Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр

на тему: «Розробка інформаційної системи електронної бібліотеки з використанням NodeJS»

Виконав: студент IV курсу, групи СП-43
спеціальності 121 «Інженерія програмного забезпечення»

		Чернихівський В.Б.
	(підпис)	(прізвище та ініціали)
Керівник		Михалик Д.М.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Стоянов Ю.М.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Петрик М.Р.
	(підпис)	(прізвище та ініціали)
Рецензент		
	(підпис)	(прізвище та ініціали)

Тернопіль
2026

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
(повне найменування вищого навчального закладу)

Факультет *комп'ютерно-інформаційних систем і програмної інженерії*

Кафедра *програмної інженерії*

ЗАТВЕРДЖУЮ

Завідувач кафедри *Петрик М.Р.*

«___» 2026 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня *бакалавр*
(назва освітнього ступеня)

за спеціальністю *121 «Інженерія програмного забезпечення»*
(шифр і назва спеціальності)

студенту *Чернихівському Вадиму Богдановичу*
(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) *Розробка інформаційної системи електронної бібліотеки з використанням NodeJS*

Керівник проекту (роботи) *Михалик Дмитро Михайлович, к.т.н., доцент*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «___» _____ 2026 року №_____

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи *Бізнес-процеси електронної бібліотеки, принципи розробки програмного забезпечення, NodeJS, типи UML-діаграм*

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз домену та вимог до інформаційної системи електронної бібліотеки
2. Проєктування інформаційної системи електронної бібліотеки
3. Програмна реалізація та тестування інформаційної системи електронної бібліотеки
4. Безпека життєдіяльності, основи охорони праці. Висновки

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Актуальність і мета роботи. 2. Основні процеси роботи електронної бібліотеки.
3. Класифікація процесів електронної бібліотеки. 4. Архітектура інформаційної системи.
5. Діаграми варіантів використання. 6. ER-діаграма бази даних. 7. Діаграми класів.
8. Схема організації REST API та структури NodeJS-проєкту. 9.Інтерфейс користувачів
10. Результати тестування системи. 10. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	*Сенчишин В.С., доц. каф. МТ*		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	*Аналіз домену та вимог до інформаційної системи електронної бібліотеки*		
2.	*Проєктування інформаційної системи електронної бібліотеки*		
3.	*Програмна реалізація та тестування інформаційної системи електронної бібліотеки*		
4.	*Безпека життєдіяльності, основи охорони праці*		
5.	*Оформлення пояснювальної записки і графічного матеріалу*		
6.	*Попередній захист кваліфікаційної роботи бакалавра*		
7.	*Захист кваліфікаційної роботи бакалавра*		

Студент _____ *Чернихівський В.Б.*
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ *Михалик Д.М.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка інформаційної системи електронної бібліотеки з використанням NodeJS // Кваліфікаційна робота освітнього рівня «Бакалавр» // Чернихівський Вадим Богданович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем і програмної інженерії, кафедра програмної інженерії, група СП-43 // Тернопіль, 2026 // С. 72, рис. – 15, табл. – 18, ліст. – 3, бібліогр. – 31.

Ключові слова: інформаційна система; електронна бібліотека; веб-застосунок; NodeJS; Express.js; REST API; PostgreSQL.

Кваліфікаційна робота присвячена проєктуванню та розробці інформаційної системи електронної бібліотеки з використанням NodeJS. Розроблена система призначена для автоматизації роботи з бібліотечними ресурсами, користувачами, електронними файлами, бронюванням та операціями видачі й повернення книг.

У першому розділі розглянуто електронну бібліотеку як об'єкт автоматизації, проаналізовано основні процеси її функціонування, сформовано вимоги до інформаційної системи та обґрунтовано вибір засобів розробки серверної частини веб-застосунку.

У другому розділі виконано проєктування інформаційної системи електронної бібліотеки. Визначено архітектуру програмного рішення, ролі користувачів, діаграми варіантів використання, класи предметної області, програмні модулі, структуру бази даних, REST API та загальну організацію NodeJS-проєкту.

У третьому розділі описано реалізацію серверної та клієнтської частин веб-застосунку. Розглянуто побудову REST API, взаємодію з базою даних PostgreSQL та інтерфейсу керування бібліотечними даними. Також наведено результати функціонального тестування основних можливостей системи.

У четвертому розділі розглянуто питання безпеки життєдіяльності та охорони праці.

ABSTRACT

Development of an Electronic Library Information System Using NodeJS// Bachelor's Qualification Work// Chernykhivskyi Vadym Bohdanovych // Ternopil Ivan Puluj National Technical University, Faculty of Computer Information Systems and Software Engineering, Department of Software Engineering, group SP-43 // Ternopil, 2026 // P. 72, fig. – 15, tabl. – 18, listings – 3, references – 31.

Keywords: software; software quality; expert evaluation; web application; evaluation criteria; ASP.NET MVC; C#; MS SQL Server.

Keywords: information system; electronic library; web application; NodeJS; Express.js; REST API; PostgreSQL.

The qualification thesis is devoted to the design and development of an electronic library information system using NodeJS. The developed system is intended to automate work with library resources, users, electronic files, reservations, and book lending and return operations.

The first chapter considers the electronic library as an object of automation, analyzes the main processes of its functioning, formulates the requirements for the information system, and substantiates the choice of tools for developing the server side of the web application.

The second chapter presents the design of the electronic library information system. It defines the software architecture, user roles, use case diagrams, domain classes, software modules, database structure, REST API, and the general organization of the NodeJS project.

The third chapter describes the implementation of the server and client parts of the web application. It considers the development of the REST API, interaction with the PostgreSQL database, and the interface for managing library data. The results of functional testing of the main system features are also presented.

The fourth chapter considers issues of life safety and occupational safety.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД	База даних
ПЗ	Програмне забезпечення
ІС	Інформаційна система
ЕБ	Електронна бібліотека
API	Application Programming Interface
BE	Back End
FE	Front End
HTTP	HyperText Transfer Protocol
JSON	JavaScript Object Notation
REST	Representational State Transfer
SQL	Structured Query Language
DBMS	Database Management System
ORM	Object-Relational Mapping
ER	Entity Relations
UML	Unified Modeling Language
UI	User Interface
UX	User Experience
JWT	JSON Web Token
MVC	Model-View-Controller

ВСТУП

У сучасних умовах інформаційні системи відіграють важливу роль у забезпеченні ефективного доступу до даних, автоматизації організаційних процесів та підвищенні якості обслуговування користувачів. Однією з актуальних сфер застосування таких систем є бібліотечна діяльність, яка поступово переходить від традиційних форм обліку та надання ресурсів до електронних сервісів, орієнтованих на швидкий пошук, централізоване зберігання даних і зручну взаємодію з користувачами.

Електронна бібліотека є програмною інформаційною системою, яка забезпечує облік книжкового фонду, збереження відомостей про авторів, категорії, електронні ресурси, користувачів, операції видачі, повернення та бронювання книг. На відміну від простого каталогу, така система повинна підтримувати різні ролі користувачів, механізми авторизації, пошуку, фільтрації, адміністрування даних і контролю доступу до ресурсів. Це потребує застосування сучасних підходів до проєктування програмного забезпечення, побудови бази даних, організації серверної логіки та розробки прикладного програмного інтерфейсу.

Актуальність теми зумовлена необхідністю створення гнучких, масштабованих і зручних інформаційних систем для автоматизації роботи з бібліотечними ресурсами. Використання електронної бібліотеки дозволяє скоротити час пошуку потрібних матеріалів, зменшити кількість ручних операцій, покращити контроль за станом книжкового фонду та забезпечити доступ користувачів до електронних ресурсів незалежно від місця їх перебування. Особливо важливим є застосування таких систем в освітніх установах, де бібліотечні ресурси використовуються студентами, викладачами та працівниками для навчальної, наукової та методичної діяльності.

Для реалізації серверної частини інформаційної системи доцільним є використання платформи NodeJS. Вона забезпечує ефективну обробку запитів, підтримує асинхронну модель виконання, має велику кількість готових програмних пакетів і добре підходить для розробки веб-орієнтованих інформаційних систем.

У поєднанні з фреймворком Express.js, базою даних і REST API, NodeJS дозволяє створити програмне рішення, яке може обслуговувати запити користувачів, виконувати операції з даними, забезпечувати авторизацію та взаємодію між клієнтською і серверною частинами системи.

Метою кваліфікаційної роботи є розробка інформаційної системи електронної бібліотеки з використанням NodeJS, яка забезпечує облік бібліотечних ресурсів, керування користувачами, пошук книг, роботу з електронними матеріалами, а також підтримку основних бібліотечних операцій через серверну частину та базу даних.

1 АНАЛІЗ ДОМЕНУ ТА ВИМОГ ДО ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

Перший розділ присвячено визначенню передумов створення інформаційної системи електронної бібліотеки. У ньому електронну бібліотеку подано як програмний об’єкт автоматизації, що поєднує роботу з бібліотечними ресурсами, користувачами, електронними файлами та обліковими операціями. Окрему увагу приділено опису процесів, які мають бути перенесені в програмне середовище. На основі виконаного аналізу визначено вимоги до майбутнього програмного продукту, зокрема до структури даних, ролей користувачів, безпеки, надійності та серверної логіки. Також охарактеризовано засоби створення серверної частини веб-орієнтованих інформаційних систем і підготовлено основу для обґрунтування використання NodeJS у подальшій реалізації.

1.1. Електронна бібліотека як об’єкт автоматизації та програмна інформаційна система

Електронна бібліотека є різновидом інформаційної системи, призначеної для впорядкованого зберігання, опису, пошуку та надання доступу до бібліотечних ресурсів у цифровому середовищі. Її використання дає змогу перейти від розрізненого обліку матеріалів до централізованого керування даними, що є важливим для навчальних закладів, наукових установ і організацій, які працюють з великими обсягами інформаційних ресурсів [1].

На відміну від традиційного каталогу, електронна бібліотека не обмежується лише відображенням переліку книг або документів. Вона повинна забезпечувати збереження структурованих даних, підтримку користувацьких ролей, контроль доступу, виконання запитів до бази даних і взаємодію з електронними матеріалами. Тому з позиції інженерії програмного забезпечення таку бібліотеку доцільно розглядати як програмну інформаційну систему, що має визначену архітектуру, внутрішню логіку та набір взаємопов’язаних компонентів.

Основу електронної бібліотеки становлять дані про бібліотечні ресурси та пов’язані з ними об’єкти. До таких даних належать відомості про книги, авторів, категорії, електронні файли, фізичні примірники, користувачів, ролі та службові записи. Кожен об’єкт має власні атрибути й зв’язки з іншими об’єктами. Наприклад, книга може належати до певної категорії, мати одного або кількох авторів, бути пов’язаною з електронним файлом або мати кілька фізичних примірників. Така структура потребує продуманого моделювання даних і забезпечення їхньої цілісності [2-4].

Як програмна система електронна бібліотека повинна мати чітко визначені межі відповідальності між основними частинами. Клієнтська частина забезпечує взаємодію користувача із системою, відображення каталогу, форм введення та результатів пошуку. Серверна частина приймає запити, перевіряє вхідні дані, виконує бізнес-правила, контролює доступ і звертається до бази даних. База даних відповідає за збереження структурованої інформації та підтримку зв’язків між сутностями. Такий поділ дозволяє розглядати електронну бібліотеку як веб-орієнтоване програмне рішення з клієнт-серверною організацією.

