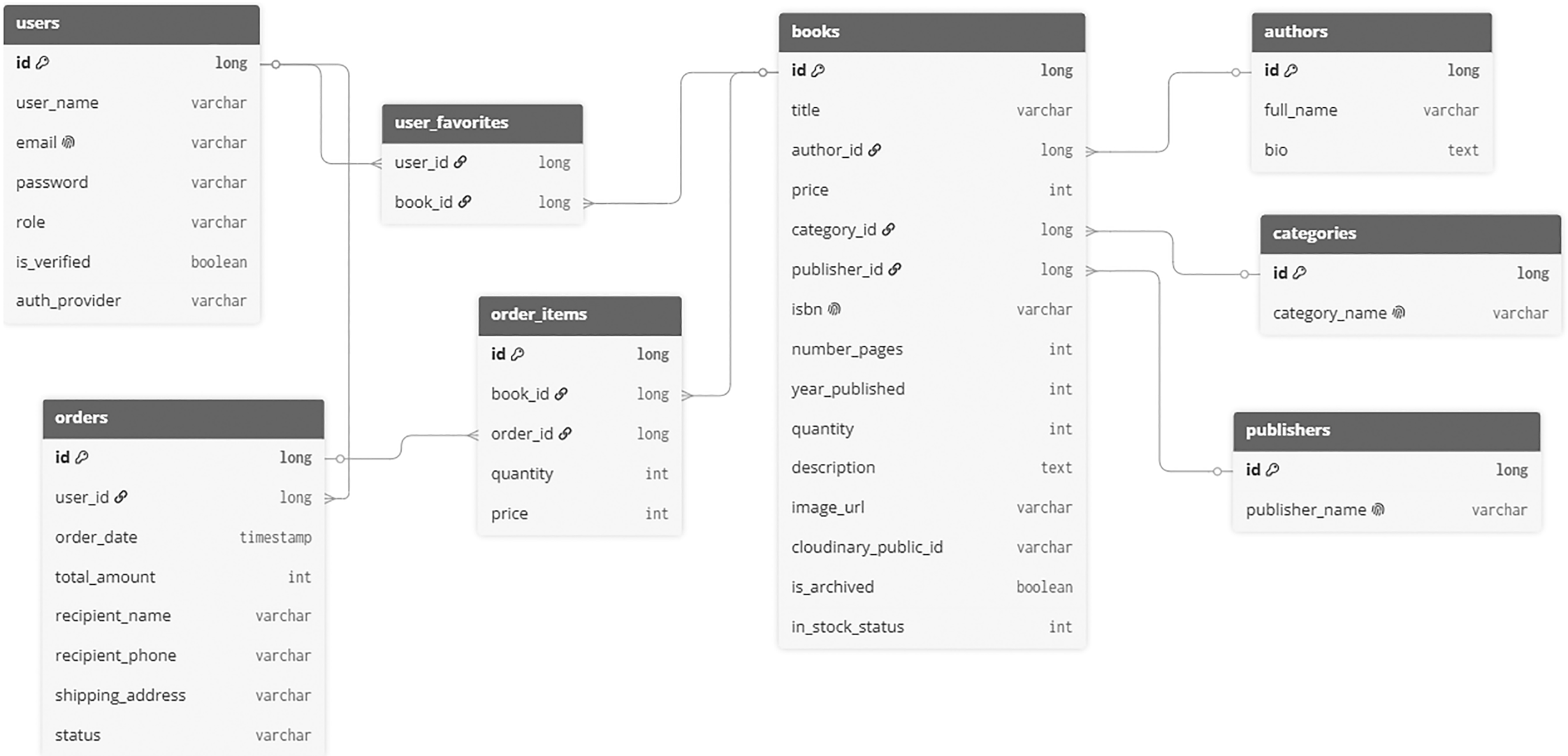


Зм.	Арк.	№ докум.	Підп.	Дата	2026.КВР.122.421.12.00.00 ДС
Розроб.		Кугаївська Є.В.			Розробка вебсайту магазину книг «BookFlow» UML-діаграма станів
Перев.		Слободян Р.О.			
Т.контр.					
Рецензент					
Н.контр.		Прийомак В.А.			
Затв.					

Літ.	Маса	Масштаб
Аркуш	Аркушів 1	
ВСП ТФК ТНТУ КН-421 м. Тернопіль		

Копіював Формат А1

Перш. застос. | Справ. № | Підп. і дата | Інв. № дубл. | Взам. інв. № | Підп. і дата | Інв. № оригін.



Зм.	Арк.	№ докум.	Підп.	Дата	2026.КВР.122.421.12.00.00 БД		
Розроб.		Кугаївська С.В.			Розробка вебсайту магазину книг «BookFlow» ER-діаграма бази даних	Літ.	Маса	Масштаб
Перев.		Слободян Р.О.						
Т.контр.						Аркуш	Аркушів 1	
Рецензент						ВСП ТФК ТНТУ КН-421 м. Тернопіль		
Н.контр.		Приймак В.А.						
Затв.								

Копіював Формат А1

Перш. застос. | Справ. № | Підп. і дата | Інв. № дубл. | Взам. інв. № | Підп. і дата | Інв. № орігін.

Таблиця техніко-економічних показників

№ п/п	Показник	Одиниці вимірювання	Значення
1	Мова програмування	–	Java, JavaScript
2	Технології	–	Spring Boot, React, Vite
3	Менеджер пакетів	–	Maven, npm
4	Система керування БД	–	PostgreSQL
5	Архітектура програмного забезпечення	–	клієнт–серверна
6	Загальний обсяг вихідного коду	МБ	4,02
7	Трудомісткість розробки	год	146
8	Чиста теперішня вартість	грн.	27 343,88
9	Собівартість	грн.	69 641,63
10	Очікуваний прибуток	грн.	38 999,31
11	Термін окупності	рік	2,1