МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії програмного забезпечення

_____ Євген ДАВИДЕНКО

«20» червня 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

ВЕБЗАСТОСУНОК ПРОДАЖУ КНИГ

Спеціальність 121 Інженерія програмного забезпечення

Освітня програма «Інженерія програмного забезпечення»

Здобувачка _____ **Мирослава ДУБІНЕЦЬКА**

«20» червня 2025 р.

Керівник роботи

канд. техн. наук,

доцент _____ **Євген ДАВИДЕНКО**

«20» червня 2025 р.

Миколаїв – 2025

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп'ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО

«28» січня 2025 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувачки

Дубінецької Мирослави

1. Тема кваліфікаційної роботи Вебзастосунок продажу книг затверджена наказом ректора ЧНУ ім. Петра Могили № 315 від «13» листопада 2024 р.
2. Строк представлення кваліфікаційної роботи «18» червня 2025 р.
3. Очікуваний результат роботи та початкові дані, якщо такі потрібні.

Очікуваним результатом є вебзастосуно продажу книг

4. Перелік питань, що підлягають розробці: аналіз предметної області та існуючих аналогів вебзастосунку для продажу книг, визначення вимог до програмного забезпечення, розробка архітектури програмного забезпечення, моделювання та проєктування системи, реалізація програмного забезпечення, тестування та перевірка працездатності системи.

5. Перелік графічних матеріалів: презентація

Дата видачі завдання «23» лютого 2025 р

КАЛЕНДАРНИЙ ПЛАН

виконання кваліфікаційної роботи

Тема: **Вебзастосунок продажу книг**

№ з/п	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	20.02.2025	23.02.2025	Виконано
2.	Огляд літератури за темою роботи	24.02.2025	26.02.2025	Виконано
3.	Складання календарного плану КБР	26.02.2025	27.02.2025	Виконано
4.	Аналіз предметної області	28.02.2025	01.03.2025	Виконано
5.	Розробка проєктних рішень	04.03.2025	10.03.2025	Виконано
6.	Моделювання та конструювання ПЗ	22.03.2025	26.03.2025	Виконано
7.	Кодування, тестування та апробація розробленого ПЗ, аналіз результатів тестування, розробка керівництва користувача	28.03.2025	22.05.2025	Виконано
8.	Відгук керівника КБР	22.05.2025	22.05.2025	Виконано
9.	Оформлення КБР та презентації	23.05.2025	23.05.2025	Виконано
10.	Попередній захист	28.05.2025	28.05.2025	Виконано
11.	Рецензування	12.06.2025	12.06.2025	Виконано
12.	Завершення оформлення КБР та презентації	14.06.2025	15.06.2025	Виконано
13.	Захист кваліфікаційної роботи	24.06.2025	24.06.2025	Виконано

Здобувачка _____ **Мирослава ДУБІНЕЦЬКА**

«27» лютого 2025 р.

Керівник роботи

канд. техн. наук, _____ **Євген ДАВИДЕНКО**

доцент

«27» лютого 2025 р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«Вебзатосунок продажу книг»

Здобувачка 408 гр.: Дубінецька Мирослава Олександрівна

Керівник: канд. техн. наук, доцент Давиденко Євген

В умовах швидкого розвитку цифрових технологій та інтернет-торгівлі, вебзастосунки для продажу товарів набувають все більшого значення. Онлайн-торгівля книгами є однією з найбільш поширених ніщ у сфері електронної комерції. Вебзастосунки для продажу книг дозволяють користувачам отримати доступ до величезного асортименту літератури, що полегшує процес придбання книг у порівнянні з традиційними методами. Крім того, система може пропонувати додаткові функції, такі як рекомендації на основі попередніх покупок, рецензії, інтеграцію з соціальними мережами, що ще більше підвищує її ефективність.

Інтернет-торгівля також має значний вплив на зниження витрат на організацію продажу, оскільки немає потреби у фізичних магазинах та складських приміщеннях. Все це робить тему створення вебзастосунку продажу книг надзвичайно актуальною, з огляду на значні економічні вигоди і попит на таку платформу в глобальному масштабі.

Розробка вебзастосунку для продажу книг допомагає вирішити важливі питання в індустрії електронної комерції, зокрема:

- покращення доступу до літератури для широкої аудиторії;
- розробка та впровадження нових технологій для організації та автоматизації процесу продажу;
- аналіз потреб користувачів і їх взаємодії з платформою для створення персоналізованих рекомендацій та покращення користувацького досвіду;
- забезпечення інтеграції з різноманітними методами оплати та доставки, що дозволяє задовольняти різні потреби користувачів.

Метою кваліфікаційної роботи є розробка вебзастосунку продажу книг для збільшення обсягу продаж.

Об'єктом кваліфікаційної роботи є процес створення вебзастосунку для продажу книг.

Предметом кваліфікаційної роботи є сукупність інструментів для розробки вебзастосунку продажу книг.

Кваліфікаційна робота складається із вступу, 4 розділів, висновків та переліку джерел посилання.

У вступі визначена актуальність теми роботи, предмет і об’єкт роботи, мета роботи та встановлено перелік завдань задля досягнення сформульованої мети.

Перший розділ описує предметну область та наявні аналоги застосунку.

У другому розділі складено специфікацію вимог до програмного забезпечення, описана функціональність застосунку.

Висновки підсумовують етапи розробки застосунку та описують результати виконаної роботи.

Кваліфікаційна бакалаврська робота містить 63 сторінки, 32 рисунки, 7 таблиць, 20 посилань, 1 додаток.

Ключові слова: книгарня, вебзастосунок, стандарти автентифікації, Node.js, Koa, JavaScript, MySQL.

ABSTRACT

to the qualifying bachelor's thesis

«Web Application for Book Sales»

Student of 408 group: Myroslava Dubinetska

Supervisor: candidate technical of Sciences, associate professor Yevhen Davydenko

In the context of the rapid development of digital technologies and online commerce, web applications for product sales are becoming increasingly important. Online book sales are one of the most widespread niches in the e-commerce sector. Web applications for book sales allow users to access a vast assortment of literature, making the book purchasing process easier compared to traditional methods. Additionally, the system can offer extra features such as recommendations based on previous purchases, reviews, and integration with social media, which further enhances its effectiveness.

Online commerce also significantly impacts the reduction of sales organization costs, as there is no need for physical stores or storage spaces. All of this makes the topic of creating a web application for book sales extremely relevant, given the significant economic benefits and demand for such platforms on a global scale.

The development of a web application for book sales helps solve important issues in the e-commerce industry, particularly:

- improving access to literature for a wide audience;
- developing and implementing new technologies for organizing and automating the sales process;
- analyzing user needs and interactions with the platform to create personalized recommendations and improve the user experience;
- ensuring integration with various payment and delivery methods to meet diverse user needs.

The purpose of the qualification work is to develop a web application for book sales to increase sales volume.

The object of the qualification work is the process of creating a web application for book sales.

The subject of the qualification work is the set of tools for developing a web application for book sales.

The qualification work consists of an introduction, 4 sections, conclusions and a list of references.

In the introduction, the relevance of the topic of the work, the subject and object of the work are defined, goal of the work and a list of tasks is outlined in order to achieve the formulated goal.

The first section describes the subject area and existing analogues of the application.

The second section, the software requirements specification is compiled, and the functionality of the application is described.

The conclusions summarize the stages of application development and present the results of the completed work.

The qualifying bachelor's thesis contains 63 pages, 32 illustrations, 7 tables, 20 sources, 1 appendix.

Keywords: bookshop, web application, authentication standards, Node.js, Koa, JavaScript, MySQL

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД	–	База даних
ПЗ	–	Програмне забезпечення
СКБД	–	Система керування базами даних
2FA	–	Two-Factor Authentication
API	–	Application Programming Interface
CRM	–	Customer Relationship Management
CSRF	–	Cross-Site Request Forgery
CSS	–	Cascading Style Sheets
DOM	–	Document Object Model UML
GDPR	–	General Data Protection Regulation
HTML	–	HyperText Markup Language
HTTP	–	HyperText Transfer Protocol
JWT	–	JSON Web Token
NPM	–	Node Package Manager
ORM	–	Object-Relational Mapping
PCI DSS	–	Payment Card Industry Data Security Standard
SQL	–	Structured Query Language
UML	–	Unified Modeling Language
XSS	–	Cross-Site Scripting

ВСТУП

В умовах швидкого розвитку цифрових технологій та інтернет-торгівлі, вебзастосунки для продажу товарів набувають все більшого значення. Онлайн-торгівля книгами є однією з найбільш поширених ніш у сфері електронної комерції. Вебзастосунки для продажу книг дозволяють користувачам отримати доступ до величезного асортименту літератури, що полегшує процес придбання книг у порівнянні з традиційними методами. Крім того, система може пропонувати додаткові функції, такі як рекомендації на основі попередніх покупок, рецензії, інтеграцію з соціальними мережами, що ще більше підвищує її ефективність.

Нещодавні дослідження вказують на те, що з кожним роком популярність покупок книг через мережу Інтернет зростає [1]. У 2022 році 12,7% жителів ЄС купували книги онлайн, а вже у 2023 році цей показник виріс до 13,4%.

Інтернет-торгівля також має значний вплив на зниження витрат на організацію продажу, оскільки немає потреби у фізичних магазинах та складських приміщеннях. Все це робить тему створення вебзастосунку продажу книг надзвичайно актуальною, з огляду на значні економічні вигоди і попит на таку платформу в глобальному масштабі.

Розробка вебзастосунку для продажу книг допомагає вирішити важливі питання в індустрії електронної комерції, зокрема:

- покращення доступу до літератури для широкої аудиторії;
- розробка та впровадження нових технологій для організації та автоматизації процесу продажу;
- аналіз потреб користувачів і їх взаємодії з платформою для створення персоналізованих рекомендацій та покращення користувацького досвіду;
- забезпечення інтеграції з різноманітними методами оплати та доставки, що дозволяє задовольняти різні потреби користувачів.

Метою кваліфікаційної роботи є розробка вебзастосунку продажу книг для збільшення обсягу продаж.

Для досягнення визначеної мети необхідно вирішити такі завдання:

- аналіз предметної галузі;

- оцінка аналогічних застосунків;
- формування вимог до ПЗ;
- визначення функціоналу кожної сторінки застосунку;
- розробка сценаріїв використання;
- створення дизайну вебзастосунку;
- проєктування бази даних книг, авторів, користувачів;
- розробка застосунку;
- тестування системи.

Об'єктом кваліфікаційної роботи є процес створення вебзастосунку для продажу книг.

Предметом кваліфікаційної роботи є сукупність інструментів для розробки вебзастосунку продажу книг.

1 АНАЛІЗ ВЕБЗАСТОСУНКІВ ПРОДАЖУ КНИГ

1.1 Актуальність розробки вебзастосунку продажу книг

У сучасному світі будь-яке підприємство, прагнучи до стабільного розвитку, потребує стратегії переходу в цифровий формат. Видавнича справа не є винятком, демонструючи активну диджиталізацію у відповідь на виклики цифрової ери. Споживачі все частіше віддають перевагу онлайн-купівлі книг, електронним та аудіоформатам, а також персоналізованим порадам щодо вибору літератури. Ці тенденції трансформують книжковий ринок, формуючи нову екосистему.

Згідно з даними дослідження «КнигарняЄ», інтерес до електронних книг за останній рік збільшився більше ніж вдвічі. Основними причинами цього є комфортність читання на електронних пристроях, доступніша вартість електронних копій та швидка можливість завантаження книги відразу після придбання [2].

Аналогічна тенденція зростання онлайн-покупок книжок підтверджується аналітичним звітом від українського інституту книги. Зокрема, 38% опитаних вказали, що протягом року купували книги для забезпечення потреб сім'ї, що майже вдвічі перевищує відсоток тих, хто купував літературу для особистого читання [3].

Одним з основних інструментів цифрової трансформації в цій галузі є розробка вебзастосунків для продажу книжкової продукції. Такі системи надають користувачам можливість:

- оперативно знаходити потрібну книгу за автором, жанром або назвою;
- замовляти друковані видання з доставкою додому;
- створювати особисті списки бажаного та отримувати рекомендації на основі попередніх замовлень.

Це сприяє значному поліпшенню сервісу, збільшує лояльність клієнтів і дозволяє видавництвам та книгарням оптимізувати основні бізнес-процеси.

Впровадження цифрових технологій у книжковій індустрії відкриває широкі перспективи, зокрема:

- використання спеціалізованих CRM-систем дозволяє ефективніше взаємодіяти з покупцями та формувати індивідуальні пропозиції, базуючись на аналізі їхніх інтересів;
- хмарні технології забезпечують масштабування бізнесу без значних капіталовкладень в інфраструктуру;
- електронні вітрини дають змогу швидко оновлювати асортимент товарів, пропонувати акції та змінювати ціни.

В цілому, цифрові рішення дозволяють книжковим магазинам виходити на міжнародні ринки, впроваджувати гнучкі моделі розповсюдження (наприклад, на основі підписки) та швидко реагувати на зміни споживчого попиту. Важливо також враховувати потенційні ризики, серед яких захист персональних даних користувачів та інформації про платежі, відповідність вимогам GDPR, а також необхідність постійного оновлення програмного забезпечення для забезпечення високої продуктивності та безпеки [4].

1.2 Конкурентний аналіз ринку вебзастосунків для продажу книг

Для успішної розробки вебзастосунку з продажу книг необхідно ретельно дослідити всі ключові аспекти: від функціональності та швидкодії до безпеки та зручності для користувача. Щоб створити ефективне рішення, потрібно детально вивчити існуючі аналоги та зрозуміти, чого очікує цільова аудиторія. Це дозволить не тільки задовольнити потреби покупців, але й чітко визначити завдання, а також виявити можливі проблеми ще на етапі проєктування.

Сьогоднішні користувачі очікують від онлайн-магазинів високого рівня сервісу. У випадку з книгами, це означає швидкий доступ до широкого вибору літератури, зручні фільтри за різними параметрами (жанр, автор, мова), а також безпечні та прості способи оплати [5].

Сучасні вебмагазини книг мають ряд важливих переваг:

- постійна доступність: можливість купувати книги в будь-який час доби, незалежно від місця знаходження;

- миттєве отримання цифрових копій: електронні книги стають доступними для завантаження відразу після оплати;
- інтелектуальний пошук та персоналізовані рекомендації: системи використовують алгоритми штучного інтелекту, щоб аналізувати інтереси користувачів і пропонувати їм цікаві новинки;
- інтеграція з соціальними мережами: можливість ділитися відгуками та списками бажань з друзями;
- надійні платіжні системи: підтримка різних способів оплати, включаючи банківські картки, Apple Pay, Google Pay та інші.

Аналіз конкурентів допомагає визначити їхні сильні та слабкі сторони. Наприклад, Amazon Kindle Store відрізняється тісною інтеграцією з електронними рідерми Kindle, а Google Books робить акцент на пошуку та можливості попереднього перегляду. Українські онлайн-книгарні, такі як Yakaboo та «Книгарня Є», вже пропонують підписку на новинки, але не всі з них мають зручні мобільні застосунки або використовують елементи гейміфікації для залучення читачів [6].