Важливою характеристикою електронної бібліотеки є підтримка різних категорій користувачів. У системі можуть працювати гість, зареєстрований читач, працівник бібліотеки та адміністратор. Кожна роль має власний набір дозволених дій, що потребує реалізації механізмів автентифікації, авторизації та розмежування прав доступу. Завдяки цьому користувачі отримують доступ лише до тих функцій і даних, які відповідають їхнім повноваженням.

Електронна бібліотека також повинна відповідати вимогам до надійності, безпеки та зручності використання. Оскільки система може працювати з персональними даними користувачів, службовою інформацією про бібліотечний фонд і електронними файлами, необхідно забезпечити захист облікових записів, перевірку введених даних, коректну обробку помилок і контроль доступу до адміністративних функцій. Крім того, програмне рішення має бути придатним до подальшого розширення, наприклад додавання нових типів ресурсів, ролей або звітів.

З погляду розробки програмного забезпечення електронна бібліотека є зручним прикладом інформаційної системи, у якій поєднуються робота з базою даних, серверна логіка, авторизація, обробка запитів, модульна структура та взаємодія з користувачем через веб-інтерфейс. Це дає змогу застосувати сучасні підходи до аналізу вимог, проєктування архітектури, побудови бази даних і тестування програмних модулів [3].

Отже, електронна бібліотека є не лише засобом доступу до книжкових або електронних матеріалів, а повноцінною програмною інформаційною системою. Її розробка потребує формалізації предметної області, визначення структури даних, ролей користувачів, правил доступу та технічних засобів реалізації. У подальших підрозділах буде розглянуто основні процеси роботи такої системи, вимоги до її функціонування та засоби розробки серверної частини.

1.2. Аналіз основних процесів роботи електронної бібліотеки

Робота електронної бібліотеки охоплює сукупність процесів, пов’язаних із формуванням бібліотечного фонду, зберіганням відомостей про ресурси, організацією доступу користувачів до матеріалів і супроводом основних бібліотечних операцій. У межах інформаційної системи ці процеси повинні бути подані у вигляді взаємопов’язаних програмних функцій, які забезпечують цілісність даних, зручність роботи користувачів і контроль за станом ресурсів [5].

Основні процеси роботи електронної бібліотеки можна подати у вигляді узагальненої схеми, у центрі якої розташована система, а навколо неї – ключові напрями її функціонування. Такий підхід дозволяє показати, що електронна бібліотека не обмежується лише зберіганням електронних книг, а виконує низку пов’язаних операцій. Узагальнену схему таких процесів наведено на рис. 1.1.

Рисунок 1.1 – Основні процеси роботи електронної бібліотеки

Одним із базових процесів є формування та ведення каталогу бібліотечних ресурсів. Каталог містить відомості про книги, електронні документи та інші ресурси, які можуть використовуватися читачами. Наявність структурованого каталогу дозволяє швидко виконувати пошук, фільтрацію та подальшу обробку інформації.

Другим важливим процесом є керування користувачами системи. Електронна бібліотека повинна підтримувати облік користувачів, їхню реєстрацію, авторизацію, зміну профільних даних і призначення ролей. Роль користувача визначає набір доступних дій у системі. Наприклад, незареєстрований користувач може переглядати відкритий каталог, читач може бронювати книги або переглядати власну історію операцій, бібліотекар може виконувати видачу та повернення, а адміністратор має доступ до керування користувачами, ролями та довідковою інформацією. Такий підхід дозволяє реалізувати контроль доступу та обмежити виконання службових операцій лише уповноваженими особами [6,7].

Окрему групу становлять процеси пошуку та фільтрації бібліотечних ресурсів. Для електронної бібліотеки важливо забезпечити можливість швидкого

знаходження потрібних матеріалів за назвою, автором, категорією, роком видання, ключовими словами або типом ресурсу. Пошуковий механізм повинен працювати з базою даних і повертати користувачу релевантний перелік результатів. Фільтрація дозволяє уточнювати результати пошуку та зменшувати кількість зайвих записів. З точки зору програмної реалізації цей процес потребує правильної структури даних, індексації основних полів і коректної побудови запитів до бази даних.

Важливим процесом електронної бібліотеки є надання доступу до електронних ресурсів. Якщо книга або документ має цифрову копію, система повинна зберігати відомості про відповідний файл, його формат, шлях або посилання для доступу, а також правила використання. Доступ до електронного файла може бути відкритим або обмеженим залежно від ролі користувача, статусу ресурсу чи внутрішніх правил організації. Тому під час розробки системи необхідно передбачити перевірку прав доступу перед переглядом або завантаженням матеріалу.

Процес бронювання книг є необхідним у випадку, коли бібліотека працює не лише з електронними, а й із фізичними примірниками. Читач може подати запит на бронювання доступної книги, після чого система фіксує дату бронювання, користувача, вибраний ресурс і поточний статус заявки. Бібліотекар або адміністратор може підтвердити, скасувати або завершити бронювання. Автоматизація цього процесу дозволяє уникнути дублювання заявок, контролювати чергу користувачів і відстежувати доступність примірників.

Процес видачі та повернення книг пов’язаний з обліком фізичних примірників бібліотечного фонду. Для кожного примірника доцільно зберігати інвентарний номер, статус, місце зберігання та зв’язок із відповідною книгою. Під час видачі система фіксує користувача, примірник, дату видачі та заплановану дату повернення. Після повернення запис оновлюється, а статус примірника змінюється на доступний. Такий механізм дозволяє зберігати історію операцій і контролювати стан бібліотечного фонду.

Для кращого розуміння структури процесів електронної бібліотеки їх доцільно згрупувати за функціональним призначенням. У такому випадку можна виділити процеси роботи з ресурсами, процеси обслуговування користувачів, операційні процеси та адміністративні процеси [8]. Така класифікація є корисною для подальшого проєктування інформаційної системи, оскільки дає змогу визначити майбутні програмні модулі, їхні функції та межі відповідальності. Класифікацію процесів електронної бібліотеки наведено на рис. 1.2.

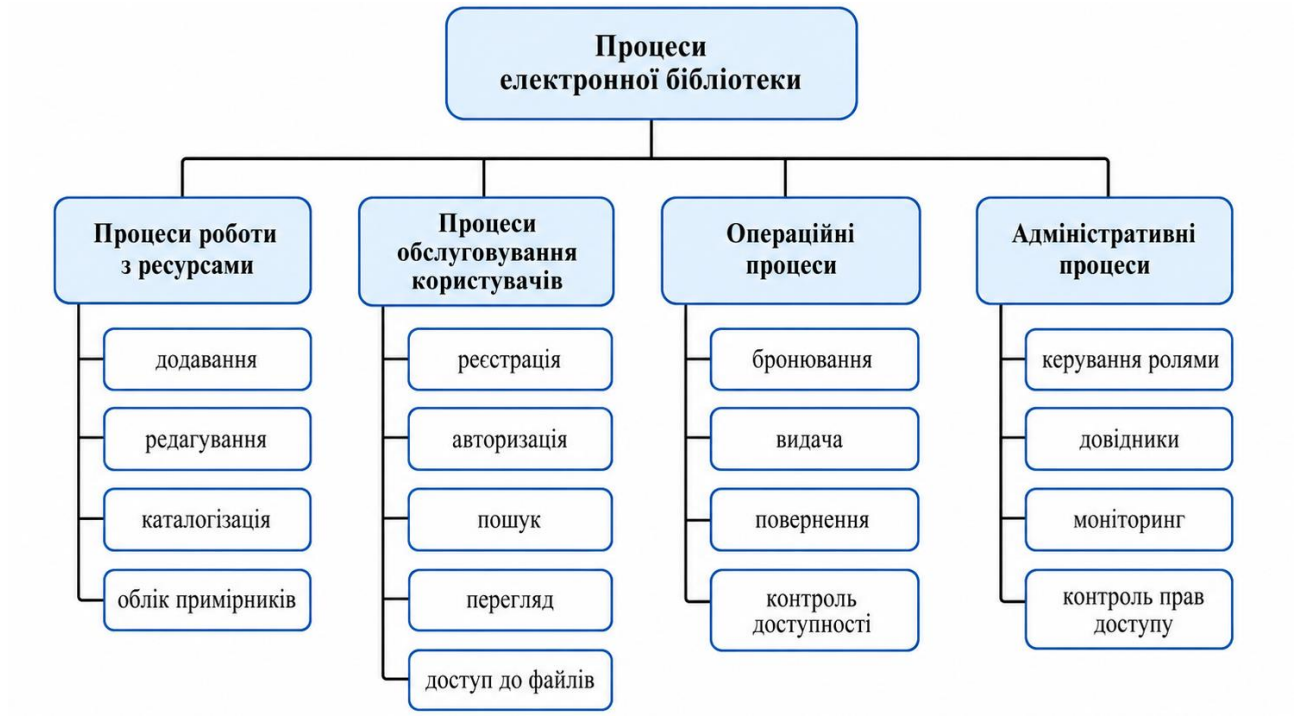

Рисунок 1.2 – Класифікація процесів електронної бібліотеки

До процесів роботи з ресурсами належать додавання, редагування, каталогізація та облік примірників. Вони формують основу інформаційного наповнення системи й безпосередньо пов’язані зі структурою бази даних. Процеси обслуговування користувачів охоплюють реєстрацію, авторизацію, пошук, перегляд ресурсів і доступ до електронних файлів. Операційні процеси включають бронювання, видачу, повернення та контроль доступності ресурсів. Адміністративні процеси пов’язані з керуванням ролями, довідковими даними, моніторингом роботи системи та контролем прав доступу.

Ще одним важливим процесом є адміністрування довідкових і службових даних. До таких даних належать ролі користувачів, категорії книг, автори, видавництва, типи ресурсів, статуси бронювання та статуси примірників. Наявність окремих довідників спрощує підтримку системи, зменшує дублювання даних і забезпечує узгодженість інформації. З програмної точки зору це дозволяє будувати більш гнучку структуру бази даних і спрощує подальше розширення функціональності.

Усі зазначені процеси мають бути взаємопов’язаними. Наприклад, процес бронювання залежить від наявності користувача, книги та доступного примірника. Процес видачі потребує перевірки статусу примірника та фіксації операції в історії. Процес доступу до електронного файла повинен враховувати роль користувача та правила доступу до ресурсу. Тому під час розробки інформаційної системи важливо не лише реалізувати окремі функції, а й забезпечити коректну взаємодію між ними [9].

З позиції інженерії програмного забезпечення аналіз процесів роботи електронної бібліотеки є основою для подальшого формування вимог, побудови архітектури, визначення модулів системи та проєктування бази даних. Кожен процес може бути поданий як окремий функціональний блок, який має вхідні дані, правила обробки, результати виконання та обмеження доступу. Такий підхід дозволяє перейти від опису предметної області до формального проєктування програмного рішення.

1.3. Формування вимог до інформаційної системи електронної бібліотеки

Після визначення основних процесів роботи електронної бібліотеки необхідно перейти до формування вимог до майбутньої інформаційної системи. На цьому етапі опис предметної області перетворюється на набір очікуваних можливостей програмного продукту. Вимоги задають межі системи, визначають її

поведінку, встановлюють правила роботи з даними та формують основу для подальшого проєктування архітектури, бази даних і програмних модулів.

Для інформаційної системи електронної бібліотеки важливо розділити вимоги на декілька груп. Такий підхід дозволяє уникнути змішування користувацьких дій, правил обробки даних і технічних характеристик програмного рішення. У межах цієї роботи визначено функціональні вимоги, вимоги до даних, вимоги до ролей і доступу, нефункціональні вимоги та вимоги до реалізації серверної частини.

Функціональні вимоги визначають, які дії повинна підтримувати система. До них належать створення та редагування записів, пошук інформації, перегляд детальних відомостей, виконання бібліотечних операцій, робота з електронними матеріалами та формування службових записів. Кожна така дія повинна мати визначений результат. Наприклад, після додавання нового ресурсу в системі має з’явитися повний запис із набором обов’язкових атрибутів, а після зміни статусу примірника відповідні дані повинні бути оновлені в базі даних.

Окрему увагу приділено вимогам до даних. Інформаційна система повинна працювати зі структурованими сутностями, між якими існують логічні зв’язки. Основними сутностями є користувач, роль, бібліотечний ресурс, автор, категорія, електронний файл, фізичний примірник, бронювання та запис про видачу. Для кожної сутності необхідно визначити набір полів, правила заповнення, обов’язковість окремих атрибутів і зв’язки з іншими сутностями. Такий підхід дає змогу уникнути хаотичного зберігання інформації та забезпечити цілісність бази даних.

Вимоги до ролей і доступу визначають, які користувачі можуть виконувати певні дії в системі. У межах електронної бібліотеки необхідно передбачити декілька рівнів доступу. Незареєстрований користувач може працювати лише з відкритими даними. Зареєстрований читач отримує доступ до персональних функцій. Працівник бібліотеки виконує операції, пов’язані з обліком і обслуговуванням користувачів. Адміністратор має права керування системними налаштуваннями, ролями та обліковими записами. Такий поділ дає змогу

підвищити керованість програмного рішення та зменшити ризик несанкціонованої зміни даних.

Нефункціональні вимоги характеризують якість роботи системи. До них належать зручність використання, швидкодія, надійність, безпека, масштабованість і можливість супроводу. Для електронної бібліотеки важливо, щоб типові дії виконувалися без зайвих кроків, пошук працював достатньо швидко, помилки оброблялися коректно, а доступ до службових функцій був обмежений. Також система повинна бути придатною до подальшого розширення, наприклад додавання нових типів ресурсів, нових ролей або додаткових звітів.

Вимоги до серверної частини пов’язані з тим, що система розробляється з використанням NodeJS. Серверна частина повинна приймати запити від клієнтської частини, перевіряти коректність отриманих даних, виконувати бізнес-логіку, взаємодіяти з базою даних і повертати результат у стандартизованому форматі. Для цього доцільно передбачити побудову REST API [10], використання маршрутів, контролерів, сервісів, проміжних обробників і окремого шару доступу до даних. Така організація спрощує розробку, тестування та супровід програмного продукту.

Під час формування вимог потрібно також врахувати правила валідації даних. Система не повинна приймати порожні або некоректні значення в обов’язкових полях. Наприклад, запис про книгу не може бути створений без назви, а обліковий запис користувача не може містити неправильний формат електронної пошти. Перевірка вхідних даних має виконуватися на серверному рівні, оскільки саме сервер відповідає за збереження коректного стану системи.

Важливою вимогою є забезпечення цілісності операцій. Деякі дії в електронній бібліотеці впливають одразу на декілька об’єктів. Наприклад, під час видачі книги потрібно створити запис про операцію та змінити статус відповідного примірника. Якщо одна з цих дій не виконалася, система не повинна залишатися в частково оновленому стані. Тому під час подальшого проєктування потрібно передбачити узгоджене виконання пов’язаних операцій із базою даних.

Перелік основних вимог до інформаційної системи електронної бібліотеки наведено в табл. 1.1.

Таблиця 1.1 – Основні вимоги до електронної бібліотеки

Група вимог	Характеристика вимог
Функціональні вимоги	Визначають основні дії системи: створення, редагування, пошук, перегляд, бронювання, видачу, повернення та роботу з електронними файлами
Вимоги до даних	Описують сутності системи, їхні атрибути, зв’язки, обов’язкові поля та правила збереження інформації
Вимоги до ролей	Встановлюють рівні доступу для гостя, читача, працівника бібліотеки та адміністратора
Вимоги до безпеки	Передбачають автентифікацію, авторизацію, захищене зберігання паролів, перевірку прав доступу та валідацію даних
Вимоги до серверної частини	Передбачають реалізацію REST API, обробку запитів, виконання бізнес-логіки та взаємодію з базою даних
Вимоги до надійності	Забезпечують коректне виконання операцій, обробку помилок і недопущення неузгодженого стану даних
Вимоги до супроводу	Передбачають модульну структуру коду, розділення відповідальності між компонентами та можливість подальшого розширення

Сформовані вимоги визначають основу для подальшого проєктування інформаційної системи. Вони дозволяють перейти від загального опису предметної області до конкретних рішень щодо архітектури, структури бази даних, програмних модулів і серверного API. У подальших підрозділах ці вимоги будуть враховані під час вибору засобів розробки та постановки задачі створення інформаційної системи електронної бібліотеки з використанням NodeJS.

1.4. Аналіз засобів розробки серверної частини веб-орієнтованих інформаційних систем

Сучасні інформаційні системи здебільшого реалізуються як веб-орієнтовані програмні рішення. Для електронної бібліотеки це є особливо важливим, оскільки користувачі повинні мати можливість переглядати каталог, виконувати пошук, працювати з електронними ресурсами та здійснювати бібліотечні операції без встановлення окремого програмного забезпечення на власний комп’ютер.

Типова веб-орієнтована інформаційна система складається з клієнтської частини, серверної частини та бази даних. Серверна частина є одним із ключових компонентів інформаційної системи, оскільки саме вона реалізує основні правила роботи програмного продукту.. Тому вибір засобів розробки серверної частини має безпосередній вплив на зручність реалізації, надійність і подальший супровід системи.

Для створення серверної частини веб-орієнтованих систем можуть використовуватися різні технології та мови програмування [11-13]. Поширеними варіантами є PHP, Python, Java, C#, JavaScript із платформою NodeJS та інші засоби. Кожна з цих технологій має власні переваги та сферу застосування. Наприклад, PHP часто використовується для класичних веб-сайтів і систем керування вмістом, Python зручний для швидкої розробки та інтеграції з аналітичними засобами, Java і C# застосовуються для великих корпоративних систем, а NodeJS добре підходить для створення API та обробки великої кількості запитів у веб-застосунках.

Під час вибору технології для серверної частини потрібно враховувати декілька критеріїв. До них належать продуктивність, зручність розробки, підтримка роботи з базами даних, наявність готових бібліотек, можливість побудови REST API [12], підтримка механізмів безпеки, активність спільноти розробників і простота розгортання. Одним із основних підходів до взаємодії між клієнтською та серверною частинами є використання REST API. REST API дозволяє організувати обмін даними між інтерфейсом користувача та сервером через HTTP-запити. Наприклад, окремі запити можуть використовуватися для отримання списку книг, створення нового користувача, оновлення інформації про ресурс або видалення запису. Такий підхід робить систему більш гнучкою, оскільки серверна частина може працювати з різними типами клієнтів: веб-інтерфейсом, мобільним застосунком або іншими зовнішніми сервісами [14].

Важливим елементом серверної частини є організація роботи з базою даних. Для інформаційної системи електронної бібліотеки база даних повинна зберігати пов’язані між собою сутності. Для роботи з такими даними можуть використовуватися реляційні бази даних, наприклад PostgreSQL або MySQL, а

також нереляційні бази даних, наприклад MongoDB. У випадку, коли структура даних має багато зв’язків між сутностями, доцільним є використання реляційної бази даних.

Для спрощення взаємодії серверної частини з базою даних часто використовують ORM-засоби. ORM дозволяє працювати з таблицями бази даних через програмні моделі, зменшує кількість ручного SQL-коду та робить код більш зрозумілим. У середовищі NodeJS для цього можуть використовуватися Prisma, Sequelize або TypeORM [15]. Такі засоби допомагають описати структуру даних, виконувати запити, створювати зв’язки між сутностями та підтримувати узгодженість програмного коду з базою даних.

Окрему увагу під час розробки серверної частини необхідно приділяти безпеці. Сервер повинен перевіряти вхідні дані, контролювати права доступу користувачів, захищати паролі та обмежувати доступ до адміністративних функцій. Для цього використовують механізми автентифікації, авторизації, хешування паролів, токени доступу та проміжні обробники запитів. У системі електронної бібліотеки це необхідно для того, щоб службові операції могли виконувати лише користувачі з відповідними правами. Також важливою вимогою до серверної частини є модульність. Програмний код доцільно поділяти на маршрути, контролери, сервіси, моделі, репозиторії та проміжні обробники. Така структура дозволяє розділити відповідальність між компонентами системи. Наприклад, маршрути визначають адреси API, контролери приймають запити, сервіси реалізують бізнес-логіку, а репозиторії відповідають за доступ до бази даних. Завдяки цьому система стає зрозумілішою, простішою для тестування та зручнішою для подальшого розширення [16].

Отже, для розробки серверної частини веб-орієнтованої інформаційної системи електронної бібліотеки необхідно обрати технології, які забезпечують обробку HTTP-запитів, реалізацію REST API, роботу з базою даних, захист доступу та модульну організацію програмного коду. Аналіз сучасних засобів розробки показує, що для такої задачі доцільно використовувати платформу NodeJS у поєднанні з серверним фреймворком, ORM-засобом і реляційною базою даних.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

У другому розділі розглянуто проєктування інформаційної системи електронної бібліотеки. На основі сформованих вимог визначено загальну архітектуру програмного рішення, описано ролі користувачів, побудовано діаграми варіантів використання, спроєктовано класи предметної області, програмні модулі серверної частини та структуру бази даних.

Основну увагу приділено інженерній підготовці системи до реалізації на основі NodeJS. У розділі визначено логіку взаємодії клієнтської частини, серверної частини та бази даних, описано принципи побудови REST API, розмежування доступу, організації модулів і збереження даних.

2.1. Проєктування архітектури інформаційної системи електронної бібліотеки на концептуальному рівні

На основі вимог, визначених у першому розділі, для інформаційної системи електронної бібліотеки обрано клієнт-серверну архітектуру з виділенням клієнтського рівня, серверного рівня та рівня зберігання даних. Такий підхід дозволяє розмежувати відповідальність між компонентами програмного рішення та забезпечити керовану взаємодію користувача із системою через серверну частину.

У межах проєктованої системи клієнтський рівень відповідає за відображення інтерфейсу та передавання запитів користувача до серверної частини. На цьому рівні розміщуються сторінки каталогу, форми авторизації, елементи пошуку, сторінки перегляду інформації про ресурси, особистий кабінет читача та службові інтерфейси для працівника бібліотеки або адміністратора. Клієнтська частина не має прямого доступу до бази даних, а всі операції виконує через програмний інтерфейс серверної частини.

Серверний рівень є центральною частиною архітектури. Він реалізується на основі NodeJS і відповідає за приймання HTTP-запитів, маршрутизацію, перевірку вхідних даних, виконання бізнес-логіки, контроль доступу та формування відповідей клієнтській частині. У серверній частині передбачається виділення окремих компонентів для авторизації, роботи з користувачами, керування бібліотечними ресурсами, обробки бронювань, обліку видачі та повернення, а також адміністрування довідкової інформації.

Взаємодія між клієнтським і серверним рівнями здійснюється через REST API. Кожна група функцій системи має бути представлена окремою групою маршрутів. Наприклад, маршрути авторизації використовуються для входу користувача в систему, маршрути ресурсів – для роботи з каталогом, маршрути бронювання – для створення та обробки заявок, а маршрути адміністрування – для керування службовими даними. Така організація дає змогу зробити серверну частину зрозумілою, розширюваною та придатною до тестування [17-19].