Вивчення існуючих програмних рішень дозволяє зрозуміти переваги та недоліки різних підходів до архітектури, функціональності та дизайну інтерфейсу, що є критично важливим для створення конкурентоспроможного продукту.

Для розробки застосунку розглянуто наступні аналоги, для повного розуміння необхідного функціонала:

- Amazon Books;
- Yakaboo;
- Книгарня Є;
- Amazon Books;
- Barnes & Noble.

Yakaboo (рис. 1.1, табл. 1.1) – найбільший український онлайн-магазин книг. Пропонує літературу українською, англійською та іншими мовами, а також електронні книги, комікси, товари для творчості й канцтовари. Орієнтований на український ринок, має зручну доставку по Україні [7].

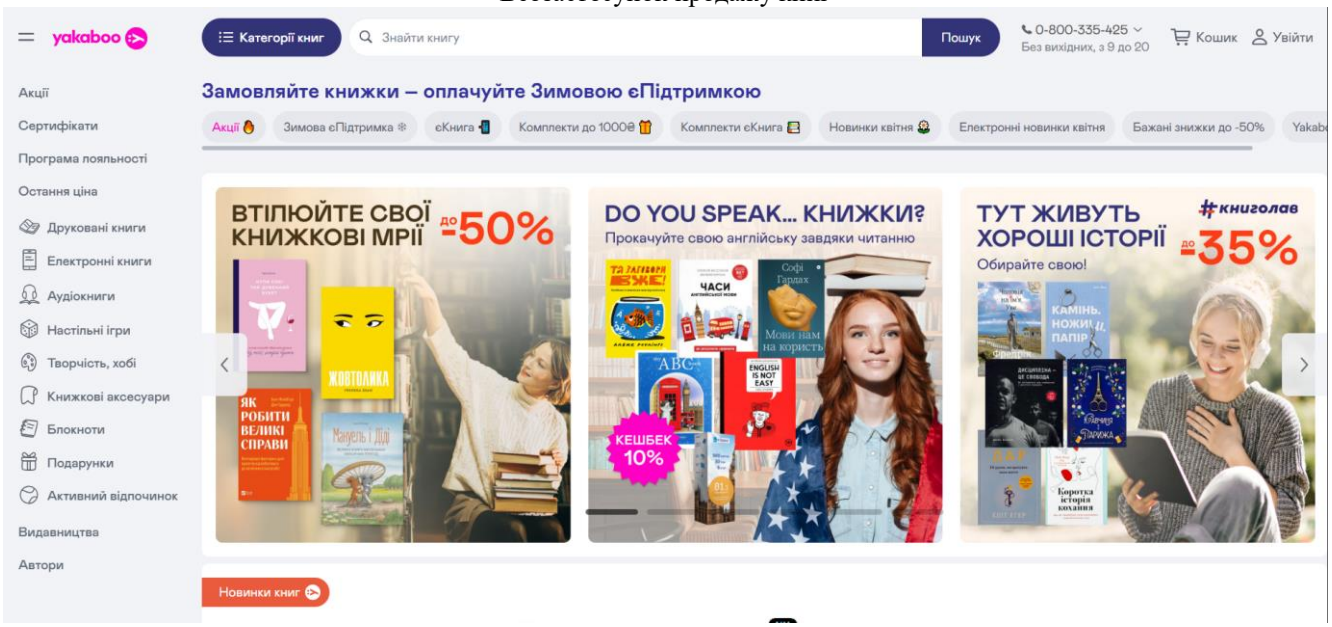

Рисунок 1.1 – Інтерфейс вебзастосунку Yakaboo

Книгарня Є (рис. 1.2, табл. 1.1) – українська мережа книжкових магазинів і онлайн-магазин, що спеціалізується на україномовній літературі. Пропонує широкий асортимент книг від українських видавництв, акцент на культурну, наукову та художню літературу. Має фізичні магазини в різних містах України.

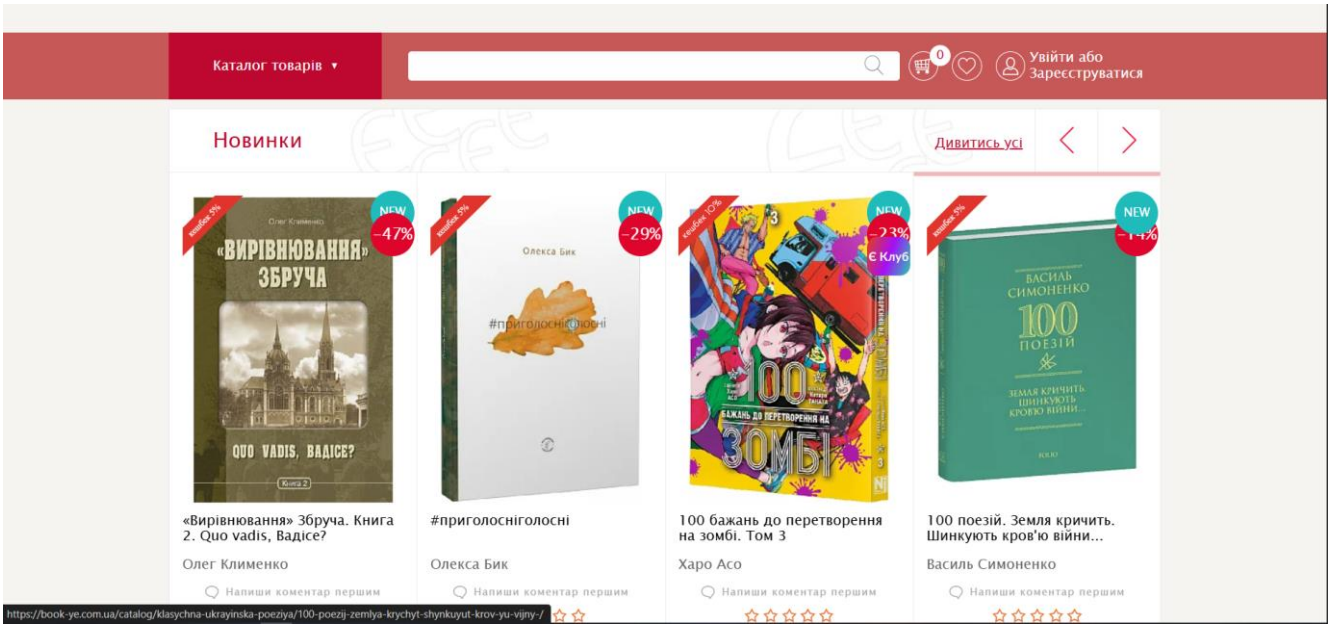

Рисунок 1.2 – Інтерфейс вебзастосунку Книгарня Є

Amazon Books (рис. 1.3, табл. 1.1) – один з найбільших онлайн-магазинів книжок у світі, що є частиною платформи Amazon. Пропонує широкий вибір друкованих книг, електронних (Kindle), аудіокниг, а також рецензії, рейтинги й персональні рекомендації. Доставка доступна в більшість країн [8].

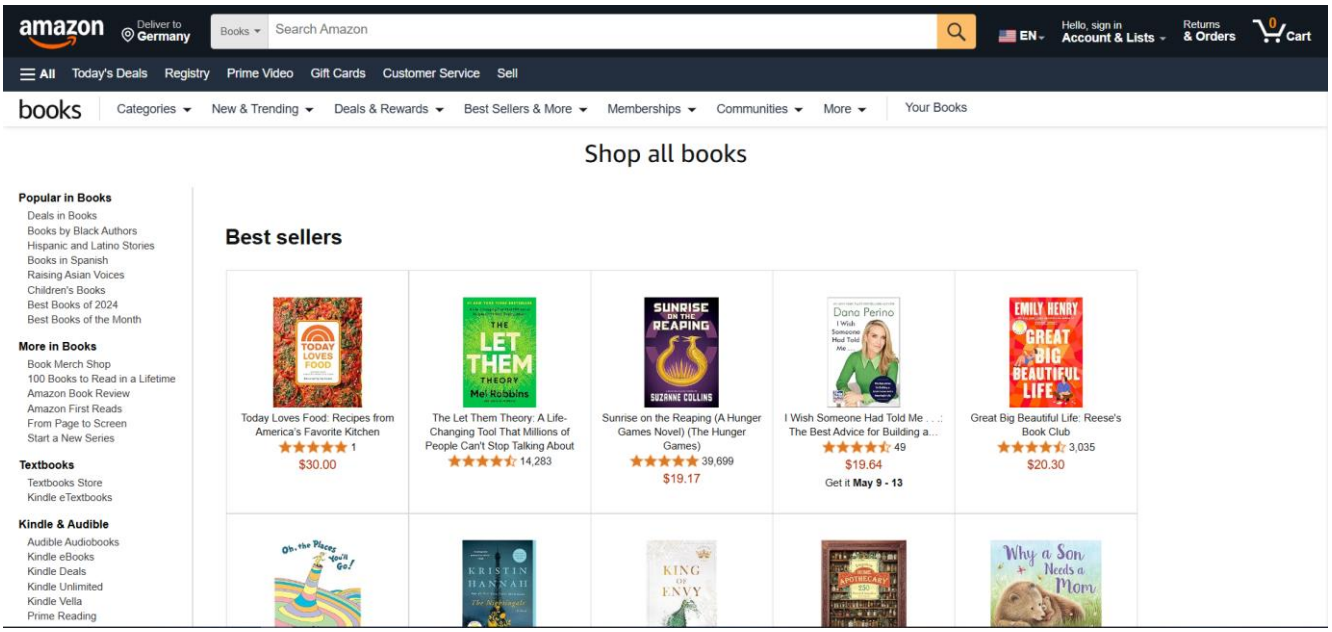

Рисунок 1.3 – Інтерфейс вебзастосунку Amazon Books

Barnes & Noble (рис. 1.4, табл. 1.1) – велика американська книжкова мережа та онлайн-магазин. Пропонує друковані книги, електронні книги (формат Nook), журнали, іграшки, канцелярію. Відомий своїми великими фізичними магазинами в США та активною участю в розвитку комерційної книжкової індустрії [9].

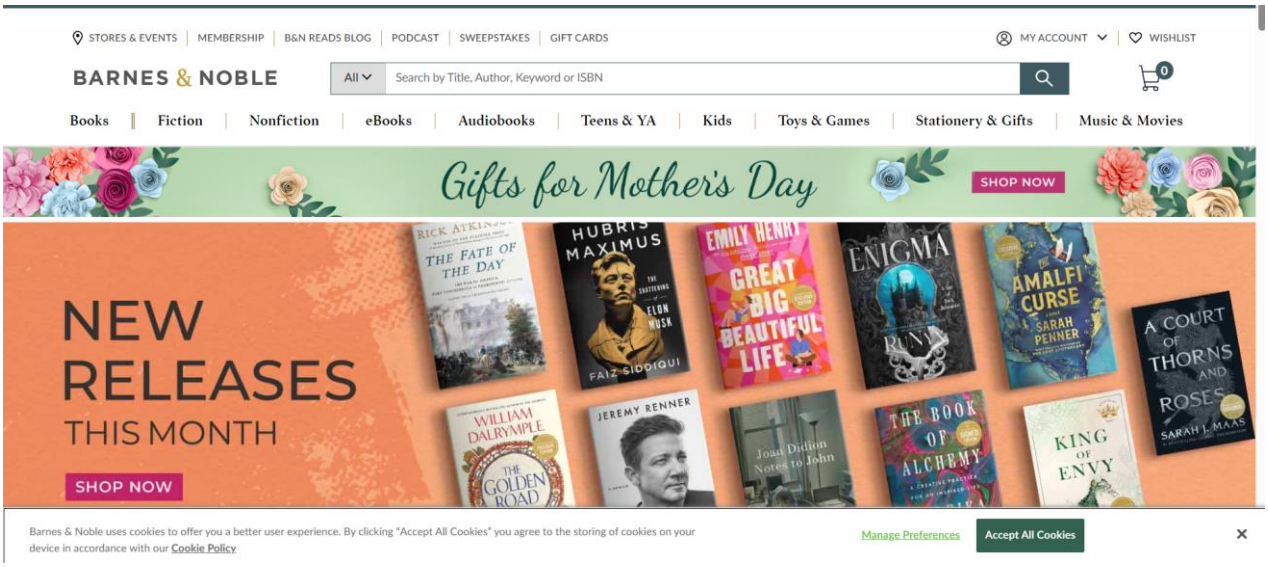

Рисунок 1.4 – Інтерфейс вебзастосунку Barnes & Noble

Таблиця 1.1 – Порівняльна характеристика аналогічних застосунків

Назва	**Архітектура**	**Мова реалізації**	**Основні функції**	**Переваги**	**Недоліки**	**Вебсайт**
Yakaboo	Web, Ios, Android.	JavaScript, PHP, Kotlin, Swift.	Продаж паперових та електронних книг, аудіокниги, рекомендації.	Найбільший український онлайн-магазин книг, українська мова, доставка по Україні.	Вузька інтеграція для eReader-пристроїв, неповна інформація про авторів.	yakaboo.ua
Книгарня Є	Web.	PHP, JavaScript.	Продаж книг українською, фільтри за тематикою та автором.	Підтримка українських видавництв, україномовний інтерфейс.	Немає мобільного застосунку, відсутність деяких жанрів.	book-ye.com.ua
Amazon (Books)	Web, Ios, Android, Kindle.	Java, JavaScript, Swift.	Продаж паперових та електронних книг, Kindle-сумісність, підписки Audible.	Найбільший асортимент, інтеграція з пристроями, швидка доставка.	Доставка в Україну не завжди доступна, часто немає українських книг,не підтримують локальні платіжні системи.	amazon.com/books
Barnes & Noble	Web, Ios, Android, Nook.	Java, JavaScript, Swift.	Продаж книг, eBook-формат Nook, аудіокниги.	Якісні видання, популярний бренд у США, власний eReader (Nook).	Не підтримують локальні платіжні системи, немає українських книг.	www.barnesandnoble.com

Для успішної розробки вебзастосунку з продажу книг необхідно враховувати як функціональні, так і нефункціональні вимоги, орієнтуючись на досвід провідних онлайн-книгарень. Вивчення таких сервісів, як Amazon Books, Yakaboo, «Книгарня Є» та Barnes & Noble, дозволило окреслити основні тенденції у сфері онлайн-продажу книг: постійна доступність, цифрова доставка, персоналізовані рекомендації

та безпечні методи оплати. Аналіз конкурентів дав змогу визначити найкращі практики, а також недоліки, яких варто уникати

Висновки до розділу 1

У першому розділі досліджено предметну область вебзастосунку, що розроблюється, включаючи аналіз частоти здійснення покупок книг через Інтернет. Проведено конкурентний аналіз ринку, від великих міжнародних гігантів до локальних онлайн-крамниць, враховуючи різницю в підходах до аудиторії та культурних особливостей. З'ясовано, що основна кількість недоліків пов’язана з відсутністю повної інформації про авторів та відгуки користувачів, а також обмеження щодо доступу до деяких жанрів книг.

На основі дослідження виявлено сильні та слабкі сторони аналогічних сервісів і націлитися на ті аспекти, які дозволять розробити більш ефективний і доступний застосунок через інтеграцію нових технологій та покращення користувацького досвіду.

2 АНАЛІЗ ІНСТРУМЕНТІВ РОЗРОБКИ

2.1 Вибір технологій для розробки

Основною мовою програмування на клієнтській частині є JavaScript – стандарт для веброзробки, який підтримується всіма сучасними браузерами. JavaScript надає широкі можливості для створення динамічних та інтерактивних інтерфейсів користувача. Хоча мова не має суворої типізації, вона дозволяє швидко створювати прототипи та гнучко адаптуватися до вимог проєкту [10].

Для спрощення процесу розробки було прийнято рішення використовувати JavaScript-фреймворк, який має набір готових інструментів для розробки інтерфейсів, а також забезпечує зручну організацію архітектури застосунку[11]. За даними дослідження журналу Electronics (рис. 2.1), найпопулярнішими фреймворками для веброзробки клієнтської частини є React, Vue та Svelte [12]. Серед цієї трійки найбільшу популярність має фреймворк ReactJS, який активно розвивається, підтримується великим співтовариством і має широку екосистему.

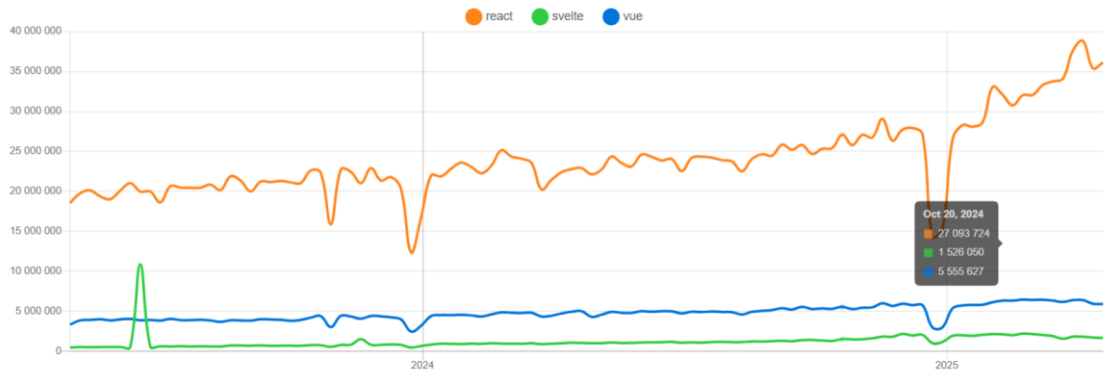

Рисунок 2.1 – Дані з NPM Trends про найбільш популярні фреймворки

React прискорює процес розробки вебзастосунків, тому що дозволяє використовувати модульний підхід, чим забезпечує перевикористання вже розроблених рішень. Також за допомогою React (рис. 2.2) щастосунок можна масштабувати при довгостроковій розробці. Фреймворк містить велику кількість бібліотек і інструментів, які були створені для ReactJS [13]. Дані інструменти значно прискорюють

процес рендеринку завдяки використанню технології Virtual DOM. У цьому випадку застосовується алгоритм зі складністю **O(n)** замість стандартного алгоритму зі складністю **O(n³)**, де *n* – кількість елементів на вебсторінці. Виходячи з наведених даних, було прийнято рішення використовувати React як основний фреймворк для розробки клієнтської частини вебзастосунку .

Рисунок 2.2 – ReactJS v19.1

Для стилізації компонентів використовується технологія CSS-in-JS. Розглянемо переваги кожного. CSS-in-JS – це метод, який дозволяє прописувати CSS стилі в JavaScript файлах, як JavaScript-об'єкти або через шаблонні рядки що дозволяє уникати створення великої кількості непотрібних CSS файлів і директорій. Використання CSS-in-JS найбільш доцільно, коли створюються компоненти, що часто використовуються, наприклад, кнопки, перемикачі, таблиці. Використання такого підходу дозволяє значно спростити динамічну стилізацію: логіка зміни стилів компонентів переміщується с рівня зміни класів модифікаторів на рівень описання самих стилів.

Для реалізації валідації даних у системних формах використано бібліотеку React-hook-form, яка реалізує технологію React-hooks, засновану на принципах реактивного програмування. Ця бібліотека має великий інструментарій для валідації користувальницьких форм

Бібліотекою, що реалізує даний підхід, було обрано Styled-Components, яка «під капотом» реалізує препроцесор StylusJS, який підтримує наступніфункції: посилання на батьківський компонент, зменшення розміру вихідного коду, видалення мертвого коду, вендорні префікси, необхідні для роботи стилів у різних браузерах. Також використовується фреймворк Tailwind CSS [14]. Особливість цієї бібліотеки в тому, що вона не визначає CSS-класи окремих елементів. Натомість вона надає службові класи, які можна поєднувати для стилізації кожного елемента.

Рисунок 2.3 – TailwindCSS v4.1

Для бекендової частини було вирішено використовувати платформу Node.js (рис. 2.4), яка призначена для роботи із серверними скриптами для вебзапитів [15]. Асинхронна моделб запуску коду , заснована на обробці подій в неблокуючому режимі та визначенні обробників зворотніх викликів, дозволяє забезпечувати обробку великої кількості паралельних запитів.

Рисунок 2.4 – Node.js v22.15

Koa (рис. 2.5) – це сучасний фреймворк для створення серверних програм на платформі Node.js, розроблений командою розробників Express. Він надає більш елегантний та зручний синтаксис з використанням асинхронних функцій та генераторів, що робить його більш легким у використанні та підтримці асинхронного коду. За допомогою Koa в прєкті реалізується маршрутизація, обробка HTTP-запитів та відповідей, робота з параметрами, обробка помилок, автентифікація, авторизація [16].

Рисунок 2.5 – Koa v22

Для даного вебзастосунку прийнято рішення використовувати реляційну базу даних, тому що вона дає можливість розробки гнучких і добре масштабованих баз даних, зручну мову запитів, що дозволяє отримувати конкретні дані. Також для опису книги вікористовується багато характеристих, для опису яких доцільно використовувати окремі таблиці, що суттєво прискорить і полегшить процес розробки [17]. В якості СКБД використовується відкрита реляційна СКБД MySQL (рис. 2.6), тому що вона підтримує велику кількість даних, має високу продуктивність.

Рисунок 2.6 – MySQL

Також для спрощення процесу проєктування БД використовується MySQL Workbench, яке дозволило наочно моделювати структуру даних, встановлювати зв’язки між таблицями та генерувати SQL-скрипти для подальшої реалізації. Застосування цього інструменту значно підвищило ефективність роботи над архітектурою застосунку [18].

2.2 Специфікація вимог до програмного забезпечення

1. Призначення та межі проєкту

1.1 Призначення системи:

Вебзастосунок BookVa призначений для надання користувачам можливості швидко та зручно замовляти книги через Інтернет. Система дозволяє переглядати каталог літератури, здійснювати пошук за категоріями, безпечно проводити оплату та доставляти бажану книгу прямо додому. Користувачі можуть створювати особисті кабінети для перегляду історії покупок, відстеження статусу доставки друкованих видань і персоналізації рекомендацій. Головна мета – забезпечити швидкий доступ до книг для широкого кола читачів, підвищити зручність покупок та підтримати ефективне адміністрування книжкової торгівлі.

1.2 Погодження та стандарти:

- підтримка української мови та англійської мови інтерфейсу;
- інтеграція модернізованих вебтехнологій для забезпечення кросплатформенності;
- відповідність регламенту GDPR;
- відповідність стандарту PCI DSS.

1.3 Межі проєкту:

- відсутність розробки мобільного застосунку;
- розробка клієнтської та серверної частини вебзастосунку;
- підключення сторонніх платіжних систем;
- підключення сторонніх сервісів для забезпечення доставки.

2. Загальний опис

2.1 Сфера застосування:

Застосунок призначений для купівлі книг фізичними особами(покупцями). Система орієнтована на використання у браузерах на ПК та мобільних пристроях, не вимагає встановлення на локальні машини. Застосунок інтегрується з платіжними сервісами для обробки транзакцій та може бути пов’язаний з зовнішніми системами постачання

2.2 Характеристики користувачів:

- покупці, що хочуть придбати друковані книги;
- розробники зі знаннями у сфері розробки вебзастосунків;
- адміністратори, відповідальні за керування контентом і обробку замовлень.

2.3 Загальна структура і склад системи:

- клієнтська частина: вебінтерфейс, побудований за допомогою React та фреймворку Tailwind CSS;
- серверна частина: сервер, побудований на Node.js та Koa.js із базою даних MySQL;
- інтеграція: підключення до платіжних шлюзів, підключення сторонніх сервісів для здійснення доставки.

2.4 Загальні обмеження:

- наявність стабільного інтернет-з’єднання;
- робота лише з браузерами останніх версій;
- обмеження на обробку транзакцій у відповідності до банківських регуляцій;
- обмеження навантаження залежить від конфігурації серверної інфраструктури;
- обмеження на обробку персональних даних відповідно до чинного законодавства про захист даних GDPR.

3. Функції системи

3.1 Опис функції «Особистий кабінет та автентифікація»:

3.1.1 Вхідна інформація:

- облікові дані користувача (email і пароль);
- для адміністратора — код двофакторної верифікації (2FA).

3.1.2 Вихідна інформація:

- токен доступу JWT;
- доступ до персонального кабінету.

3.1.3 Функціональні вимоги:

Двофакторна верифікація для адміністраторів**.** Шифрування паролів.

3.2 Опис функції «Пошук за назвою»:

3.2.1 Вхідна інформація:

- дані з бази книг (назва, автор, жанр, ціна, мова, рік видання тощо);
- ключові слова, введені користувачем у пошуковий рядок;
- обрані параметри фільтрації та сортування.

3.2.2 Вихідна інформація:

- відфільтрований/відсортований список книг;
- результати пошуку, відповідні запиту;
- детальна інформація про книгу при натисканні.

3.3.3 Функціональні вимоги:

Перевірка на наявність шуканої книги/автора у базі даних. Сортування книг. Фільтрація книг.

3.3 Опис функції «Кошик і замовлення»:

3.3.1 Вхідна інформація:

- список товарів, обраних користувачем;
- дані користувача (адреса доставки, контактні дані);
- обраний спосіб доставки та оплати.

3.3.2 Вихідна інформація:

- підтвердження створення замовлення;
- номер замовлення та його статус;
- повідомлення на email про успішне оформлення.

3.3.3 Функціональні вимоги:

Перевірка на наявність товару. Перевірка на правильність адреси.

3.4 Опис функції «Платіжна система»:

3.4.1 Вхідна інформація:

- дані замовлення (ID замовлення, сума до сплати);
- обраний метод оплати (картка, Google Pay тощо);
- дані, передані до платіжного шлюзу через API.

3.4.2 Вихідна інформація:

- підтвердження успішної або неуспішної транзакції;
- оновлений статус замовлення в базі даних;
- повідомлення користувачу про результат оплати.

3.4.3 Функціональні вимоги:

Перевірка на достатність коштів на балансі. Підтвердження транзакції за допомогою 2FA.

3.5 Опис функції «Аналітика і звітність»:

3.5.1 Вхідна інформація:

- дані з бази замовлень, профілів користувачів, переглядів товарів;
- період, категорії або параметри для фільтрації (дата, жанр, автор, статус замовлення).

3.5.2 Вихідна інформація:

- візуалізація статистичних показників (графіки, таблиці) ;
- завантажувані звіти (CSV, PDF) ;
- дашборд з ключовими метриками.

3.5.3 Функціональні вимоги:

Дозвіл експорту звітів у форматах CSV та PDF. Можливість фільтрації статистики за категоріями книг, авторами, жанрами, статусами замовлень. Забезпечення оновлення даних в реальному часі.

4. Вимоги до інформаційного забезпечення

4.1 Вимоги до способів організації, збереження та ведення інформації:

– зберігання даних в реляційній базі даних (MySQL);
– постійне резервне копіювання даних (бажано щоденно);
– шифрування конфіденційних даних.

4.2 Технічні вимоги:

– серверна частина: сервери з підтримкою масштабування AWS/ Railway;
– клієнтська частина: підтримка останніх версій браузерів.
– мережа: стабільне інтернет-з'єднання з підтримкою HTTPS.

4.3 Властивості програмного забезпечення:

– оптимізація запитів до бази даних (індекси, JOIN-операції);

– асинхронна обробка запитів (async/await);

– можливість горизонтального масштабування (через кластеризацію або мікросервіси).

4.4 Сумісність:

– підтримка сучасних вебстандартів(HTML5, CSS3, ECMAScript 6+);

– сумісність з основними браузерами останніх версій;

– наявність відкритого REST API для потенційної інтеграції з іншими сервісами.

5. Вимоги до програмного забезпечення

5.1 Мова і технологія розробки:

– frontend: React, JavaScript, Tailwind CSS;

– backend: Node.js, Koa;

– база даних: MySQL.

5.2 Мережне програмне забезпечення

протокол HTTPS із сертифікатом SSL/TLS;

підтримка WebSocket для оновлення даних у реальному часі.

6. Властивості програмного забезпечення

6.1 Доступність

Час роботи системи: 99.9%.

6.2 Супроводжуваність

– документація API;

– модульна архітектура коду.

6.3 Продуктивність

– час відгуку API не більше 5 секунд;

– обробка одного замовлення: до 7 секунд;

– підтримує до 1000 одночасних користувачів.

6.4 Безпека:

– використання HTTPS для всіх API-запитів;

– захист від SQL Injection, XSS, CSRF через middleware та ORM;

– аутентифікація через JWT.

Висновки до розділу 2

У другому розділі було Досліджено інструменти веброзробки, які можуть бути застосовані для роботи з вебзастосунком продажу книг «BookVa». На основі аналізу виявлено доцільність використання певних інструментів для забезпечення високої продуктивності системи.

Розроблено специфікацію вимог до програмного забезпечення, де розглянути функціональні та нефункціональні вимоги.

В результаті роботи отримано бачення архітектури, функцій і обмежень системи, що створює надійну основу для її подальшої реалізації.

3 ПРОЄКТУВАННЯ ВЕБЗАСТОСУНКУ

3.1 Визначення ролей користувачів

Для того, щоб розробити вебзастосунок продажу книг потрібно визначити ролі користувачів у системі:

- неавторизований користувач;
- авторизований користувач: покупець, адміністратор.

Неавторизований користувач повинен мати змогу пошуку книг, перегляду інформації про книги, читання відгуків.

Авторизований користувач включає в себе 2 ролі: покупець та адміністратор. Як покупець користувач повинен мати можливість додавати книги до кошика, купувати їх, залишати відгуки на книги та відстежувати замовлення. Усі результати дій аокупця повинні зберігатися у його профілі.

Адміністратор у цій системі повинен модерувати користувачів, включаючи блокування їх обліковий записів, у разі порушення ними правил користування магазином.

3.2 Аналіз варіантів використання

Аналіз варіантів використання – це один із ключових етапів моделювання програмних систем, який базується на побудові діаграм варіантів використання Цей метод дозволяє описати функціональні вимоги до системи через взаємодію користувачів (акторів) із системою в межах конкретних сценаріїв (варіантів використання) [19].