Рівень зберігання даних представлено реляційною базою даних. Вона призначена для збереження відомостей про користувачів, ролі, книги, авторів, категорії, електронні файли, примірники, бронювання та записи про бібліотечні операції. Уся взаємодія з базою даних виконується тільки через серверну частину, що дозволяє централізовано контролювати коректність операцій, права доступу та цілісність даних.

Для внутрішньої організації серверної частини доцільно використати багаторівневу структуру. На рівні маршрутів визначаються адреси API та HTTP-методи. Контролери приймають запити, отримують параметри та передають їх у відповідні сервіси. Сервіси реалізують прикладну логіку електронної бібліотеки. Шар доступу до даних відповідає за виконання операцій із базою даних. Проміжні обробники використовуються для перевірки токенів, ролей, валідації даних і централізованої обробки помилок [18].

Архітектуру інформаційної системи електронної бібліотеки на концептуальному рівні проілюстровано на рис. 2.1.

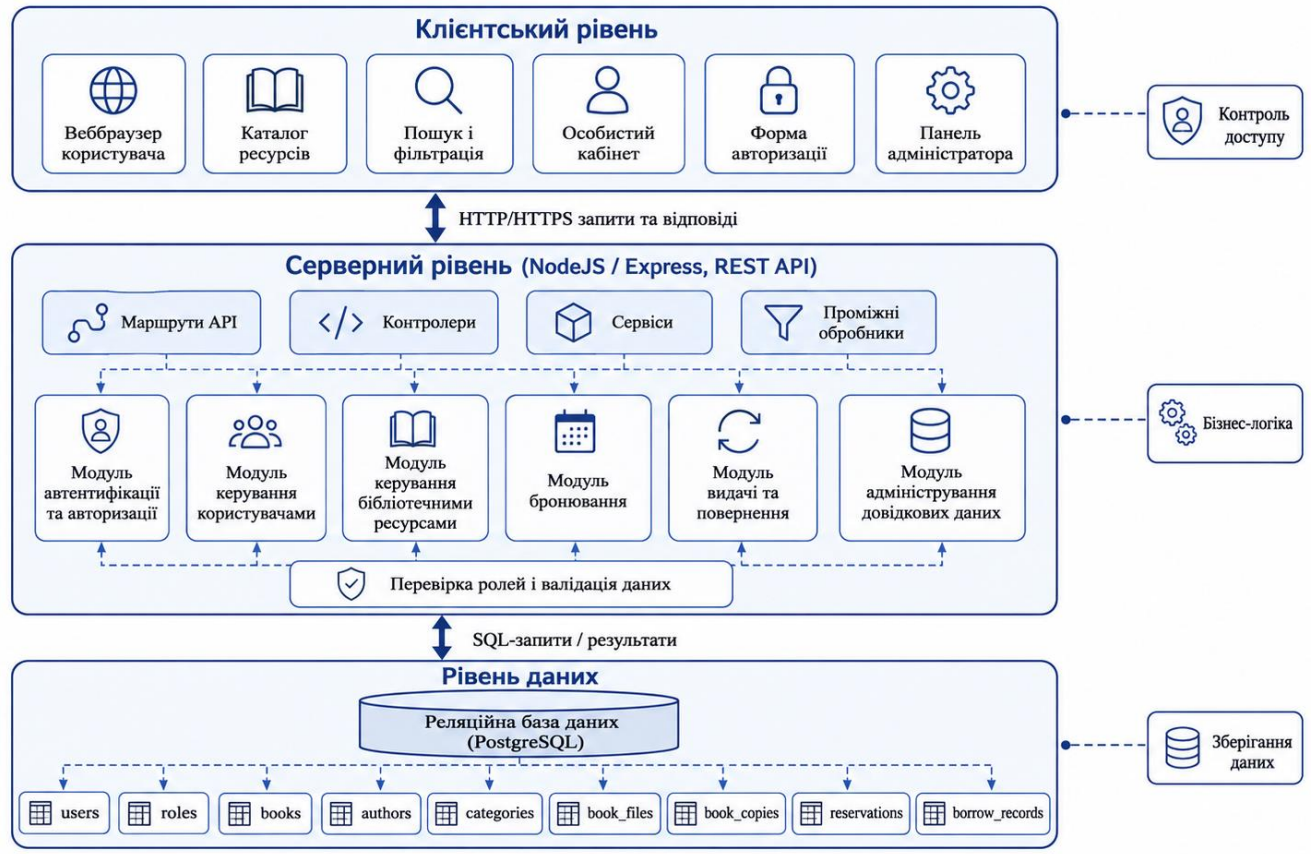

Рисунок 2.1 – Загальна архітектура інформаційної системи електронної бібліотеки

Запит користувача, в запропонованій архітектурі, проходить послідовно через кілька етапів. Спочатку користувач виконує дію у веб-інтерфейсі. Далі клієнтська частина формує HTTP-запит до відповідного маршруту REST API. Сервер перевіряє отримані дані, визначає права доступу користувача, викликає потрібний сервіс і, за потреби, звертається до бази даних. Після виконання операції сервер повертає відповідь клієнтській частині, а результат відображається у веб-інтерфейсі [19].

Окрему роль в архітектурі відіграє механізм контролю доступу. Він розміщується на серверному рівні, оскільки саме сервер повинен визначати, чи має користувач право виконувати певну операцію. Це дозволяє уникнути ситуації, коли обмеження існують лише в інтерфейсі, але можуть бути обійдені прямим зверненням до API. Тому перевірка ролі користувача має виконуватися перед доступом до службових маршрутів і операцій, які змінюють дані.

Запропонована архітектура забезпечує розділення відповідальності між основними частинами системи. Клієнтський рівень відповідає за взаємодію з користувачем, серверний рівень – за виконання прикладної логіки та контроль доступу, а база даних – за збереження інформації. Такий поділ спрощує подальше проєктування функціональних модулів, структури бази даних і REST API серверної частини.

Отже, архітектура інформаційної системи електронної бібліотеки будується як клієнт-серверне програмне рішення з серверною частиною на NodeJS, REST API та реляційною базою даних. Вона враховує вимоги до модульності, безпеки, цілісності даних і подальшого розширення системи, а також створює основу для детального проєктування варіантів використання та функціональних модулів.

2.2. Побудова діаграм варіантів використання системи

Після визначення загальної архітектури інформаційної системи необхідно деталізувати зовнішню поведінку програмного рішення з погляду користувачів. Для цього використано діаграми варіантів використання, які дають змогу подати систему не через її внутрішню реалізацію, а через набір дій, доступних окремим категоріям користувачів. Такий підхід є зручним на етапі проєктування, оскільки дозволяє пов’язати сформовані вимоги з майбутніми програмними модулями [21].

У межах інформаційної системи електронної бібліотеки виділено чотири основні категорії користувачів: гість, читач, бібліотекар та адміністратор. Кожен із них взаємодіє із системою в межах власних повноважень. Гість має мінімальний набір дій, пов’язаний із переглядом відкритої частини каталогу. Читач працює з персональними функціями та бібліотечними ресурсами. Бібліотекар відповідає за операції обслуговування користувачів і роботу з примірниками. Адміністратор виконує службові дії, пов’язані з налаштуванням системи, обліковими записами та довідковими даними.

Загальна діаграма варіантів використання відображає основні межі інформаційної системи та показує, які групи функцій доступні кожному актору. Її

доцільно використовувати як початкову модель, що фіксує взаємодію користувачів із системою на верхньому рівні. На такій діаграмі не деталізуються внутрішні алгоритми виконання операцій, а показуються лише дії, які користувач може ініціювати через веб-інтерфейс. Загальну діаграму варіантів використання інформаційної системи електронної бібліотеки наведено на рис. 2.2.

Рисунок 2.2 – Загальна діаграма варіантів використання інформаційної системи електронної бібліотеки

На загальній діаграмі гість пов’язаний із варіантами використання, що не потребують авторизації: перегляд відкритого каталогу, пошук ресурсів і перегляд загальної інформації про книгу. Читач має доступ до розширених дій, зокрема входу в систему, роботи з особистим кабінетом, перегляду доступних електронних матеріалів і створення заявки на бронювання. Бібліотекар взаємодіє з функціями обліку примірників, підтвердження бронювань, реєстрації видачі та повернення. Адміністратор має доступ до керування користувачами, ролями, довідниками та системними записами.

Для зменшення складності загальної моделі окремо спроєктовано діаграму варіантів використання для читача. Вона дає змогу уточнити, які саме дії виконує користувач після авторизації та які функції повинні бути реалізовані в клієнтській і серверній частинах системи. Основними варіантами використання для читача є авторизація, перегляд каталогу, пошук і фільтрація ресурсів, перегляд детальної інформації про книгу, доступ до електронного файла, створення бронювання та перегляд власної історії операцій. Діаграму варіантів використання для читача наведено на рис. 2.3.

Рисунок 2.3 – Діаграма варіантів використання для читача

На діаграмі для читача варіант використання «Авторизуватися» є передумовою для виконання персональних дій. Перегляд каталогу та пошук можуть бути доступними і без входу в систему, однак бронювання, перегляд особистої історії та доступ до закритих електронних матеріалів потребують ідентифікації користувача. Це враховується під час подальшого проєктування

REST API, оскільки для таких запитів необхідно передбачити перевірку токена та прав доступу.

Окрема діаграма для бібліотекаря й адміністратора потрібна тому, що службові дії мають більшу кількість обмежень і безпосередньо змінюють стан системи. Бібліотекар працює з операціями, пов’язаними з обслуговуванням читачів: перевіряє заявки на бронювання, реєструє видачу примірника, фіксує повернення та оновлює статус доступності. Адміністратор виконує ширший набір службових функцій, серед яких керування обліковими записами, призначення ролей, редагування довідників, додавання або коригування бібліотечних ресурсів. Діаграму варіантів використання для бібліотекаря та адміністратора наведено на рис. 2.4.

Рисунок 2.4 – Діаграма варіантів використання для бібліотекара та адміністратора

У цій діаграмі важливо відокремити операційні дії бібліотекара від адміністративних дій. Бібліотекар не повинен мати повного доступу до системних

налаштувань, тоді як адміністратор може керувати службовими об’єктами та користувацькими ролями. Такий поділ дозволяє точніше визначити правила авторизації та уникнути надмірного надання прав доступу. У подальшій реалізації це буде враховано через перевірку ролей на рівні серверних маршрутів і проміжних обробників.

Проєктування кількох діаграм варіантів використання дає змогу розділити систему на зрозумілі функціональні області. Загальна діаграма фіксує повну картину взаємодії акторів із системою, діаграма читача деталізує користувацькі сценарії, а діаграма бібліотекара та адміністратора описує службові операції. Отже, діаграми варіантів використання визначають зовнішню поведінку інформаційної системи електронної бібліотеки та показують, які дії повинні бути доступні різним категоріям користувачів. На їх основі можна перейти до проєктування функціональних модулів, оскільки кожна група варіантів використання відповідає окремій частині майбутньої серверної логіки.

2.3. Проєктування класів та програмних модулів інформаційної системи

Після визначення архітектури системи та побудови діаграм варіантів використання необхідно перейти до проєктування внутрішньої програмної структури інформаційної системи. На цьому етапі визначаються основні класи, їхні атрибути, зв’язки між об’єктами, а також програмні модулі, які забезпечуватимуть реалізацію серверної логіки. Такий підхід дає змогу перейти від опису зовнішньої поведінки системи до її логічної будови.

Для інформаційної системи електронної бібліотеки доцільно виділити дві групи класів. Перша група описує об’єкти предметної області, які відповідають основним сутностям системи. До неї належать класи користувача, ролі, книги, автора, категорії, примірника, електронного файлу, бронювання та запису про видачу. Друга група пов’язана з реалізацією серверної логіки й охоплює контролери, сервіси та компоненти доступу до даних.