Аналіз варіантів використання дозволяє:

- сформувати зрозумілу для замовника картину системи – діаграми подаються у вигляді простих сценаріїв, що легко читаються не лише технічними фахівцями, а й користувачами;
- визначити межі системи – use case допомагає чітко окреслити, що входить до функціоналу системи, а що – ні;

- підтримати розробку та тестування – кожен варіант використання може стати основою для реалізації певної функції та написання тестів до неї;
- полегшити командну співпрацю – спільне обговорення варіантів використання сприяє кращому розумінню потреб користувачів і зменшує ризик помилок.

1) Короткий сценарій

Вхід до системи. Користувач натискає кнопку «Увійти». Користувач бачить форму для входу з полями «пароль» та «емейл». Користувач вводить емейл та пароль у форму. Користувач натискає кнопку увійти. Система шукає користувача за емейлом. Система знаходить користувача і перевіряє пароль. Усі дані введено коректно. Користувач отримує доступ до свого профілю.

Перегляд каталогу та покупка книг: після входу користувач переходить до каталогу книг. Користувач вибирає книгу, додає її до кошика, переходить до оформлення замовлення, вводить платіжні дані, адресу доставки та завершує покупку. Після успішної покупки користувач отримує в особисті повідомлення треквнговий номер замовлення.

Персоналізовані рекомендації: на основі історії покупок і переглядів система генерує персоналізовані рекомендації книг для користувача. Рекомендації можуть базуватися на жанрах, яким користувач надає перевагу, або на книгах, які він переглядав раніше.

2) Поверхневий сценарій

Вхід до системи. Користувач натискає кнопку «Увійти». Користувач бачить форму для входу з полями «пароль» та «емейл». Користувач вводить емейл та пароль у форму. Користувач натискає кнопку увійти. Система шукає користувача за емейлом. Система знаходить користувача і перевіряє пароль. Усі дані введено коректно. Користувач отримує доступ до свого профілю.

Альтернативні сценарії:

– користувач неправильно увів пароль. Система повідомляє, що пароль введено неправильно. Користувач не може отримати доступ до профілю;

– користувач не заповнив форму для входу. Система повідомляє, що поля не заповнені;

– користувач неправильно увів емейл. Система повідомляє, що такого користувача не існує. Користувач не може отримати доступ до профілю;

– виникає технічна помилка під час обробки запиту (наприклад, проблеми з інтернет-з'єднанням або сервером Google). Система видає повідомлення про технічний збій і просить спробувати пізніше.

Перегляд каталогу та покупка книг: після входу в систему користувач переходить до свого профілю та обирає «Книги» з головного меню. Він переглядає доступні книги, фільтруючи їх за жанром, автором або популярністю. Користувач вибирає книгу для перегляду детальної інформації (опис, рецензії, ціна) та натискає кнопку «Додати до кошика». Після цього він може перейти до оформлення замовлення, ввести платіжні дані та завершити покупку. Після успішної покупки система повідомляє про успішне завершення операції та надсилає користувачеві трекінговий номер замовлення в особисті повідомлення.

Альтернативні сценарії:

- користувач вибирає книгу, але через технічну проблему не може додати її до кошика. Система повідомляє про помилку;
- користувач переглядає книгу, але намагається зробити покупку, коли вона вже розпродана. Система повідомляє про відсутність товару та пропонує подати заявку на повідомлення, коли книга з'явиться в наявності;
- під час оформлення покупки виникає технічна проблема з платіжною системою. Користувач отримує повідомлення про помилку і пропозицію спробувати пізніше;
- користувач вводить неправильні платіжні дані. Система просить перевірити введену інформацію і надає можливість виправити помилки.

Персоналізовані рекомендації: після покупки користувач отримує персоналізовані рекомендації книг, засновані на його виборах, переглядах або попе-

редніх покупках. Рекомендації можуть включати схожі книги або новинки в обраних жанрах. Користувач може зберегти ці рекомендації для подальшого перегляду або одразу додати книги до свого кошика.

Альтернативні сценарії:

- система не може надати персоналізовані рекомендації через відсутність достатніх даних про користувача. Користувачу пропонується переглянути популярні книги чи новинки;
- користувач переглядає рекомендації, але бажає побачити більше варіантів. Система надає додаткові книги з аналогічного жанру чи категорії;
- користувач незадоволений отриманими рекомендаціями і хоче переглянути інші варіанти. Система пропонує інші варіанти на основі його попередніх вподобань.

3) Повна форма

Таблиця 3.1 – Повна форма сценарію «Вхід до системи»

Use case section	**Comment**
Use Case Name	Реєстрація у системі
Scope	Система
Level	Користувацький
Primary Actor	Користувач
Stakeholders and interests	**Користувач:** хоче отримати доступ до облікового запису для персоналізованого досвіду покупок, відстеження замовлень і збереження історії покупок.
	Власник магазину: хоче зібрати інформацію про користувачів для поліпшення взаємодії з клієнтами, надання персоналізованих пропозицій і реклами. **Адміністратор системи:** хоче забезпечити доступ користувачів до облікових записів, перевірку введених даних та безпеку облікових записів.
Preconditions	Користувач має обліковий запис. Користувач має намір увійти у наявний обліковий запис.

Кінець таблиці 3.1

Success guarantee	Користувач успішно входить до свого облікового запису та може використовувати всі функціональні можливості системи.
Main Success Scenario	Користувач заходить на сторінку входу. Користувач заповнює обов'язкові поля форми входу: електронна адреса, пароль. Користувач натискає кнопку «Увійти». Система перевіряє, чи існує акаунт з такою електронною адресою та паролем. Система оновлює сесію користувача. Користувач переходить на сторінку облікового запису.
Extensions	**Якщо** користувач вводить некоректні дані (наприклад, неправильний email), система показує повідомлення про помилку і повідомляє, що користувача не існує. **Якщо** електронна адреса вже зареєстрована в системі, але пароль не підходить система повідомляє про це та пропонує відновити пароль або перевірити правильність його написання. **Якщо** виникає технічна проблема, система повідомляє про помилку й пропонує спробувати увійти пізніше.
Special Requirements	Валідація введених даних на стороні клієнта і сервера. Забезпечення конфіденційності та безпеки даних користувачів. Надання користувачу можливості відновлення пароля через електронну пошту.
Technology and Data Variations List	Інтеграція з email-сервісами для надсилання повідомлень про скидання паролю.
Frequency of Occurrence	Висока: кожен користувач, який бажає увійти в систему, проходить через цей процес.
Miscellaneous	Прогнозований час на вхід: 1-2 хвилини. Підтримка двофакторної автентифікації для забезпечення додаткової безпеки облікових записів.

Таблиця 3.2 – Повна форма сценарію «Придбання книги»

Use case section	**Comment**
Use Case Name	Придбання книги
Scope	Система
Level	Користувацький
Primary Actor	Користувач

Продовження таблиці 3.2

Stakeholders and interests	Клієнт: хоче отримати зручний процес перегляду, вибору та покупки книг. Власник магазину: хоче продавати книги та забезпечити зручну та надійну транзакцію для клієнтів. Платіжний процесор: зацікавлений у безпечних фінансових транзакціях. Служба доставки: зацікавлена в оперативному виконанні замовлення. Адміністратор системи: забезпечує правильну роботу платформи.
Preconditions	Користувач авторизований або вибрав опцію покупки як гість.
Success guarantee	Користувач отримує підтвердження успішної покупки. Платіж оброблений, книга відзначена як продана. Користувач отримує повідомлення з трекінговим номером замовлення.
Main Success Scenario	Користувач входить в систему або купує як гість. Користувач обирає книгу для покупки. Користувач додає книгу в кошик. Система перевіряє, що кошик не порожній та користувач авторизований. Користувач переходить до оформлення замовлення. Система відображає форму введення даних для доставки. Користувач заповнює обов'язкові поля форми та підтверджує замовлення. Система зберігає замовлення та показує сторінку підтвердження з номером замовлення. Користувач підтверджує замовлення. Користувач отримує підтвердження замовлення з деталями доставки. Система оновлює інвентар та відображає статус продажу.
Extensions	**Якщо** користувач вибирає книгу, але через технічну проблему не може додати її до кошика, система повідомляє про помилку. **Якщо** користувач переглядає книгу, але намагається зробити покупку, коли вона вже розпродана., система повідомляє про відсутність товару та пропонує подати заявку на повідомлення, коли книга з’явиться в наявності. **Якщо** під час оформлення покупки виникає технічна проблема з платіжною системою, система не отримує підтвердження оплати від платіжного

Кінець таблиці 3.2

	шлюзу протягом 30 с., Система відображає, система скасовує оформлення замовлення (повернення до стану кошика). **Якщо** користувач вводить неправильні платіжні дані, система просить перевірити введену інформацію і надає можливість виправити помилки.
Special Requirements	Забезпечення захищеного проведення платежів. Система повинна взаємодіяти з надійними платіжними системами.
Technology and Data Variations List	Інтеграція платіжних систем
Frequency of Occurrence	Дуже часто: кожен користувач, який захоче купити книгу буде проходити через цю операцію.
Miscellaneous	Прогнозована швидкість – 3-5 хвилин. Підтримка купонів/промокодів, які впливають на суму платежу, Валідація податків та локальних зборів

Таблиця 3.3 – Повна форма сценарію «Персоналізовані рекомендації»

Use case section	**Comment**
Use Case Name	Отримання персоналізованих рекомендацій
Scope	Система
Level	Користувацький
Primary Actor	Користувач
Stakeholders and interests	**Клієнт**: хоче отримати персоналізовані рекомендації, які відповідають його інтересам. **Власник магазину:** хоче збільшити кількість продажів та покращити взаємодію з користувачами. **Адміністратор системи**: відповідає за правильну роботу системи рекомендацій та її інтеграцію з базою даних та історією покупок.
Preconditions	Користувач авторизований та вже взаємодіяв із системою. Система зібрала достатню кількість інформацію про інтереси користувача.
Success guarantee	Користувач отримує персоналізовані рекомендації, які відповідають його інтересам і збільшують ймовірність здійснення покупки.

Кінець таблиці 3.3

Main Success Scenario	Користувач входить в акаунт або заходить на сайт. Система аналізує активність користувача, зокрема: попередні покупки, переглянуті книги, оцінки, залишені користувачем, вибрані категорії чи жанри книг. Система генерує список персоналізованих рекомендацій на основі зібраних даних. Користувач бачить на головній сторінці або в окремому розділі списки рекомендованих книг, що відповідають його інтересам. Користувач переглядає рекомендовані книги та може додати їх до кошика або придбати. Користувач має можливість додавати книги до списку що допомагає системі вдосконалювати рекомендації. Якщо користувач робить покупку, система використовує цю інформацію для покращення рекомендацій в майбутньому.
Extensions	**Якщо** система не має достатньо даних для формування персоналізованих рекомендацій (наприклад, новий користувач або незареєстрований гість), система пропонує базові рекомендації на основі популярних книг. **Якщо** користувач часто ігнорує певний тип рекомендацій, система може скорегувати алгоритм і почати пропонувати інші категорії або жанри..
Special Requirements	Система повинна бути здатною адаптуватися до змін в інтересах користувача. Висока точність алгоритмів для персоналізації рекомендацій
Technology and Data Variations List	Інтеграція з базами даних книг для автоматичного оновлення рекомендацій.
Frequency of Occurrence	Дуже часто: система персоналізує рекомендації для кожного користувача який має обліковий запис.
Miscellaneous	Прогнозована швидкість 5 секунд.

3.3 Побудова діаграми варіантів використання системи

Діаграма варіантів використання – це тип діаграми в UML, який використовується для моделювання функціональності системи з точки зору користувача. Вона показує, які дії може виконувати користувач (актор) у системі, і як система реагує на ці дії.

Вони використовуються для:

- зрозумілого описання функціональних вимог до системи;

- аналізу взаємодії користувача з системою;
- на етапі проєктування ПЗ, коли потрібно визначити основні сценарії використання.

На діаграмі варіантів використання відображено функціональні можливості системи онлайн-магазину книг для двох основних ролей: клієнта та адміністратора. Клієнт має змогу зареєструватися в системі, вводячи електронну пошту, номер телефону, створюючи профіль або використовуючи соціальні мережі. Після авторизації клієнт може редагувати свої дані, шукати книги за назвою, автором або характеристиками, обирати книги та оформлювати замовлення з вибором способу оплати. Також клієнт має доступ до особистого кабінету, де можна переглядати історію замовлень і список бажаних товарів.

Адміністратор, крім авторизації, має доступ до розширених можливостей: керування каталогом книг (додавання, редагування, видалення), авторами, акціями та замовленнями. Він може переглядати нові замовлення, змінювати їх статус і отримувати детальну інформацію. Уся функціональність структурована за допомогою зв’язків «include» (включає) та «extend» (розширює), що дозволяє гнучко моделювати обов’язкові та додаткові дії користувачів у системі.

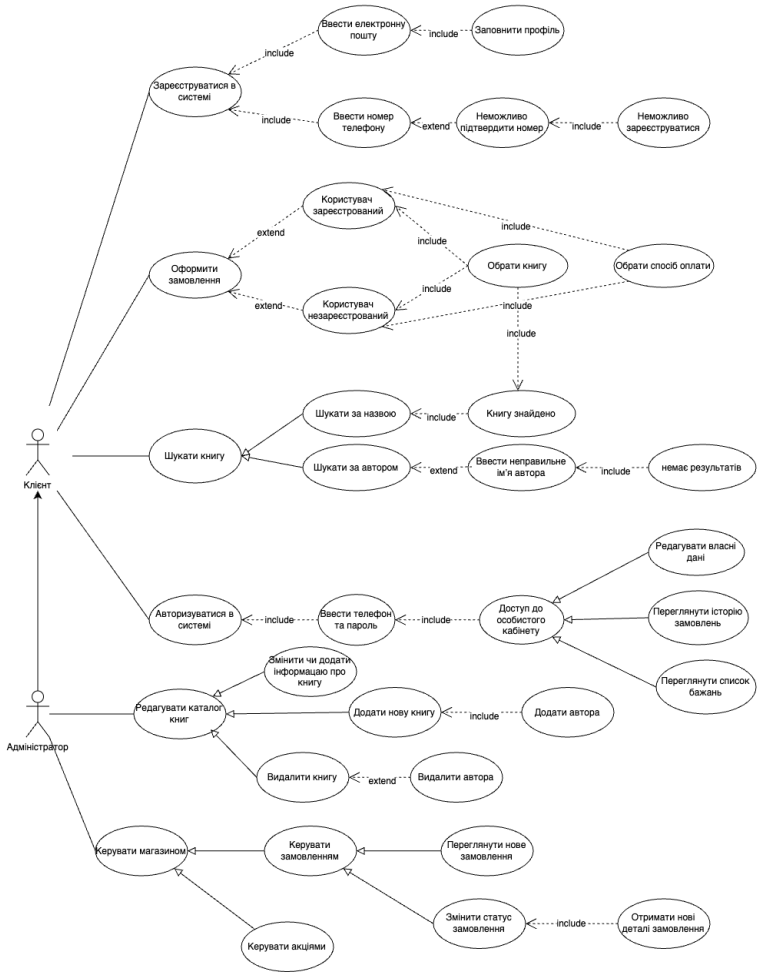

Рисунок 3.1 – Діаграма варіантів використання

Це один із перших і найбільш загальних кроків у моделюванні системи, який допомагає всім учасникам проєкту (замовникам, аналітикам, розробникам) мати спільне бачення того, як має працювати система.

3.4 Побудова діаграм послідовностей

Діаграма послідовності (Sequence Diagram) є одним із базових інструментів моделювання в уніфікованій мові моделювання UML, що широко застосовується для візуалізації динамічної поведінки об’єктно-орієнтованих систем. Вона належить до категорії інтеракційних діаграм і демонструє, як об’єкти або компоненти системи взаємодіють між собою у певному часовому порядку в межах одного сценарію використання (use case).

Основною метою побудови діаграм послідовності є формалізація та візуалізація логіки взаємодії між об’єктами або підсистемами під час виконання

конкретної функції або бізнес-процесу. Це дозволяє розробникам, архітекторам програмного забезпечення та аналітикам:

- краще зрозуміти поведінку системи з точки зору часу;
- зафіксувати порядок викликів методів між об’єктами;
- виявити залежності та критичні місця у взаємодії компонентів;
- спростити комунікацію між технічними та нетехнічними учасниками команди.

Діаграми послідовності ефективно використовуються на етапах проєктування та аналізу, особливо у випадках, коли система має складну внутрішню логіку або реалізує взаємодію з кількома зовнішніми сервісами.

Діаграма послідовності будується у вигляді впорядкованого за часом набору повідомлень, що обмінюються між учасниками (об'єктами, модулями або акторами). Основними елементами діаграми є актор або об’єкт (учасник взаємодії) – зображується у вигляді прямокутника вгорі діаграми. Життєва лінія (Lifeline) – вертикальна пунктирна лінія, що вказує на період існування об’єкта протягом сценарію. Повідомлення (Messages) – горизонтальні стрілки між лініями життя, які позначають виклики методів, передачу даних або сигнали.

Повідомлення можуть бути:

- синхронними (стрілка з суцільним наконечником) – вимагає очікування відповіді;
- асинхронними (стрілка з відкритим наконечником) – відправник не чекає відповіді.

Активність (Activation box) – прямокутник на життєвій лінії, який вказує на виконання об’єктом певної операції.

Уся діаграма будується зверху вниз відповідно до часової шкали, що дозволяє наочно представити послідовність дій.

Побудова діаграми послідовності має відповідати кільком ключовим принципам:

- **єдність сценарію:** одна діаграма повинна описувати одну логічну послідовність дій у межах певного сценарію використання.

- **точність і послідовність:** усі об’єкти, що беруть участь у взаємодії, повинні бути визначені, а повідомлення – мати зрозумілі назви, які відповідають функціям у системі.
- **часова впорядкованість:** кожне наступне повідомлення повинно логічно витікати з попередніх, забезпечуючи коректне уявлення про хронологію процесів.
- **інформативність:** діаграма має бути достатньо деталізованою, аби зрозуміти логіку реалізації функціоналу, але водночас не перевантаженою другорядними деталями.

На рисунку 3.2 зображена діаграма, що описує процес пошуку книги за назвою. Користувач бачить рядок пошуку й вводить назву книги. Вебінтерфейс надсилає запит із назвою до системи, яка у свою чергу звертається до бази даних. База повертає список книг, які відповідають запиту. Користувач натискає на обрану книгу – вебінтерфейс надсилає запит на детальну інформацію про неї. Система знову звертається до бази даних, отримує деталі і відображає їх користувачу. Цей сценарій ілюструє типовий шлях взаємодії при пошуку й перегляді книги.

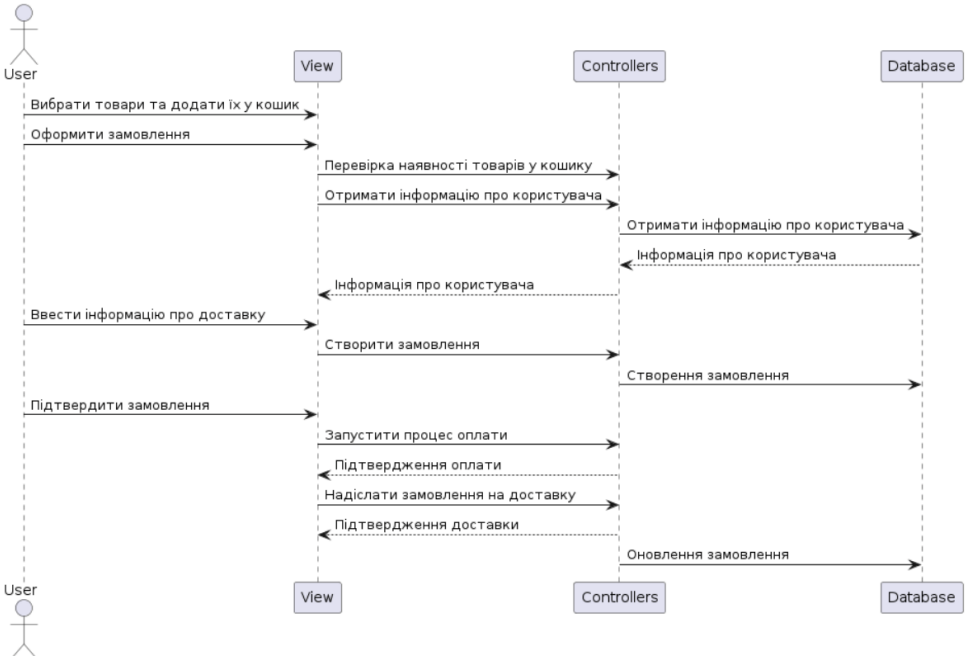

Рисунок 3.2 – Діаграма послідовной для процесу покупки книги

На рисунку 3.3 зображена друга діаграма (авторизація адміністратора). Ця діаграма описує сценарій, коли адміністратор забув пароль і проходить процес авторизації. Адміністратор натискає кнопку "Увійти", після чого обирає "Забув пароль". Система перевіряє дані для входу, надсилає перевірочні запитання. Після правильних відповідей перевіряється доступ через базу даних. Якщо все правильно, система надсилає лист для зміни пароля. Після створення нового пароля формується сеанс доступу, і користувача пускають до внутрішніх сторінок. Вхід завершується повідомленням про успіх, або, в разі помилки, повертається повідомлення про некоректні дані.

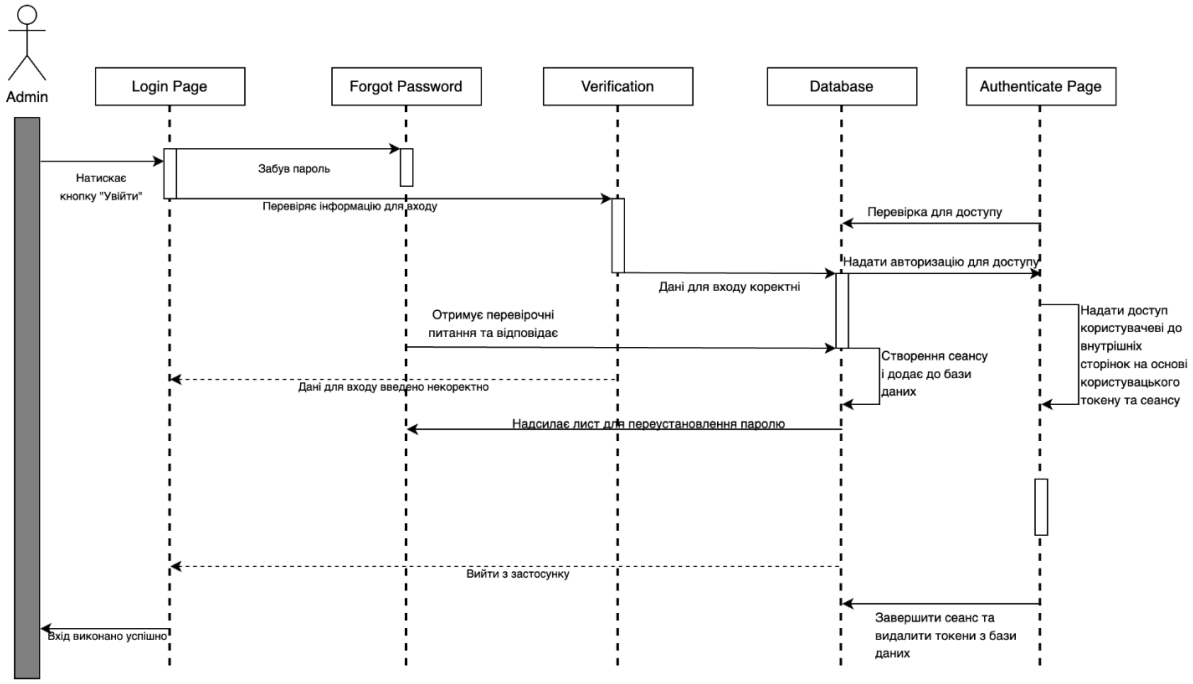

Рисунок 3.3 – Діаграма послідовностей для процесу авторизації адміністратора

На рисунку 3.3 зображена діаграма послідовностей для процесу пошуку книги. Спочатку користувач вводить назву книги, після чого вебінтерфейс показує рядок для пошуку та надсилає запит до системи. Система звертається до бази даних, яка повертає список відповідних збігів. Ці результати відображаються користувачу через вебінтерфейс. Далі користувач натискає на назву однієї з книг, і вебінтерфейс надсилає новий запит на отримання детальної інформації. Система знову звертається до бази даних, отримує розширені дані про вибрану книгу та передає

їх назад через вебінтерфейс для відображення користувачу. Уся послідовність демонструє логіку пошуку та перегляду даних у системі каталогізації книг.

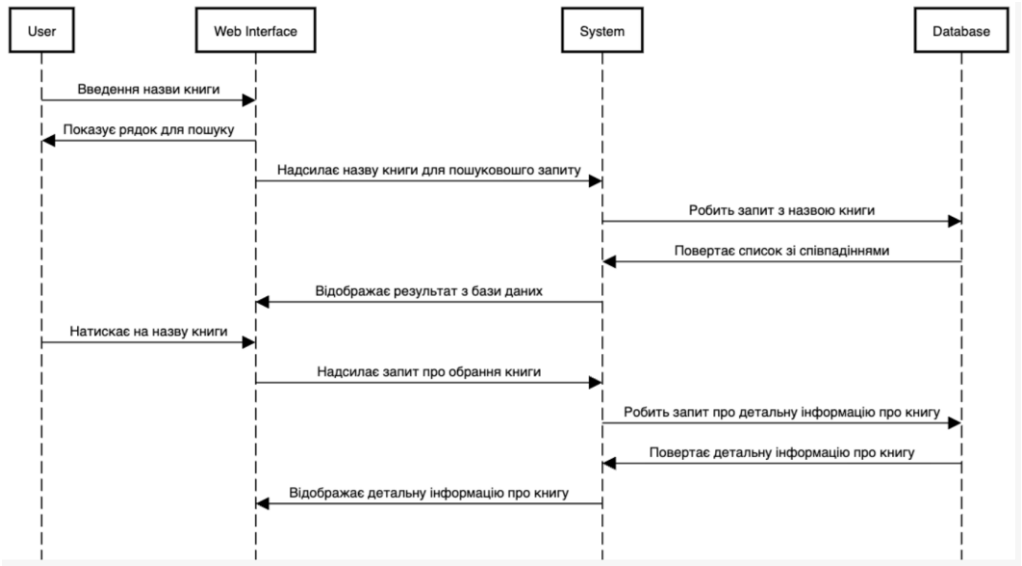

Рисунок 3.4 - Діаграма послідовностей для процесу пошуку книги

Побудована діаграма дозволяє чітко зрозуміти послідовність взаємодій між користувачем, вебінтерфейсом, системою та базою даних під час пошуку та перегляду книги. Вона допомагає виявити логіку роботи системи, спростити комунікацію між розробниками, аналітиками та замовниками, а також полегшує виявлення можливих помилок чи покраще

3.5 Побудова діаграм розгортання

Діаграма розгортання – це тип діаграми UML, яка показує фізичне розміщення програмного забезпечення на апаратних ресурсах системи. Іншими словами, вона відображає, як компоненти системи (наприклад, бази даних, сервери, клієнтські додатки) розташовані у реальному середовищі, яким чином між ними встановлюється зв’язок і як розподіляються функції. На діаграмі зображуються вузли (сервери, пристрої), програмні артефакти (компоненти, сервіси), а також зв’язки між ними у процесі.

Користь діаграми розгортання:

- візуалізує інфраструктуру системи;

- допомагає планувати розміщення компонентів у хмарі або на фізичних серверах;
- полегшує комунікацію між розробниками, DevOps та адміністраторами;
- сприяє виявленню потенційних проблем з продуктивністю або безпекою.

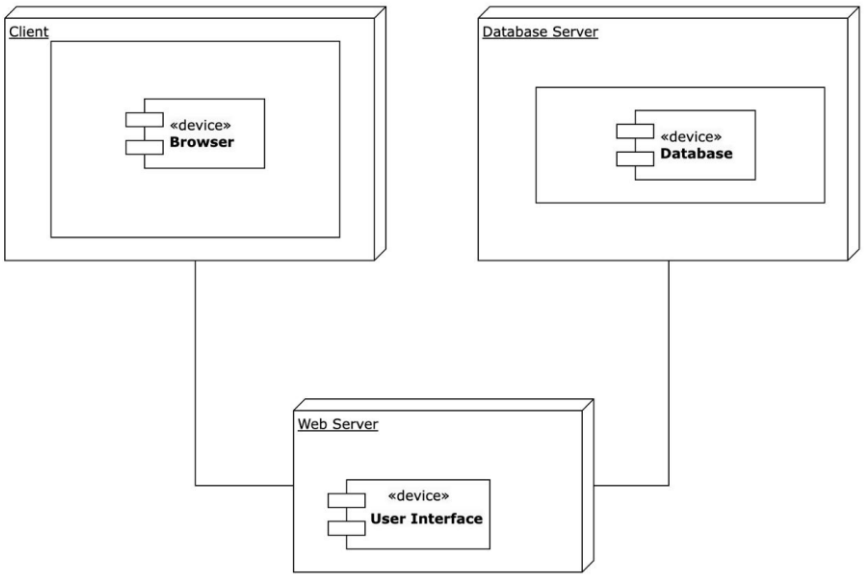

Рисунок 3.5 – Діаграма розгортання

За допомогою програмного забезпечення, встановленого на своєму пристрої, користувач може використовувати застосунок для купівлі та продажу книг. Здійснюючи взаємодію з інтерфейсом програми, користувач надсилає запити до централізованої системи, яка потім повертає потрібну інформацію. Ця система відповідає за управління всіма операціями, пов'язаними з відображенням даних та інтерфейсу. Інформація про книги, покупки, користувачів та інші аспекти системи зберігається в централізованій базі даних, що гарантує постійний доступ до оновлених даних для безперебійної функції всієї системи.

3.6 Побудова діаграм діяльності

Діаграма діяльності є одним із типів UML-діаграм, що використовується для моделювання динамічних аспектів систем. Вона відображає послідовність дій,

умовні переходи та паралельні процеси в межах конкретного бізнес-процесу або функціонального сценарію. Основна мета діаграми – продемонструвати, як користувач або система виконує певну задачу від початку до завершення.