Класи предметної області відображають інформаційні об’єкти, з якими працює система. Клас User описує користувача системи та містить відомості про його ім’я, електронну пошту, пароль у захищеному вигляді та роль. Клас Role визначає рівень доступу користувача. Клас Book зберігає основну інформацію про бібліотечний ресурс, зокрема назву, опис, рік видання, видавництво та зв’язки з авторами й категоріями. Класи Author і Category використовуються для впорядкування каталогу та забезпечення зручного пошуку.

Для обліку фізичних примірників передбачається клас BookCopy. Він пов’язаний із відповідною книгою та містить інвентарний номер, місце зберігання і поточний статус. Такий підхід дозволяє відокремити опис книги як інформаційного ресурсу від конкретних примірників, які можуть бути доступними, виданими, забронюваними або тимчасово недоступними.

Клас FileResource призначений для зберігання даних про електронні матеріали, пов’язані з книгою або іншим бібліотечним ресурсом. Він може містити назву файла, шлях до нього, тип файла та інформацію про доступність. Наявність такого класу дозволяє підтримувати роботу не лише з друкованими примірниками, а й з електронними документами.

Операції, пов’язані з бронюванням і видачею, доцільно подати окремими класами. Клас Reservation описує заявку користувача на бронювання ресурсу, дату створення заявки та її статус. Клас BorrowRecord фіксує факт видачі примірника, дату видачі, очікувану дату повернення та фактичну дату повернення. Завдяки цьому система може зберігати історію роботи з примірниками й контролювати їхню доступність. Діаграму класів предметної області інформаційної системи електронної бібліотеки наведено на рис. 2.5.

На діаграмі класів предметної області показано зв’язок користувача з роллю, книги з авторами та категоріями, книги з примірниками й електронними файлами, а також зв’язок користувача з бронюваннями та записами про видачу. Така модель є проміжною між аналізом вимог і проєктуванням бази даних, оскільки вона визначає, які об’єкти потрібно реалізувати в програмному коді та які зв’язки мають бути збережені в структурі даних.

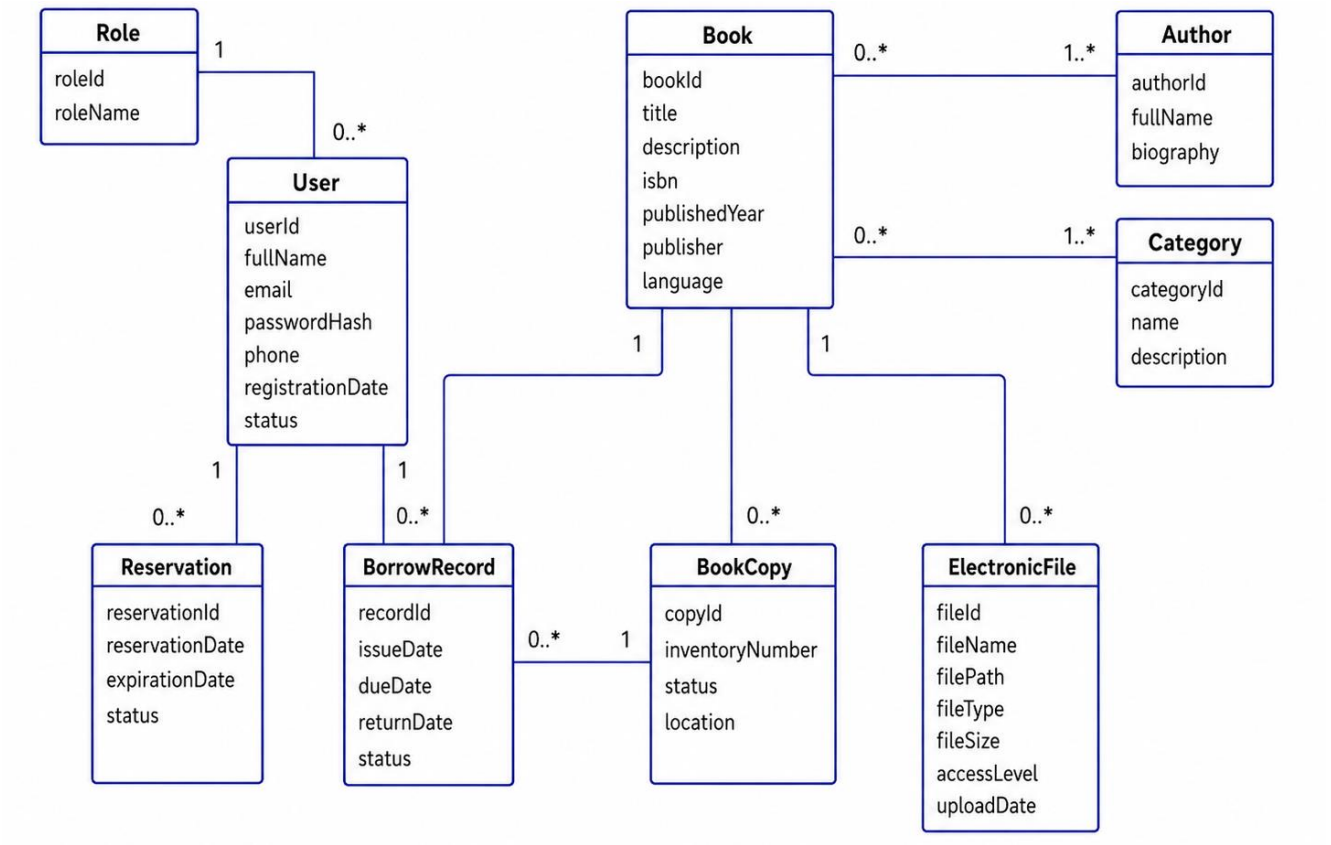

Рисунок 2.5 – Діаграма класів предметної області інформаційної системи електронної бібліотеки

Друга частина проєктування стосується програмних модулів серверної частини. Оскільки система реалізується з використанням NodeJS, доцільно використати багаторівневу організацію коду. У такій структурі запит від клієнтської частини спочатку надходить до маршруту, потім передається в контролер, після цього обробляється сервісом, а операції з базою даних виконуються через окремий репозиторій або ORM-модель.

Контролери відповідають за приймання HTTP-запитів і формування відповідей. Наприклад, AuthController обробляє запити входу та реєстрації, BookController працює із запитами до каталогу, UserController відповідає за операції з користувачами, ReservationController обробляє заявки на бронювання, а BorrowController використовується для операцій видачі та повернення примірників.

Сервіси містять основну прикладну логіку системи. Наприклад, AuthService перевіряє облікові дані користувача, формує токен доступу та виконує дії, пов’язані

з автентифікацією. BookService відповідає за правила роботи з бібліотечними ресурсами, ReservationService перевіряє можливість створення бронювання, а BorrowService забезпечує зміну статусу примірника під час видачі або повернення. Виділення сервісного рівня дозволяє не перевантажувати контролери та зробити код простішим для тестування.

Репозиторії або ORM-моделі забезпечують доступ до бази даних. Вони виконують створення, читання, оновлення та видалення записів, а також формують запити для отримання пов’язаних даних. Наприклад, BookRepository може відповідати за отримання списку книг, пошук за параметрами та вибірку детальної інформації про ресурс. UserRepository працює з обліковими записами користувачів, а ReservationRepository і BorrowRepository забезпечують збереження даних про бібліотечні операції. Діаграму класів серверної логіки інформаційної системи наведено на рис. 2.6.

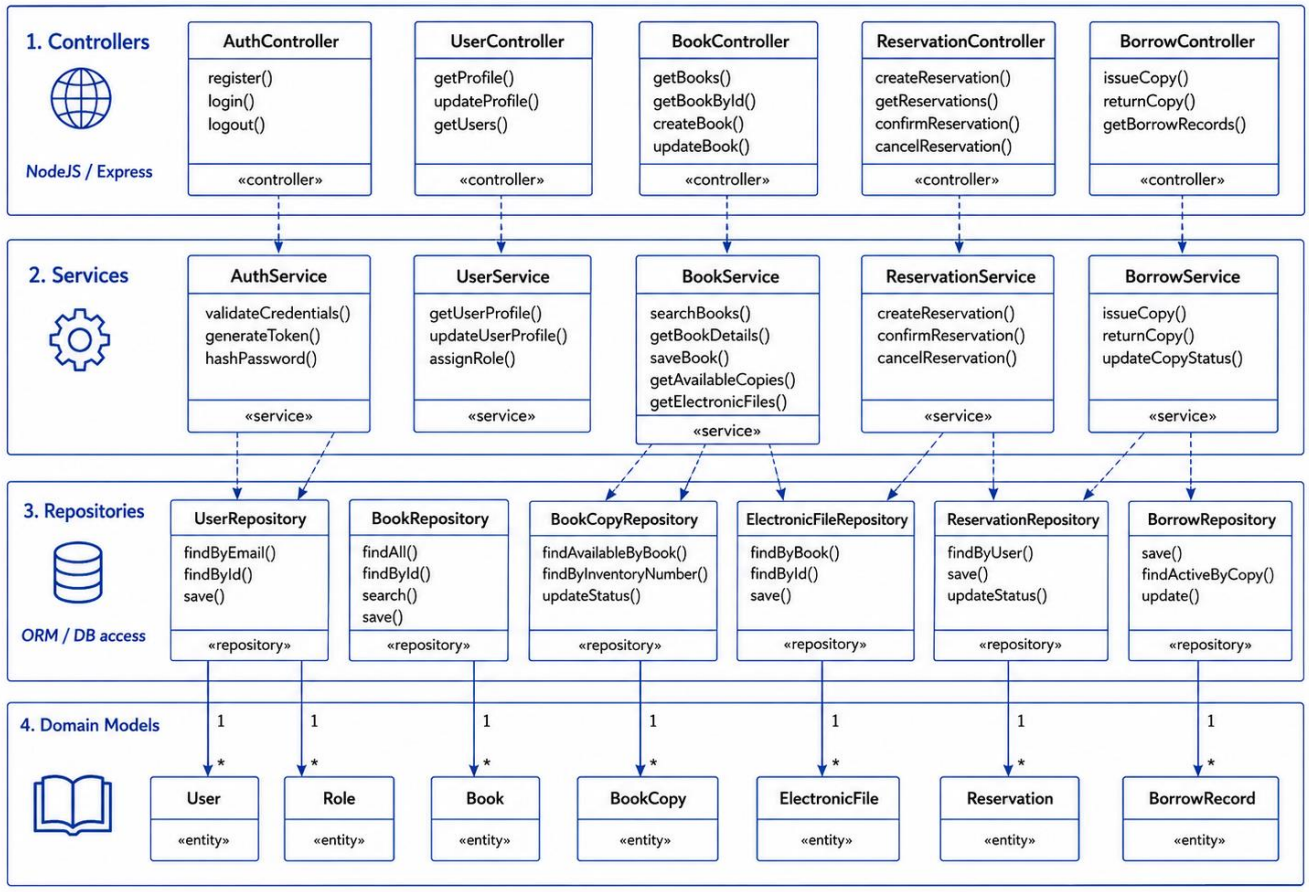

Рисунок 2.6 – Діаграма класів серверної логіки інформаційної системи електронної бібліотеки

На діаграмі серверної логіки показано взаємодію між контролерами, сервісами та репозиторіями [22]. Контролери не повинні напряму звертатися до бази даних, оскільки це ускладнює підтримку системи та порушує розділення відповідальності. Вони мають передавати дані до сервісів, а сервіси, у свою чергу, повинні використовувати репозиторії або ORM-засоби для виконання операцій із даними.