Типові елементи діаграми діяльності включають: дії (активності), рішення (умовні вузли), початкові й кінцеві точки, переходи, а також паралельні гілки. Ці елементи дозволяють моделювати як прості послідовності, так і складні логічні потоки з умовами та повтореннями.

Загалом, діаграма діяльності – це інструмент, який спрощує розуміння складних процесів і є незамінним у фазах проєктування, аналізу та документації програмного забезпечення.

На рисунку 3.6 зображено діаграму, що показує, як користувач взаємодіє з застосунком для купівлі книг. Процес починається з відкриття сторінки програми. Далі користувач або шукає певну книгу, або переглядає каталог. Якщо книга знайдена, користувач переглядає її деталі й за бажанням додає до кошика. Після цього він може або продовжити покупки, або перейти до перегляду кошика. У кошику можна редагувати вміст, а коли все готово – перейти до перевірки й оформити замовлення. Процес містить циклічні етапи, де користувач щоразу вирішує, чи купувати ще.

Рисунок 3.6 – Діаграма діяльності купівлі книги

Третя діаграма описує процес входу до облікового запису та можливість відновлення пароля. Спочатку користувач натискає кнопку «Увійти», вводить електронну адресу і пароль. Система перевіряє, чи існує такий користувач, і якщо все вірно – надає доступ до акаунту. Якщо користувач забув пароль, він може скористатися кнопкою відновлення: система надсилає код на пошту, який потрібно ввести. Якщо код вірний – користувач вводить новий пароль і підтверджує його. У випадку помилки можна повторно надіслати код. Ця діаграма показує як основний сценарій входу, так і альтернативний – через відновлення доступу.

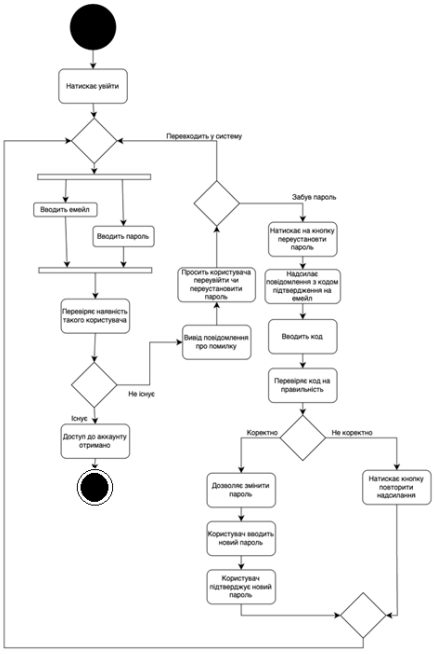

Рисунок 3.7 – Діаграма діяльності входу у систему

Діаграми активності допомагають візуально відобразити послідовність дій, що виконуються в межах процесу або сценарію. Вони полегшують розуміння логіки роботи системи, дозволяють виявити паралельні або альтернативні гілки виконання, а також чітко показують початок і завершення процесу. Такі діаграми особливо корисні для аналізу вимог, моделювання бізнес-процесів, виявлення помилок у логіці та ефективного планування функціональності програмного забезпечення.

Висновки до розділу 3

У третьому розділі було проведено детальний аналіз можливих сценаріїв використання вебзастосунку продажу книг, зокрема короткого, поверхневого та повного сценаріїв. Визначено потреби, ролі та очікувані результати взаємодії користувача з системою. На основі проведеного аналізу побудовано діаграму варіантів використання системи. Створено діаграму послідовностей для процесу покупки книги та авторизації адміністратора. Визначено поняття діаграми розгортання та побудовано відповідну діаграму. Створено діаграму діяльності для покупки книги та авторизації користувача.

В результаті роботи отримано бачення логіки виконання процесів у системі для кращого розуміння її поведінки та взаємодії з користувачем. Проведена робота створює надійну основу для її подальшої реалізації.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ

4.1 Розробка вебзастосунку

На бекенді застосовується Node.js у зв’язці з Koa, що забезпечує швидку та гнучку обробку HTTP-запитів. Через API реалізуються функції: реєстрація й авторизація користувачів (у тому числі з використанням JWT), отримання списку книг, пошук, додавання товарів до кошика, формування замовлень, а також панель адміністратора для керування книгами та перегляду замовлень. MySQL виступає в ролі бази даних, де зберігається інформація про книги (назва, автор, ціна, опис), користувачів, транзакції й кошики.

Таким чином, результатом є зручний та сучасний вебзастосунок, що дозволяє користувачам купувати книги онлайн, а адміністрації – ефективно керувати вмістом платформи. Такий проєкт може легко масштабуватися, доповнюватися новими функціями (наприклад, рецензії, електронні книги, система знижок) та інтегруватися з платіжними сервісами.

Для підключення до бази даних MySQL у середовищі Node.js використовується бібліотека mysql2/promise (рис 4.1) з використанням асинхронного підходу async/await. Це сучасний підхід, що дозволяє зручно працювати з базою даних без вкладених колбеків, які часто ускладнюють читання коду.

```
4   import mysql from 'mysql2/promise';
5
6   const pool = await mysql.createPool({
7     host: 'localhost',
8     user: 'root',
9     password: 'mufvak-tegcU9-xaxvig',
10    database: 'books',
11    port: 3306,
12    waitForConnections: true,
13    connectionLimit: 10,
14    queueLimit: 0,
15  });
16  try {
17    const connection = await pool.getConnection();
18    console.log('✅');
19    connection.release(); // важно!
20  } catch (err) {
21    console.error('❌', err.message);
22  }
23  export default pool;
24
```

Рисунок 4.1 – Підключення до бази даних

У сучасній розробці ПЗ архітектурні паттерни грають ключову роль, гарантують структуровані підходи до організації систем. Вони допомагають розробникам створювати додатки, що легко підтримуються, масштабуються та адаптуються до змін. Одним із найбільш популярних і ефективних паттернів є MVC, який має широке застосування як у веброзробці, так і у створення складних програмних систем.

MVC розділяє систему на три логічні компоненти: модель, представлення і контролер. Модель відповідає за бізнес-логіку та управління даними, зберігає стан і надає доступ до інформації. Представлення забезпечує UI і візуалізацію, а контролер оброблює події користувача та контролює взаємодію між моделлю і представленням.

Розділення цих компонентів відповідає принципу розділення обов’язків, що сприяє підвищенню гнучкісті системи і дозволяє розробникам одночасно працювати над різними частинами проєкту без виникнення конфліктів.

У рамках сучасних вебзастосунків та API архітектура MVC дозволяє забезпечувати обробку HTTP-запитів відповідно до CRUD-операцій (створення, читання, оновлення, видалення). Контролер відповідає за виконання маршрутизації та оброблення запитів, модель управляє даними, а представлення надає відповіді у вигляді JSON або інших форматів.

Цей шаблон забезпечує ефективну підтримку розподілених команд розробників, полегшує тестування та інтегрування нових функцій у вже існуючі системи.

Реалізація CRUD-операцій наведена у додатку А.

Щоб отримати список усіх авторів, які писали книги створюється пул з'єднань (pool) за допомогою функції mysql.createPool(). Це означає, що замість створення нового з'єднання кожного разу, коли потрібно виконати запит до бази, буде використовуватись пул вже відкритих з'єднань. Це значно підвищує продуктивність і масштабованість застосунку. Параметри пулу вказують:

- host, user, password, database, port – дані для підключення до локальної бази даних books.
- waitForConnections – чи слід чекати, якщо всі з'єднання зайняті;

- connectionLimit – максимальна кількість з'єднань одночасно;
- queueLimit – кількість запитів, які можуть очікувати в черзі (0 означає необмежену чергу).

Далі йде спроба отримати з'єднання з пулу через pool.getConnection(). Якщо це вдається, виводиться повідомлення про успішне підключення. Після цього з'єднання обов'язково повертається назад у пул за допомогою connection.release(). Це важливо для запобігання витоку з'єднань.

Якщо ж підключення не вдалося (наприклад, через неправильні дані чи відсутність сервера MySQL), то помилка ловиться в блоці catch, і виводиться повідомлення з текстом помилки.

Вкінці модуль експортує pool, щоб його можна було використовувати в інших частинах програми у маршрутах або сервісах для виконання SQL-запитів.

Список книг за іменем автора (рис. 4.2)

```
// @ts-check

/**
 * @typedef {import('@koa/router')} Router
 * @typedef {import('mysql2/promise').Pool} SQL
 */

/**
 * @param {SQL} sql
 * @param {Router} router
 */
export default (sql, router) => {
  router.get('booksByAuthorName/:name', async (ctx) => {
    const name = `%${decodeURIComponent(ctx.params.name)}%`;
    const [rows] = await sql.query(
      `SELECT books.id, title, year
       FROM books
       INNER JOIN authors ON books.authorId = authors.id
       WHERE authors.name LIKE ?`,
      [name]
    );
    ctx.body = { data: rows };
  });
};
```

Рисунок 4.2 – Отримання списку книг за іменем автора

Функція за замовчуванням експортується як middleware-модуль. Вона приймає два параметри: об'єкт sql (пул з'єднань до MySQL) і router (роутер для оголошення маршрутів). В межах цієї функції визначається маршрут типу GET з шляхом

booksByAuthorName/:name. Це динамічний маршрут, де :name – це частина URL, яка означає ім’я автора.

Коли надходить запит на цей маршрут, сервер спочатку витягує параметр name з URL, декодує його і додає символи для використання в операторі LIKE, що дозволяє шукати авторів за частковим збігом імені. Далі виконується SQL-запит, який об'єднує таблиці books і authors за авторським ідентифікатором та повертає ті книжки, де ім’я автора співпадає з переданим фрагментом.

Результат запиту зберігається в змінній rows, яка потім повертається клієнту у форматі JSON. Це дозволяє фронтенду легко обробити отриману відповідь, наприклад – відобразити список книг, написаних автором з певним іменем.

У межах реалізації API у вебзастосунку, було створено маршрут (рис. 4.3), який реалізує пошук книг за частковим збігом імені автора з можливістю сортування результатів за різними параметрами. Цей маршрут є прикладом гнучкого та динамічного запиту, що відповідає принципам REST-архітектури.

Маршрут GET /booksWithAuthorByName/:name/:column?/:direct? приймає один обов’язковий і два необов’язкові параметри. Параметр name використовується для фільтрації імен авторів за частковим збігом, тобто запит до бази даних містить конструкцію LIKE, що дозволяє знайти усі записи, де ім’я автора містить заданий фрагмент. Додаткові параметри column і direct використовуються для сортування результатів запиту. Параметр column визначає, за яким стовпцем буде виконано сортування (наприклад, title, year, name), а direct визначає напрямок сортування (ASC – за зростанням, DESC – за спаданням).

Щоб запобігти SQL-ін’єкціям, у коді реалізовано перевірку допустимих значень для параметрів сортування. Перелік дозволених назв колонок (columns) і допустимих напрямків сортування (directs) перевіряється перед тим, як сформувати остаточний SQL-запит. У разі, якщо користувач вказав некоректне ім’я стовпця або напрямок сортування, застосовуються значення за замовчуванням (books.id та ASC відповідно).

Сформований SQL-запит реалізує з’єднання таблиць books та authors через внутрішній JOIN по полю authorId. У результаті цього запиту користувач отримує

список книг разом з інформацією про авторів (ім’я, стать, дата народження), які відповідають умовам фільтрації за іменем.

```
export default (sql, router) => {
  router.get('booksWithAuthorByName/:name/:column?/:direct?', async (ctx) => {
    const name = `%${decodeURIComponent(ctx.params.name)}%`;
    const directs = ['ASC', 'DESC'];
    const columns = [
      'books.id',
      'title',
      'year',
      'authorId',
      'name',
      'sex',
      'birth',
    ];
    const { column, direct } = ctx.params;
    const orderby = columns.includes(column) ? column : 'books.id';
    const asc = directs.includes((direct || '').toUpperCase()) ? direct.toUpperCase() : 'ASC';
    const query = `
      SELECT books.id, title, year, authorId, name, sex, birth
      FROM books
      INNER JOIN authors ON books.authorId = authors.id
      WHERE authors.name LIKE ?
      ORDER BY ${orderby} ${asc}
    `;
    const [rows] = await sql.query(query, [name]);
    ctx.body = { data: rows };
  });
};
```

Рисунок 4.3 – Реалізація пошуку і фільтрації книг

Результати запиту повертаються у вигляді JSON-об’єкта, де основні дані містяться у полі data. Такий підхід відповідає сучасним стандартам побудови RESTful сервісів та дозволяє клієнтському застосунку зручно обробляти отримані дані.

Загалом, цей маршрут є прикладом гнучкої реалізації пошуково-сортувального механізму в рамках API-доступу до реляційної бази даних. Він демонструє важливість попередньої валідації параметрів запиту, а також ефективного використання SQL-операторів JOIN, LIKE і ORDER BY.

4.2 Розробка модулю авторизації користувачів

JWT – це сучасний і легкий механізм передачі даних у шифрованому вигляді між клієнтом і сервером, який знайшов широке застосоування для аутентифікації та авторизації в вебзастосунках.

Процес роботи з JWT починається з того, що сервер після вдалої аутентифікації користувача створює токен, який має головну інформацію (ідентифікатор, роль, термін дії тощо) та підписується криптографічним ключем. Цей токен передається клієнту, який додає його до кожного запиту. Сервер перевіряє підпис, і при

позитивному результаті надає доступ до ресурсів [19]. Даний підхід дозволяє зменшити кількість зайвих звернень до бази даних для підтвердження прав доступу, що має позитивний вплив на продуктивність застосунку.

Також існують імовірні загрози використання JWT. Зокрема, вірогідність перехоплення токена при недостатньо надійному зберіганні або передаванні, труднощі з відкликанням токена до закінчення строку дії та ризики застосування ненадійних алгоритмів підпису. Використання HTTPS, вірне налаштування криптографії та зменшення часу дії токена мінімізує ризики повязані з безпекою [20].

Коли користувач надсилає запит про реєстрацію, сервер отримує ім’я користувача та пароль з тіла запиту (рис. 4.4). Пароль шифрується за допомогою bcrypt з «сіллю» (salt) рівня 10. Створюється новий користувач з ім’ям і зашифрованим паролем, і зберігається в базу даних. Якщо все проходить успішно – повертається статус 201 і повідомлення про успішну реєстрацію. Якщо виникає помилка –повертається статус 500 і повідомлення про невдалу реєстрацію.

```
router.post('/register', async (req, res) => {
try {
const { username, password } = req.body;
const hashedPassword = await bcrypt.hash(password, 10);
const user = new User({ username, password: hashedPassword });
await user.save();
res.status(201).json({ message: 'User registered successfully' });
} catch (error) {
res.status(500).json({ error: 'Registration failed' });
}
});
```

Рисунок 4.4 – Програмна реалізація реєстрації користувача

Коли користувач надсилає запит про авторизацію (рис. 4.5), сервер отримує логін та пароль. Шукається користувач у базі даних за іменем користувача. Якщо його не знайдено –повертається 401 (автентифікація не вдалася). Якщо користувач знайдений – перевіряється відповідність паролів (введений пароль і хешований у БД). Якщо паролі не збігаються – повертається 401. Якщо все гаразд – генерується JWT-токен з ID користувача. Термін дії токена 1 година. Токен повертається клієнту у відповіді з кодом 200. У випадку помилки повертається статус 500 та повідомлення про помилку входу.

```
router.post('/login', async (req, res) => {
try {
const { username, password } = req.body;
const user = await User.findOne({ username });
if (!user) {
return res.status(401).json({ error: 'Authentication failed' });
}
const passwordMatch = await bcrypt.compare(password, user.password);
if (!passwordMatch) {
return res.status(401).json({ error: 'Authentication failed' });
}
const token = jwt.sign({ userId: user._id }, 'your-secret-key', {
expiresIn: '1h',
});
res.status(200).json({ token });
} catch (error) {
res.status(500).json({ error: 'Login failed' });
}
});

module.exports = router;
```

Рисунок 4.5 – Програмна реалізація авторизації користувача

Отже, JWT є потужним інструментом для реалізації безпечної авторизації, який поєднує ефективність, масштабованість і зручність у використанні, проте вимагає уважного підходу до безпеки та реалізації.

4.3 Розробка користувацького інтерфейсу

На фронтенді застосовується React, що дозволяє реалізувати динамічний та зручний інтерфейс: каталог книг із фільтрацією, сторінку товару з детальним описом, особистий кабінет користувача, інтерактивний кошик і форму для оформлення замовлення. Стилізація виконується за допомогою TailwindCSS, що забезпечує адаптивний дизайн, легке налаштування компонентів і швидку побудову інтерфейсу.

Navbar.

У коді створено компонент Navbar це шапка сайту (рис. 4.6), яка відображається у верхній частині сторінки. Вона містить логотип магазину, назву "BOOKSHOP" та іконку кошика.

Компонент імпортує елементи з бібліотеки Material UI: верхню панель (AppBar), панель інструментів (Toolbar), кнопки (IconButton), значки (Badge, ShoppingCart) і текст (Typography). Також використовується компонент Link з react-router-dom для переходів між сторінками без перезавантаження.

```
const StyledAppBar = styled(AppBar)(({ theme }) => ({
  backgroundColor: "#fff",
  boxShadow: "none",
}));
const Title = styled(Typography)(({ theme }) => ({
  display: "flex",
  alignItems: "center",
  textDecoration: "none",
  color: "inherit",
  gap: theme.spacing(1),
}));
const Grow = styled("div")({
  flexGrow: 1,
});
const Navbar = ({ totalItems }) => {
return (
    <StyledAppBar position="fixed" color="inherit">
      <Toolbar>
        <Title variant="h5" component={Link} to="/">
          <img src={logo} alt="Book Store App" height="50px" />
          BOOKSHOP
        </Title>

        <Grow />

        <IconButton
          component={Link}
          to="/cart"
          aria-label="Show cart items"
          color="inherit"
        >
          <Badge badgeContent={totalItems} color="secondary">
            <ShoppingCart />
          </Badge>
        </IconButton>
      </Toolbar>
    </StyledAppBar>
  );
};

export default Navbar;
```

Рисунок 4.6 – Створення панелі навігації сайту

Логотип (circles.png) і текст "BOOKVA" є клікабельним посиланням на головну сторінку. З правого боку розміщується кнопка з іконкою кошика, яка веде на сторінку кошика (/cart). Біля іконки кошика показується кількість товарів, яка передається через змінну totalItems.

Для стилів використовується зовнішній файл styles.js, звідки імпортується useStyles.Компонент експортується для подальшого використання в інших частинах застосунку.

Cart.

React-компонент під назвою Cart реалізує відображення сторінки кошика в інтернет-магазині. Він використовує бібліотеки Material-UI для візуального оформлення інтерфейсу та React Router для навігації між сторінками. Компонент отримує дані про кошик через пропси: об'єкт cart, а також функції для оновлення кількості товарів, видалення товарів із кошика та очищення всього кошика.

Спочатку компонент перевіряє, чи є дані про товари (cart.line_items). Якщо таких даних ще немає, відображається повідомлення "Loading". Якщо кошик порожній, показується текст з посиланням на головну сторінку, де користувач може почати додавати товари (рис. 4.7). Якщо ж у кошику є товари, компонент створює сітку з елементами CartItem, які представляють окремі товари в кошику. Для кожного товару передаються функції керування: оновлення кількості, видалення тощо.

Під списком товарів відображається підсумок суми замовлення (subtotal), а також дві кнопки: одна для очищення кошика, інша – для переходу до оформлення замовлення (checkout). Весь компонент обгорнуто в контейнер для забезпечення адаптивного та акуратного макету. Такий підхід дозволяє зручно та функціонально відображати вміст кошика користувача у вебзастосунку.

```
const Cart = ({ cart, onUpdateCartQty, onRemoveFromCart, onEmptyCart }) => {
  const classes = useStyles();

  const handleEmptyCart = () => onEmptyCart();

  const renderEmptyCart = () => (
    <Typography variant="subtitle1">You have no items in your shopping cart,
      <Link className={classes.link} to="/"> start adding some</Link>!
    </Typography>
  );

  if (!cart.line_items) return 'Loading';

  const renderCart = () => (
    <>
      <Grid container spacing={4}>
        {cart.line_items.map((lineItem) => (
          <Grid item xs={12} sm={4} key={lineItem.id}>
            <CartItem item={lineItem} onUpdateCartQty={onUpdateCartQty} onRemoveFromCart={onRemoveFromCart} />
          </Grid>
        ))}
      </Grid>
      <div className={classes.cardDetails}>
      <Typography variant="h5" >Subtotal: <b >{cart.subtotal.formatted_with_symbol}</b></Typography>
        <div>
          <Button className={classes.emptyButton} size="large" type="button" variant="contained" color="secondary" onClick={handleEmptyCart}>Empty cart</Button>
          <Button className={classes.checkoutButton} component={Link} to="/checkout" size="large" type="button" variant="contained" >Checkout</Button>
        </div>
      </div>
    </>
  );

  return (
    <Container>
      <div className={classes.toolbar} />
      <Typography className={classes.title} variant="h5" gutterBottom><b>Your Shopping Cart</b></Typography>
      <hr/>
      { !cart.line_items.length ? renderEmptyCart() : renderCart() }
    </Container>
  );
};

export default Cart;
```

Рисунок 4.8 – Створення корзини сайту

ProfilePage.

Цей компонент UserProfile реалізує сторінку профілю користувача в React-додатку з використанням бібліотеки Material-UI для стилізації інтерфейсу. Основною метою цього компонента є відображення особистої інформації користувача, такої як ім’я, електронна пошта, дата реєстрації, а також надання можливості вийти з облікового запису за допомогою кнопки "Logout".

Компонент отримує два пропси: об'єкт user, що містить дані про користувача (ім’я, email, дата приєднання, аватар), і функцію onLogout, яка викликається при натисканні кнопки виходу. Інформація користувача виводиться всередині компонента Paper, який візуально виділяє блок на сторінці та додає тінь. За допомогою компонента Avatar відображається перша літера імені користувача або картинка (якщо передано URL аватара).

Також використовується сіткова система Material-UI (Grid) для структурування елементів у вертикальну колонку з відступами. Оформлення задається через зовнішній файл стилів styles.js, де визначено зовнішній вигляд аватара, відступи, розміри елементів тощо.

```
const UserProfile = ({ user, onLogout }) => {
  const classes = useStyles();

  return (
    <Container maxWidth="sm">
      <div className={classes.toolbar} />
      <Paper className={classes.profileContainer} elevation={3}>
        <Grid container spacing={2} alignItems="center" direction="column">
          <Grid item>
            <Avatar className={classes.avatar} alt={user.name} src={user.avatarUrl}>
              {user.name.charAt(0)}
            </Avatar>
          </Grid>
          <Grid item>
            <Typography variant="h5"><b>{user.name}</b></Typography>
            <Typography variant="body1">{user.email}</Typography>
            <Typography variant="body2" color="textSecondary">Member since: {user.memberSince}</Typography>
          </Grid>
          <Grid item>
            <Button variant="contained" color="secondary" onClick={onLogout}>
              Logout
            </Button>
          </Grid>
        </Grid>
      </Paper>
    </Container>
  );
};

export default UserProfile;
```

Рисунок 4.9 – Створення сторінки профілю користувача

Таким чином, цей код створює базову, але повноцінну сторінку профілю користувача, яка добре виглядає на будь-яких пристроях, легко розширюється і підтримує інтеграцію з іншими частинами застосунку, наприклад, системою аутентифікації або замовленнями.

4.4 Інструкція користувача

При першому переході у вебзастосунок з зовнішнього ресурсу користува потрапляє на головну сторінку, що представлена на рисунку 4.10.

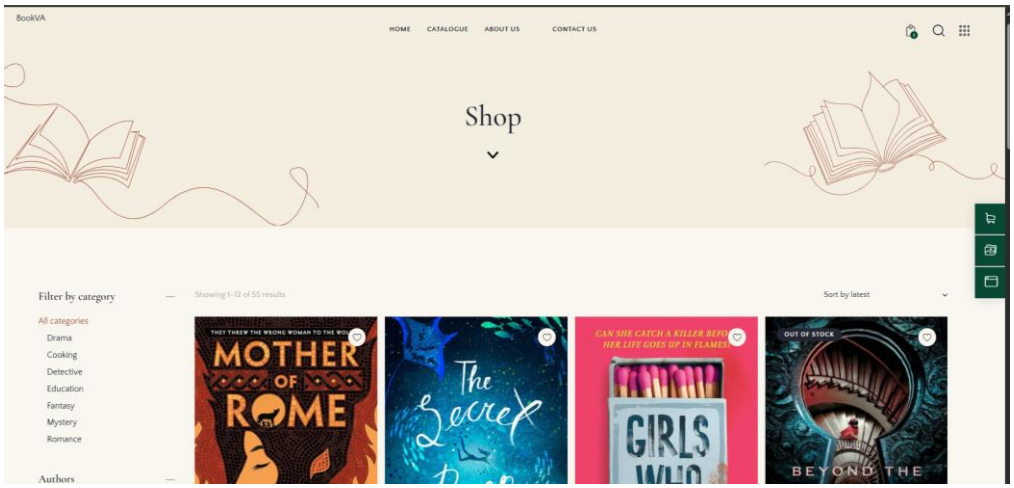

Рисунок 4.10 – Головна сторінка вебзастосунку

З цієї сторінки користувач може продовжувати переміщення по вебзастосунку за допомогою розташованої у шапці сайту навігаційної панелі. З головної сторінки користувач може перейти до каталогу книг (рис. 4.11) або ж до пошуку певної книги (рис. 4.13).

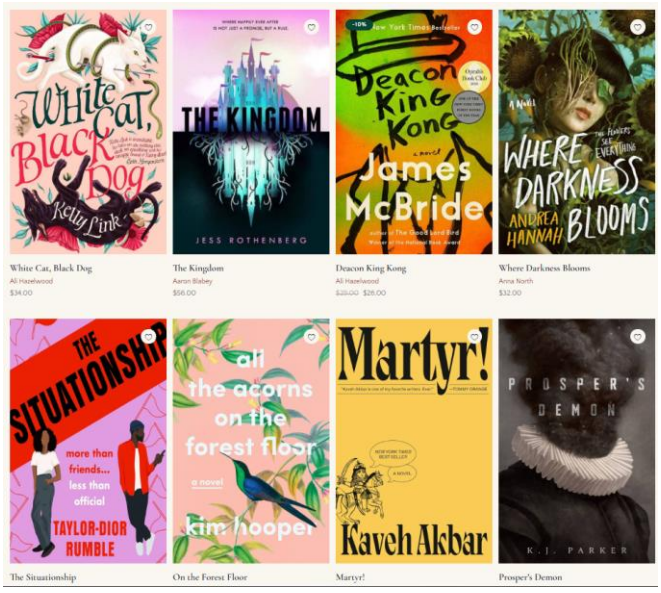

Рисунок 4.11 – Каталог книг

У випадку, якщо користувач хоче перейти до каталогу книг та не знаю яку саме книжку він хоче купити, він може скористатися можливістю фільтрувати

книги за категорією та автором (рис. 4.12). Користувачеві достатньо лише обратно жанр та автора, і ці книги буде відображено на сторінці.

Рисунок 4.12 – Панель для фільтрації книг за категорією та автором

У випадку, якщо користувач відвудує сайт з метою знайти певну книгу, то він може скористатися пошуком на сайті. Користувачеві достатньо ввести назву книги, або слова з назви які він пам’ятає і система покаже йому книги з такими назвами.

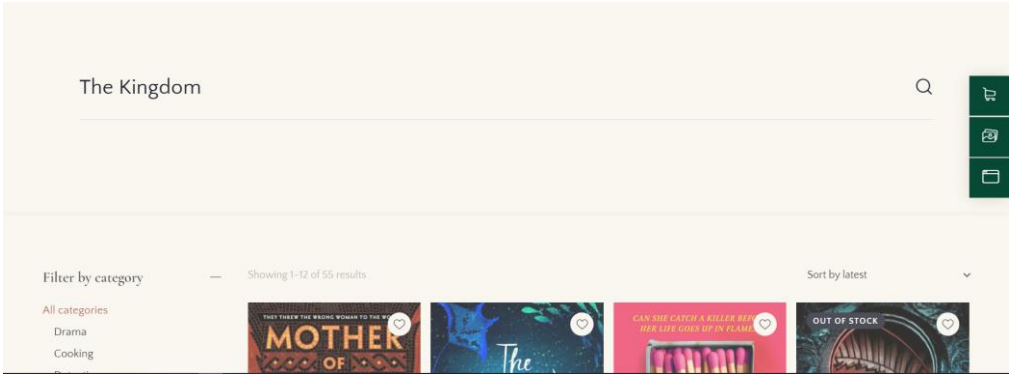

Рисунок 4.13 – Пошук книги за назвою

Якщо користувач знаходить за допомогою пошуку або у каталозі потрібну йому книгу, він може перейти на сторінку книги (рис. 4.14), щоб детальніше ознайомитися з інформацією про дану книгу.

Рисунок 4.14 – Сторінка товару

Після ознайомлення з товаром користувач може вирішити для себе чи хоче він купувати даний товар чи ні. У випадку, якщо користувачеві сподобалася книга, він може натиснути на кнопку «Додати у кошик» і відразу перейти до оформлення покупки (рис. 4.15). Першим кроком у корзині він побачить інформацію про товар, що він додав, а саменазву, ціну та кількість доданого товару, а також користувачеві буде надана можливість видали товар з кошику.

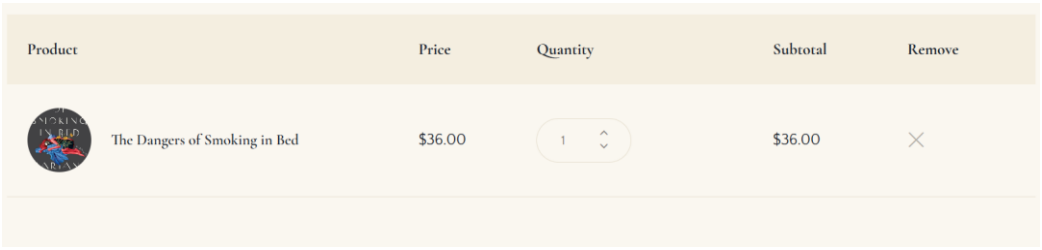

Рисунок 4.15 – Сторінка кошику

Після того, як користувач перевірив товар у кошику, він може перейти до оформлення замовлення, де йому потрібно буде вказати всю інформацію про себе, включаючи адресу, електронну пошту та номер телефону (рис. 4.16). Також стандартним способом оплати є оплата при отриманні.