Для узагальнення програмної структури інформаційної системи основні модулі серверної частини наведено в табл. 2.1.

Таблиця 2.1 – Програмні модулі серверної частини інформаційної системи

Програмний модуль	**Призначення модуля**
Модуль авторизації	Реєстрація, вхід у систему, формування токена доступу, перевірка автентичності користувача
Модуль користувачів	Робота з обліковими записами, профілями користувачів і ролями
Модуль бібліотечних ресурсів	Додавання, редагування, пошук і перегляд інформації про книги та пов’язані матеріали
Модуль авторів і категорій	Підтримка довідкових даних для впорядкування каталогу
Модуль примірників	Облік фізичних примірників, інвентарних номерів, місць зберігання та статусів
Модуль електронних файлів	Збереження відомостей про файли та контроль доступу до електронних матеріалів
Модуль бронювання	Створення, перегляд, підтвердження або скасування заявок на бронювання
Модуль видачі та повернення	Реєстрація операцій видачі, повернення та оновлення стану примірників
Модуль адміністрування	Керування службовими даними, ролями, довідниками та системними налаштуваннями

Запропонована структура модулів відповідає логіці серверної реалізації на NodeJS. Кожен модуль може мати власні маршрути, контролери, сервіси та моделі даних. Наприклад, модуль бібліотечних ресурсів може містити маршрути для отримання списку книг, контролер для приймання запитів, сервіс для виконання перевірок і репозиторій для звернення до бази даних. Такий підхід дозволяє поступово розширювати систему без зміни всієї структури програмного продукту.

Отже, проєктування класів і програмних модулів дозволяє визначити внутрішню логіку інформаційної системи електронної бібліотеки. Класи предметної області описують основні об’єкти системи, а серверні класи й модулі задають спосіб реалізації функціональності на рівні NodeJS-застосунку. Отримана структура буде використана під час подальшого проєктування бази даних, REST API та реалізації серверної частини.

2.4. Проєктування бази даних електронної бібліотеки

Проєктування бази даних є важливим етапом створення інформаційної системи електронної бібліотеки, оскільки саме база даних забезпечує довготривале збереження інформації про користувачів, бібліотечні ресурси, електронні файли, фізичні примірники та операції з ними. Якщо діаграма класів відображає логічну структуру програмних об’єктів, то модель бази даних визначає спосіб збереження цих об’єктів у реляційному сховищі.

Для розроблюваної інформаційної системи доцільно використати реляційну базу даних, оскільки предметна область має чітко виражені зв’язки між сутностями. Користувач належить до певної ролі, книга може мати кількох авторів, ресурс може належати до різних категорій, одна книга може мати кілька фізичних примірників і декілька електронних файлів. Операції бронювання, видачі та повернення також пов’язані з користувачами, книгами або конкретними примірниками. Реляційна модель дозволяє подати ці зв’язки за допомогою первинних і зовнішніх ключів.

У межах проєктування бази даних електронної бібліотеки визначено такі основні таблиці: roles, users, books, authors, categories, book_authors, book_categories, book_copies, file_resources, reservations і borrow_records. Такий набір таблиць забезпечує збереження довідкових даних, основної інформації про бібліотечні ресурси, облікових записів користувачів і записів про бібліотечні операції.

Таблиця roles призначена для збереження ролей користувачів системи. Вона використовується для розмежування прав доступу між гостем, читачем, бібліотекарем і адміністратором. Структуру таблиці roles наведено в табл. 2.2.

Таблиця 2.2 – Структура таблиці roles

Поле	Тип даних	Ключ	Призначення
role_id	INT	PK	Унікальний ідентифікатор ролі
role_name	VARCHAR(50)	UNIQUE	Назва ролі користувача
description	VARCHAR(255)		Опис призначення ролі

Таблиця users зберігає облікові записи користувачів. Вона містить персональні дані, електронну пошту, хеш пароля, дату створення запису та посилання на роль користувача. Структуру таблиці users наведено в табл. 2.3.

Таблиця 2.3 – Структура таблиці users

Поле	Тип даних	Ключ	Призначення
user_id	INT	PK	Унікальний ідентифікатор користувача
full_name	VARCHAR(150)		Прізвище та ім’я користувача
email	VARCHAR(120)	UNIQUE	Електронна пошта для входу в систему
password_hash	VARCHAR(255)		Хеш пароля користувача
role_id	INT	FK	Посилання на роль користувача
created_at	TIMESTAMP		Дата створення облікового запису
is_active	BOOLEAN		Ознака активності користувача

Таблиця books є центральною таблицею бази даних, оскільки містить основний опис бібліотечних ресурсів. Вона не зберігає дані про кожен фізичний примірник окремо, а описує книгу як інформаційний ресурс. Структуру таблиці books наведено в табл. 2.4.

Таблиця 2.4 – Структура таблиці books

Поле	Тип даних	Ключ	Призначення
book_id	INT	PK	Унікальний ідентифікатор книги
title	VARCHAR(255)		Назва книги або ресурсу
description	TEXT		Короткий опис ресурсу
published_year	INT		Рік видання
publisher	VARCHAR(150)		Назва видавництва
keywords	VARCHAR(255)		Ключові слова для пошуку
created_at	TIMESTAMP		Дата додавання запису
updated_at	TIMESTAMP		Дата останнього оновлення запису

Таблиця authors призначена для збереження відомостей про авторів бібліотечних ресурсів. Оскільки один автор може бути пов’язаний з багатьма книгами, а одна книга може мати кількох авторів, зв’язок між книгами й авторами реалізується через окрему проміжну таблицю. Структуру таблиці authors наведено в табл. 2.5.

Таблиця 2.5 – Структура таблиці authors

Поле	Тип даних	Ключ	Призначення
author_id	INT	PK	Унікальний ідентифікатор автора
full_name	VARCHAR(150)		Повне ім’я автора
biography	TEXT		Короткі біографічні відомості
created_at	TIMESTAMP		Дата додавання автора

Таблиця categories використовується для тематичного впорядкування бібліотечних ресурсів. Вона дає змогу групувати книги за напрямами, дисциплінами або типами матеріалів. Структуру таблиці categories наведено в табл. 2.6.

Таблиця 2.6 – Структура таблиці categories

Поле	Тип даних	Ключ	Призначення
category_id	INT	PK	Унікальний ідентифікатор категорії
name	VARCHAR(100)	UNIQUE	Назва категорії
description	VARCHAR(255)		Опис категорії

Для реалізації зв’язку багато-до-багатьох між книгами та авторами використовується таблиця book_authors. Вона містить лише зовнішні ключі на відповідні таблиці. Структуру таблиці book_authors наведено в табл. 2.7.

Таблиця 2.7 – Структура таблиці book_authors

Поле	Тип даних	Ключ	Призначення
book_id	INT	PK, FK	Посилання на книгу
author_id	INT	PK, FK	Посилання на автора

Для зв’язку книг із категоріями використовується таблиця book_categories. Вона дозволяє віднести один ресурс до кількох тематичних категорій. Структуру таблиці book_categories наведено в табл. 2.8.

Таблиця 2.8 – Структура таблиці book_categories

Поле	Тип даних	Ключ	Призначення
book_id	INT	PK, FK	Посилання на книгу
category_id	INT	PK, FK	Посилання на категорію

Таблиця book_copies призначена для обліку фізичних примірників книг. Її використання дозволяє відокремити опис книги від конкретних екземплярів, які можуть мати різний статус, інвентарний номер і місце зберігання. Структуру таблиці book_copies наведено в табл. 2.9.

Таблиця 2.9 – Структура таблиці book_copies

Поле	Тип даних	Ключ	Призначення
copy_id	INT	PK	Ідентифікатор примірника
book_id	INT	FK	Посилання на книгу
inventory_number	VARCHAR(50)	UNIQUE	Інвентарний номер примірника
status	VARCHAR(30)		Поточний статус примірника
location	VARCHAR(100)		Місце зберігання примірника
created_at	TIMESTAMP		Дата додавання примірника

Таблиця file_resources використовується для збереження відомостей про електронні файли, пов’язані з бібліотечними ресурсами. Один запис у таблиці books може мати один або кілька електронних файлів. Структуру таблиці file_resources наведено в табл. 2.10.

Таблиця 2.10 – Структура таблиці file_resources

Поле	Тип даних	Ключ	Призначення
file_id	INT	PK	Унікальний ідентифікатор файла
book_id	INT	FK	Посилання на книгу
file_name	VARCHAR(255)		Назва електронного файла
file_path	VARCHAR(255)		Шлях або посилання на файл
file_type	VARCHAR(30)		Тип файла
access_level	VARCHAR(50)		Рівень доступу до файла
uploaded_at	TIMESTAMP		Дата завантаження файла

Таблиця reservations призначена для збереження заявок на бронювання книг. Вона пов’язує користувача з книгою або конкретним примірником і фіксує поточний стан заявки. Структуру таблиці reservations наведено в табл. 2.11.

Таблиця 2.11 – Структура таблиці reservations

Поле	Тип даних	Ключ	Призначення
reservation_id	INT	PK	Унікальний ідентифікатор бронювання
user_id	INT	FK	Посилання на користувача
book_id	INT	FK	Посилання на книгу
copy_id	INT	FK	Посилання на примірник, якщо бронюється конкретний екземпляр
reservation_date	TIMESTAMP		Дата створення заявки
status	VARCHAR(30)		Статус бронювання
expires_at	TIMESTAMP		Дата завершення дії бронювання

Таблиця borrow_records зберігає записи про видачу та повернення фізичних примірників. Вона потрібна для ведення історії операцій і контролю стану примірників. Структуру таблиці borrow_records наведено в табл. 2.12.

Таблиця 2.12 – Структура таблиці borrow_records

Поле	Тип даних	Ключ	Призначення
record_id	INT	PK	Унікальний ідентифікатор запису видачі
user_id	INT	FK	Посилання на користувача
copy_id	INT	FK	Посилання на виданий примірник
issue_date	TIMESTAMP		Дата видачі примірника
due_date	TIMESTAMP		Запланована дата повернення
return_date	TIMESTAMP		Фактична дата повернення
status	VARCHAR(30)		Стан операції видачі

Після визначення структури таблиць необхідно встановити зв’язки між ними. Таблиця roles пов’язана з таблицею users зв’язком один-до-багатьох, оскільки одна роль може бути призначена багатьом користувачам. Таблиця books пов’язана з book_copies і file_resources також зв’язками один-до-багатьох.

Зв’язки між книгами й авторами, а також між книгами й категоріями реалізуються через проміжні таблиці book_authors і book_categories. ER-діаграму бази даних електронної бібліотеки наведено на рис. 2.7.