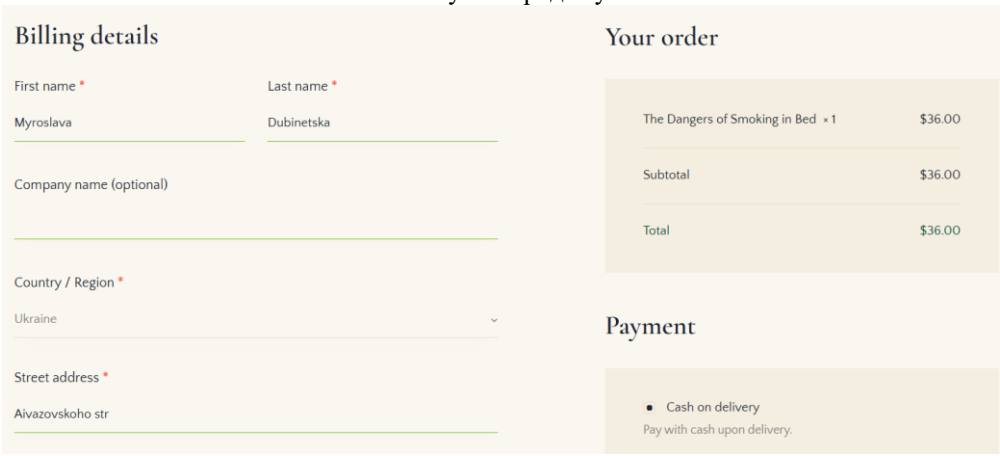

Рисунок 4.16 – Інформація про користувача

Інтерфейс інтуїтивно зрозумілий, забезпечує швидкий доступ до каталогу товарів, дозволяє додавати книги до кошика, переглядати деталі замовлення та керувати профілем користувача. В результаті, система забезпечує ефективний і приємний користувацький досвід для читачів будь-якого рівня.

4.5 Тестування системи

Тестування системи це важливий завершальний етап розробки вебзастосунку, під час якого перевіряється, наскільки коректно працюють усі його функціональні компоненти.

Тестування вручну (manual testing) це процес перевірки програмного забезпечення без використання автоматизованих інструментів. У цьому підході тестувальник виконує тестові сценарії вручну, щоб виявити помилки, перевірити функціональність, зручність інтерфейсу, відповідність вимогам та загальну якість програмного продукту.

Таблиця 4.1 – Тестування пошуку книги за назвою

Діючі особи	Користувач
Мета	Знайти книгу за назвою
Передумова	Користувач має перебувати на сторінці пошуку
Успішний сценарій: 1) Користувач заходить на сторінку пошуку. 2) Користувач вводить назву книги у графі пошуку. 3) Користувач натиснув кнопку «Знайти». 4) Система виводить книгу, яку шукав користувач. 5) Користувач може переглянути детальну інформацію про книгу.	

Кінець таблиці 4.1

Сценарій успішний. Користувач знайшов шукану книгу.	
Розширення:	
1a	Користувач ввів неправильну назву книги. Система не може знайти збігів з книгами, що є у базі даних. Система повідомляє, що такої книги не існує. Користувач не може знайти шукану книгу.
Усі сценарії розширення успішно виконані.	

Таблиця 4.2 – Тестування внесення змін у власному профілі

Діючі особи	Користувач
Мета	Внести зміни до даних у профілі
Передумова	Користувач має бути зареєстрованим у системі
Успішний сценарій: 1) Користувач заходить на головну сторінку. 2) Користувач натискає кнопку «Увійти». 3) Користувач бачить форму для входу. 4) Користувач вводить електронну пошту та пароль. 5) Користувач отримує доступ до свого аккаунту. 6) Користувач натискає кнопку «Змінити». 7) Користувач обирає поле, яке він хоче змінити. 8) Користувач змінює дані. 9) Користувач натискає кнопку «Зберегти зміни».	
Сценарій успішний. Користувач змінив дані у власному аккаунті.	
Розширення:	
1a	Користувач ввів неправильні дані у поля для входу, система повідомляє про це користувача та пропонує спробувати альтернативні способи.
1b	Користувач увів некоректні дані у поля, які хотів змінити. Зміни не збережуться, доки користувач не виправить усі дані на коректні.
Усі сценарії розширення успішно виконані.	

Таблиця 4.3 – Тестування оформлення покупки книги

Діючі особи	Користувач
Мета	Оформити покупку книги
Передумова	Користувач має обрати хоча б одну книгу у кошик
Успішний сценарій: 1) Користувач натискає кнопку «Оформити». 2) Система направляє користувача на сторінку для оформлення замовлення. 3) Користувач заповнює усі поля. 4) Користувач натискає кнопку «Готово» 5) Система повідомляє, що користувач успішно оформив замовлення.	

Кінець таблиці 4.3

Сценарій успішний. Користувач оформив замовлення.	
Розширення:	
1a	Користувач хоче додати кількість книг більжу ніж зареєстровано у системі. Система повідомляє, що такої кількості книг немає.
1b	Користувач увів некоректні дані у поля для оформлення замовлення. Зміни не збережуться, доки користувач не виправить усі дані на коректні.
Усі сценарії розширення успішно виконані.	

У процесі тестування було проаналізовано роботу користувацьких сценаріїв: авторизація та пошук книги за назвою. Особливу увагу приділено перевірці взаємодії з базою даних, збереження та оновлення інформації. Було протестовано інтерфейс на різних розширеннях екрана для забезпечення адаптивності дизайну. Також перевірено безпеку передачі даних через токени авторизації (JWT). Усі знайдені помилки та недоліки були усунуті, що дозволило досягти стабільної, безпечної та зручної роботи системи для кінцевого користувача.

Висновки до розділу 4

Здійснено практичну реалізацію вебзастосунку відповідно до визначених технічних і функціональних вимог. Зокрема, реалізовано повноцінний програмний прототип, що включає модуль авторизації на основі технології JWT, що забезпечує безпечну та надійну автентифікацію користувачів.

Розроблений користувацький інтерфейс відповідає сучасним вимогам до зручності, інтуїтивності та адаптивності, що підвищує ефективність взаємодії користувача з системою. Особливу увагу приділено ергономіці розміщення елементів, доступності функціоналу та візуальній привабливості.

Окремим етапом стала підготовка детальної інструкції користувача, яка спрямована на полегшення ознайомлення з функціоналом вебзастосунку та забезпечення його ефективного використання кінцевими користувачами. Інструкція містить опис основних сценаріїв взаємодії з системою та поради щодо розв’язання можливих проблем.

В процесі тестування було перевірено коректність виконання основних функцій, зокрема авторизації, обробки запитів і відображення інтерфейсних елементів. Результати тестування підтвердили стабільність, надійність та відповідність програмного продукту очікуваним вимогам.

Продемонстровано успішну реалізацію усіх запланованих етапів розробки та підтверджено готовність вебзастосунку до практичного використання.

ВИСНОВКИ

Написання кваліфікаційної бакалаврської роботи стало важливим етапом для здобуття практичних навичок та застосування теоретичних знань у реальних умовах шляхом розробки вебзастосунку продажу книг.

Для досягнення визначеної мети було виконано такі заплановані завдання:

- проаналізовано предметну галузь;
- оцінінено аналогічні застосунки;
- сформувано вимоги до ПЗ;
- визначено функціонал кожної сторінки застосунку;
- розроблено сценарії використання;
- створено дизайн вебзастосунку;
- спроєктовано базу даних книг, авторів, користувачів;
- розроблено та реалізовано логіку у застосунку;
- протестовано систему.

Головним підсумком кваліфікаційної бакалаврської роботи стало детальне дослідження теми, опрацювання найновіших джерел інформації та виокремлення ключових моментів.

Безпосередня робота дала змогу ознайомитися з поточними процесами у відповідній галузі, визначити перспективні вектори розвитку та можливості для оптимізації.

Особливу увагу приділено вивченню сучасних підходів до побудови вебзастосунків, зокрема архітектур клієнт-серверної взаємодії, принципів REST, засобів автентифікації та авторизації, засобів збереження й обробки даних, що дозволило обрати ефективну модель реалізації проєкту та застосувати актуальні технології.

Виконано порівняльний аналіз існуючих платформ для онлайн-продажу книг, що дозволило визначити функціональні особливості, які найбільшою мірою задовольняють потреби користувачів. На основі аналізу було виокремлено основні функції, які повинні бути реалізовані в розроблюваному застосунку.

Обґрунтовано обраний стек технологій, проведено їх попередню оцінку та сформовано технічну специфікацію, що охоплює функціональні та нефункціональні вимоги до системи.

Робота дала можливість покращити професійні навички, закріпити уже набуті знання по умінню досліджувати та систематизувати дані з різних джерел, а також реалізувати вебзастосунок ждя користувачів з можливістю перегляду каталогу книг, фільтрації книг за назвами та жанрами, реєстрації, формування та оформлення замовлень. Проведено мануальне тестування створеної системи.

Результатом виконаної роботи став вебзастосунок продажу книг.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Eurostat. Eurostat.com URL: https://ec.europa.eu/eurostat/en/web/products-eurostat-news/w/ddn-20240423-2 (last accessed: 26.04.2025).

2. «Книгарня Є» website. book-ye.com. URL: https://book-ye.com.ua/ (last accessed: 27.05.2025)

3. Шуренкова А., Прочуханова О. АНАЛІТИЧНИЙ ЗВІТ за результатами всеукраїнського соціологічного дослідження рівня читання дітей та дорослих у 2024 році. Київ, 2024. 45 с

4. Ke T. T., Sudhir K. Privacy Rights and Data Security: GDPR and Personal Data Markets. Management Science. 2022. URL: https://doi.org/10.1287/mnsc.2022.4614 (last accessed: 11.06.2025).

5. Guo Y., Zhang K., Wang C. Way to success: Understanding top streamer's popularity and influence from the perspective of source characteristics. Journal of Retailing and Consumer Services. 2022. Vol. 64. P. 102786. URL: https://doi.org/10.1016/j.jretconser.2021.102786 (last accessed: 11.06.2025).

6. Amazon Books website. Amazon.de. URL: https://www.amazon.de/-/en/bücher-buch-lesen/b/?ie=UTF8&node=186606&ref_=topnav_storetab_b (last accessed: 28.04.2025).

7. Yakaboo website. Yakaboo. URL: https://www.yakaboo.ua/ (last accessed: 26.05.2025).

8. Amazon Books website. Amazon.de. URL: https://www.amazon.de/-/en/bücher-buch-lesen/b/?ie=UTF8&node=186606&ref_=topnav_storetab_b (last accessed: 28.04.2025).

9. Online Bookstore: Books, NOOK ebooks, Music, Movies & Toys. *Barnes & Noble*. URL: https://www.barnesandnoble.com/ (last accessed: 11.06.2025).

10. Eurostat. URL: https://ec.europa.eu/eurostat/en/web/products-eurostat-news/w/ddn-20240423-2 (last accessed: 26.04.2025).

11. Bielak K., Borek B., Plechawska-Wójcik M. Web application performance analysis using Angular, React and Vue.js frameworks. *Journal of Computer Sciences*

Institute. 2022. Vol. 23. P. 77–83. URL: https://doi.org/10.35784/jcsi.2827 (date of access: 11.06.2025).Swacha J., Kulpa A. Evolution of Popularity and Multiaspectual Comparison of Widely Used Web Development Frameworks. *Electronics*. 2023. Vol. 12, no. 17. P. 3563. URL: https://doi.org/10.3390/electronics12173563 (date of access: 06.05.2025).

12. React documentation. React. URL: https://react.dev/ (last accessed: 29.04.2025).

13. Tailwind CSS documentation. Tailwind CSS. URL: https://tailwindcss.com/ (last accessed: 29.04.2025).

14. Node.js documentation. Node.js. URL: https://nodejs.org/en (last accessed: 27.05.2025).

15. Koa - next generation web framework for node.js. Koa - next generation web framework for node.js. URL: https://koajs.com/ (last accessed: 11.06.2025).

16. Comparison of SQL and NoSQL databases with different workloads: MongoDB vs MySQL evaluation. *IEEE Xplore*. URL: https://ieeexplore.ieee.org/document/10041513 (date of access: 01.05.2025).

17. MySQL documentation. MySQL. URL: https://www.mysql.com/ (last accessed: 24.05.2025).

18. Dahri N., Setya Hadi H., Formis R. PERANCANGAN SISTEM E-COMMERCE BERBASIS KEMITRAAN DAGANG. *Jurnal Manajemen Teknologi Informatika.* 2023. Vol. 1, no. 3. P. 176–188. URL: https://doi.org/10.70038/jentik.v1i3.58 (last accessed: 11.06.2025).

19. Official PCI Security Standards Council Site. PCI Security Standards Council. URL: https://www.pcisecuritystandards.org/ (last accessed: 01.05.2025).

20. Hu Q., Asghar M. R., Brownlee N. A large-scale analysis of HTTPS deployments: Challenges, solutions, and recommendations. *Journal of Computer Security*. 2020. P. 1–26. URL: https://doi.org/10.3233/jcs-200070 (last accessed: 01.05.2025).

ДОДАТОК А

Реалізація CRUD-операцій

```
// @ts-check
/**
 * @typedef {import('mysql2/promise').Pool} SQL
 * @typedef {import('@koa/router')} Router
 */
/**
 * @param {SQL} sql
 * @param {Router} router
 */
export default (sql, router) => {
  // LIST
  router.get('books', async (ctx) => {
    const [rows] = await sql.query('SELECT * FROM books');
    ctx.body = { data: rows };
  });
  router.get('books/:id', async (ctx) => {
    const { id } = ctx.params;
    const [rows] = await sql.query('SELECT * FROM books WHERE id = ?', [id]);
    ctx.body = { data: rows };
  });
  // CREATE
  router.post('books', async (ctx) => {
    const { title, year, authorId } = ctx.request.body;
    const [result] = /** @type {[import('mysql2').ResultSetHeader, any]} */ (await
sql.query(
      'INSERT INTO books (title, year, authorId) VALUES (?, ?, ?)',
      [title, year, authorId]
    ));
    ctx.body = { data: { count: result.affectedRows } };
  });
  // UPDATE
  router.put('books/:id', async (ctx) => {
    const { id } = ctx.params;
    const { title, year, authorId } = ctx.request.body;
    const [result] = /** @type {[import('mysql2').ResultSetHeader, any]} */ (await
sql.query(
      'UPDATE books SET title = ?, year = ?, authorId = ? WHERE id = ?',
      [title, year, authorId, id]
    ));
    ctx.body = { data: { count: result.affectedRows } };
  });
  // DELETE
  router.del('books/:id', async (ctx) => {
    const { id } = ctx.params;
    const [result] = /** @type {[import('mysql2').ResultSetHeader, any]} */ (await
sql.query(
      'DELETE FROM books WHERE id = ?',
      [id]
    ));
    ctx.body = { data: { count: result.affectedRows } };
  });
};
```