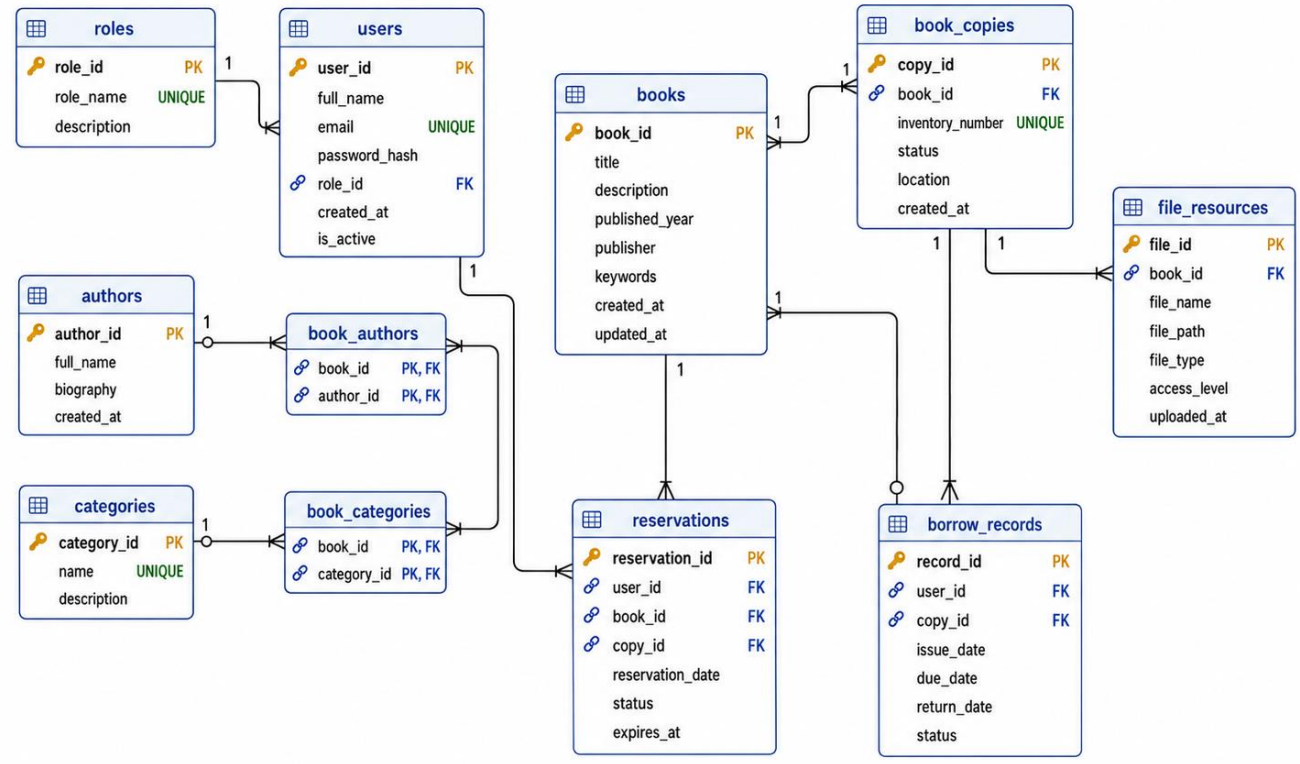

Рисунок 2.7 – ER-діаграма бази даних електронної бібліотеки

На ER-діаграмі відображено основні таблиці бази даних, первинні ключі, зовнішні ключі та зв’язки між сутностями. Центральне місце займає таблиця books, оскільки з нею пов’язані автори, категорії, фізичні примірники та електронні файли. Операційні таблиці reservations і borrow_records пов’язані з користувачами та бібліотечними ресурсами, що дозволяє зберігати історію дій у системі. Узагальнений перелік зв’язків між таблицями наведено в табл. 2.13.

Під час реалізації бази даних необхідно забезпечити обмеження цілісності. Кожна таблиця повинна мати первинний ключ, а зовнішні ключі мають посилатися лише на наявні записи. Наприклад, запис про видачу не може бути створений без користувача та примірника, а електронний файл не може існувати без зв’язку з відповідною книгою. Такі обмеження дозволяють уникнути появи некоректних або неповних даних.

Таблиця 2.13 – Основні зв’язки між таблицями бази даних

Зв’язок	Тип зв’язку	Опис
roles – users	1…N	Одна роль може бути призначена багатьом користувачам
books – book_copies	1…N	Одна книга може мати декілька фізичних примірників
books – file_resources	1…N	Один ресурс може мати декілька електронних файлів
books – authors	N…M	Один ресурс може мати кількох авторів, а автор може мати багато книг
books – categories	N…M	Один ресурс може належати до кількох категорій
users – reservations	1…N	Один користувач може створити багато заявок на бронювання
books – reservations	1…N	Одна книга може бути пов’язана з багатьма заявками
users – borrow_records	1…N	Один користувач може мати багато записів про видачу
book_copies – borrow_records	1…N	Один примірник може мати багато записів в історії видачі

Для полів, які часто використовуються під час пошуку, доцільно передбачити індекси. До таких полів належать email у таблиці users, title і keywords у таблиці books, full_name у таблиці authors, name у таблиці categories, а також inventory_number у таблиці book_copies. Використання індексів дозволяє пришвидшити виконання запитів і зробити роботу каталогу ефективнішою.

Отже, у межах проєктування бази даних визначено повну структуру таблиць інформаційної системи електронної бібліотеки, їхні поля, ключі та зв’язки. Запропонована реляційна модель забезпечує збереження даних про користувачів, ролі, книги, авторів, категорії, фізичні примірники, електронні файли, бронювання та операції видачі. Вона є основою для подальшого проєктування REST API й реалізації серверної частини системи з використанням NodeJS.

2.5. Проєктування REST API та структури проєкту на основі NodeJS

Після проєктування архітектури, класів і бази даних необхідно визначити спосіб взаємодії клієнтської частини із серверною логікою.

Для цього в інформаційній системі електронної бібліотеки передбачено використання REST API, яке забезпечує обмін даними між вебінтерфейсом і серверною частиною на NodeJS.

REST API виконує роль програмного інтерфейсу доступу до функцій системи. Клієнтська частина надсилає HTTP-запит до відповідного маршруту, сервер перевіряє отримані дані, визначає права доступу користувача, виконує необхідну операцію та повертає відповідь у форматі JSON. Такий підхід дозволяє відокремити інтерфейс користувача від серверної реалізації та спростити подальше розширення системи.

Маршрути API доцільно згрупувати за функціональними напрямами. Основними групами є маршрути авторизації, користувачів, бібліотечних ресурсів, авторів, категорій, примірників, електронних файлів, бронювань, а також операцій видачі та повернення. Такий поділ відповідає попередньо визначеним модулям системи та полегшує організацію серверного коду.

Таблиця 2.14 – Основні групи REST API інформаційної системи

Група маршрутів	**Призначення**
/auth	Реєстрація, авторизація та завершення сеансу користувача
/users	Робота з обліковими записами, профілями та ролями
/books	Пошук, перегляд, додавання та редагування бібліотечних ресурсів
/authors	Керування відомостями про авторів
/categories	Керування тематичними категоріями
/copies	Облік фізичних примірників книг
/files	Робота з електронними файлами та доступом до них
/reservations	Створення, перегляд і обробка бронювань
/borrow-records	Реєстрація видачі, повернення та перегляд історії операцій

Під час проєктування API враховано відповідність між HTTP-методами та змістом операцій. Метод GET використовується для отримання даних, POST – для створення нових записів, PUT або PATCH – для оновлення інформації, а DELETE – для видалення або скасування запису. Деталізований перелік ендпоінтів REST API доцільно подано у додатку А.

Для захисту серверних маршрутів передбачено використання проміжних обробників. Вони виконують перевірку токена доступу, контроль ролі користувача, валідацію вхідних даних і централізовану обробку помилок. Завдяки цьому службові операції, наприклад додавання книг, зміна статусу примірника або підтвердження бронювання, можуть виконувати лише користувачі з відповідними правами.

Відповіді API мають бути уніфікованими. У випадку успішного виконання операції сервер повертає ознаку успіху, повідомлення та дані. У разі помилки відповідь повинна містити пояснення причини та код помилки. Єдина структура відповідей спрощує обробку результатів на клієнтському рівні.

Приклад структури успішної відповіді API:

```
{
  "success": true,
  "message": "Операцію виконано успішно",
  "data": {}
}
```

Приклад структури відповіді у випадку помилки:

```
{
  "success": false,
  "message": "Недостатньо прав для виконання операції",
  "error": "FORBIDDEN"
}
```

Серверний застосунок на NodeJS організовано за модульним принципом. У структурі проєкту передбачаються окремі каталоги для маршрутів, контролерів, сервісів, репозиторіїв, моделей, проміжних обробників і конфігураційних файлів. Такий поділ забезпечує розділення відповідальності між частинами програмного коду та робить систему зручнішою для супроводу.

Логіку проходження запиту через серверну частину можна подати у вигляді послідовності: маршрут приймає HTTP-запит, контролер обробляє параметри, сервіс виконує прикладну логіку, репозиторій звертається до бази даних, після чого результат повертається клієнтській частині. Схему організації REST API та структури NodeJS-проєкту наведено на рис. 2.8.

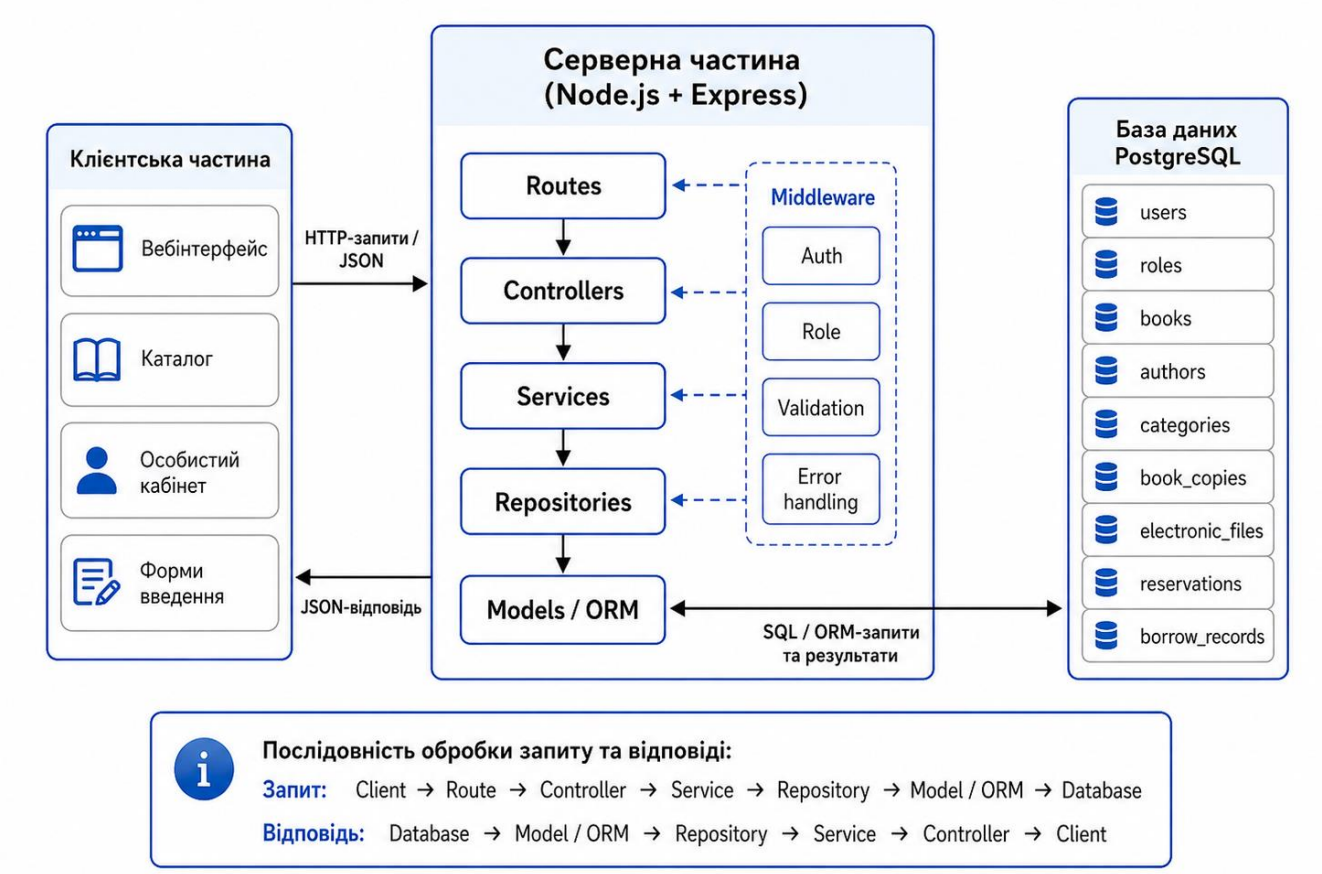

Рисунок 2.8 – Схема організації REST API та структури NodeJS-проєкту

Отже, у межах проєктування REST API та структури NodeJS-проєкту визначено основні групи маршрутів, правила використання HTTP-методів, підхід до захисту серверних операцій і модульну організацію програмного коду. Це створює основу для подальшої реалізації серверної частини інформаційної системи електронної бібліотеки.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЕЛЕКТРОННОЇ БІБЛІОТЕКИ

У третьому розділі розглянуто особливості програмної реалізації інформаційної системи електронної бібліотеки, описано структуру серверної та клієнтської частин веб-застосунку, наведено основні екранні форми користувацького інтерфейсу, а також виконано перевірку функціональних можливостей системи. Основну увагу приділено реалізації REST API, організації взаємодії з базою даних PostgreSQL, обробці користувацьких запитів та перевірці основних сценаріїв роботи читача.

3.1. Реалізація серверної частини та REST API

Серверна частина інформаційної системи електронної бібліотеки реалізована на основі Node.js та Express. Вона виконує роль проміжного рівня між клієнтським інтерфейсом і базою даних PostgreSQL. Основне призначення серверної частини полягає в прийманні HTTP-запитів від клієнта, перевірці їх коректності, виконанні бізнес-логіки та формуванні відповідей у форматі JSON.

Реалізація серверної частини побудована за багаторівневою структурою. Такий підхід зменшує зв’язність між окремими частинами системи та дає змогу розширювати функціональність без істотної зміни вже реалізованих модулів. У структурі серверної частини виділено такі основні рівні:

– маршрути;
– контролери;
– сервіси;
– репозиторії;
– моделі даних;
– проміжне програмне забезпечення.

Маршрути визначають відповідність між URL-адресами REST API та методами контролерів. Контролери приймають запити від клієнта, отримують

параметри, передають їх на рівень сервісів і формують HTTP-відповідь. Сервіси містять основну бізнес-логіку системи [23]. Репозиторії відповідають за доступ до даних, а моделі та ORM-рівень забезпечують зв’язок між об’єктами програмної системи та таблицями бази даних. Загальна схема обробки запиту має такий вигляд, як показано на рис. 3.1.

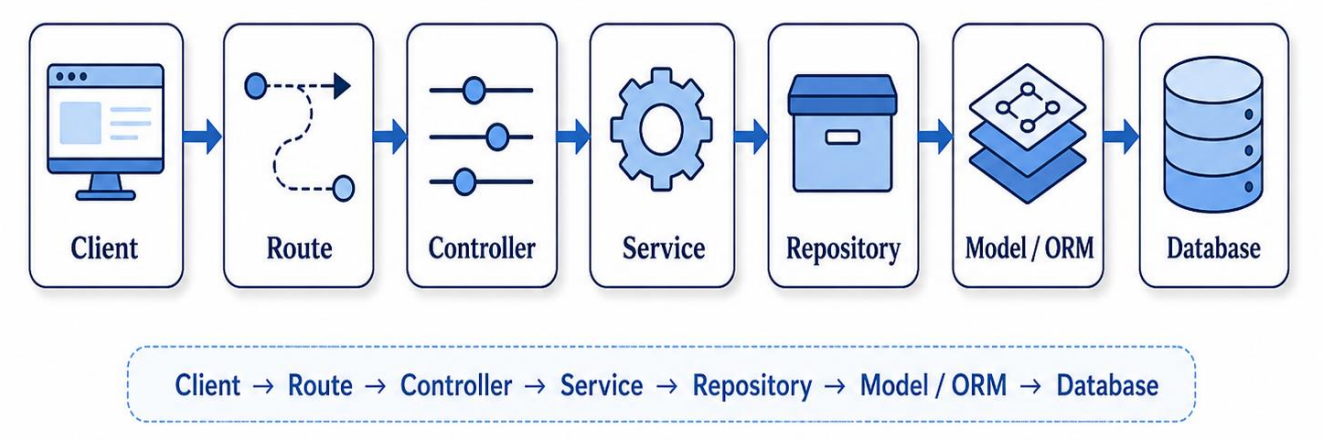

Рисунок 3.1 – Схема обробки запиту в інформаційній системі

Після виконання запиту база даних повертає результат у зворотному напрямку, як продемонстровано на рис. 3.2.

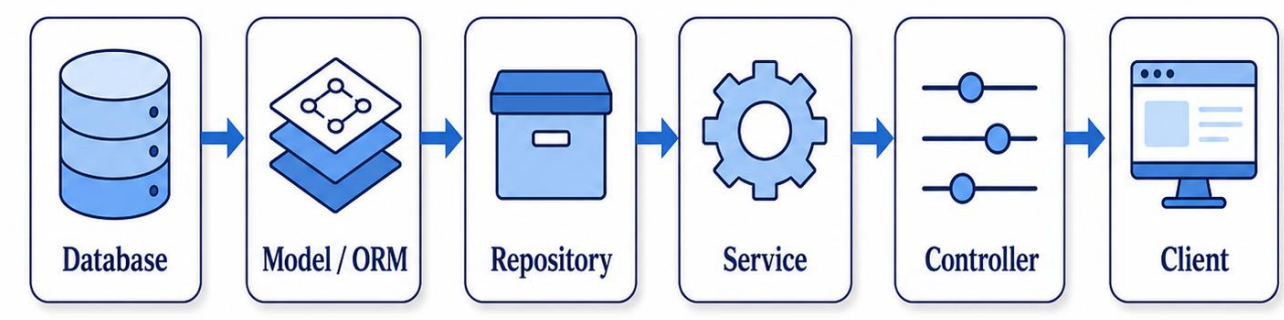

Рисунок 3.2 – Схема формування відповіді в інформаційній системі

Таким чином, клієнтська частина не взаємодіє з базою даних напряму. Усі операції проходять через серверну частину, що дає змогу централізовано контролювати доступ до даних, перевіряти права користувача, обробляти помилки та повертати клієнту уніфіковані JSON-відповіді.

Основні функціональні групи серверної частини наведено в табл. 3.1.

Таблиця 3.1 – Основні модулі серверної частини інформаційної системи

Модуль	**Призначення**
AuthController / AuthService	Реєстрація, авторизація користувача, формування токена доступу
UserController / UserService	Перегляд і редагування профілю, керування даними користувачів
BookController / BookService	Отримання каталогу ресурсів, пошук, фільтрація, перегляд деталей книги
ReservationController / ReservationService	Створення, підтвердження та скасування бронювання
BorrowController / BorrowService	Облік видачі та повернення примірників книг
Middleware	Перевірка авторизації, ролей, валідація даних, обробка помилок
Repository-рівень	Виконання операцій доступу до бази даних
Models / ORM	Відображення таблиць бази даних на програмні моделі

Для організації роботи з користувачами реалізовано модуль авторизації. Він забезпечує реєстрацію нового користувача, перевірку облікових даних під час входу в систему та формування токена доступу. Пароль користувача не зберігається у відкритому вигляді, а перед записом у базу даних перетворюється у хеш. Це підвищує рівень безпеки системи та зменшує ризик розкриття облікових даних у разі несанкціонованого доступу до бази. Фрагмент логіки авторизації представлено у лістингу 3.1.

Лістинг 3.1 – Фрагмент програмного коду авторизації користувача

```
class AuthService {
    async login(email, password) {
        const user = await userRepository.findByEmail(email);
        if (!user) {
            throw new Error("Користувача не знайдено");
        }
        const isPasswordValid = await passwordHelper.compare(
            password,
            user.passwordHash
        );
        if (!isPasswordValid) {
            throw new Error("Невірний пароль");}
```

```
            const token = tokenService.generateToken({
                userId: user.userId,
                role: user.roleName
            });
            return {
                token,
                user: {
                    id: user.userId,
                    fullName: user.fullName,
                    email: user.email,
                    role: user.roleName
                }
            };
        }
    }
```

Цей фрагмент демонструє загальний принцип роботи сервісу авторизації. Спочатку виконується пошук користувача за електронною поштою. Далі перевіряється пароль, після чого формується токен доступу та повертаються основні дані користувача.

Для роботи з каталогом електронної бібліотеки реалізовано модуль книг. Він забезпечує отримання списку ресурсів, пошук за назвою, автором або категорією, перегляд детальної інформації про книгу, а також отримання даних про доступні примірники та електронні файли. Програмна реалізація методу, який входить до сервісу отримання детальної інформації про книгу представлено у лістингу 3.2.

Лістинг 3.2 – Отримання детального опису книги

```
    class BookService {
        async getBookDetails(bookId) {
            const book = await bookRepository.findById(bookId);
            if (!book) {
                throw new Error("Ресурс не знайдено");
            }
            const copies = await
bookCopyRepository.findAvailableByBook(bookId);
            const files = await
electronicFileRepository.findByBook(bookId);
            return {
                ...book,
                availableCopies: copies,
                electronicFiles: files
            };
        }
    }
```

У лістингу 3.2 сервіс не лише отримує основні відомості про книгу, а й доповнює результат інформацією про доступні примірники та електронні файли. Такий підхід дає змогу клієнтській частині отримати повну інформацію одним запитом.

Модуль бронювання відповідає за створення та скасування бронювань. Під час створення бронювання серверна частина перевіряє наявність книги, ідентифікує користувача та створює відповідний запис у базі даних. Якщо користувач не авторизований або ресурс недоступний, система повертає повідомлення про помилку. У лістингу 3.3 наведено фрагмент програмного коду, який реалізує бронювання та скасування броні книги.

Лістинг 3.3 – Модуль бронювання книги

```
class ReservationService {
    async createReservation(userId, bookId) {
        const book = await bookRepository.findById(bookId);
        if (!book) {
            throw new Error("Книгу не знайдено");
        }
        const reservation = {
            userId,
            bookId,
            reservationDate: new Date(),
            expirationDate: calculateExpirationDate(),
            status: "active"
        };
        return await reservationRepository.save(reservation);
    }
}
```

У разі успішного виконання операції клієнт отримує JSON-відповідь із даними створеного бронювання. Якщо виникає помилка, сервер повертає відповідний HTTP-статус і текст повідомлення.

Для уніфікованої обробки помилок використовується middleware. Він перехоплює помилки, що виникають у контролерах або сервісах, і формує стандартну відповідь. Це спрощує роботу клієнтської частини, оскільки всі помилки мають однакову структуру.

REST API системи організовано навколо основних ресурсів предметної області. Основні групи endpoint-ів наведено в табл. 3.2.

Таблиця 3.2 – Основні endpoint-и серверної частини

Метод	**Endpoint**	**Призначення**
POST	/api/auth/register	Реєстрація нового користувача
POST	/api/auth/login	Авторизація користувача
POST	/api/auth/logout	Завершення сеансу роботи
GET	/api/users/profile	Отримання профілю користувача
PUT	/api/users/profile	Оновлення профілю користувача
GET	/api/books	Отримання каталогу ресурсів
GET	/api/books/:id	Перегляд детальної інформації про ресурс
POST	/api/reservations	Створення бронювання
GET	/api/reservations	Отримання списку бронювань користувача
PUT	/api/reservations/:id/cancel	Скасування бронювання
POST	/api/borrow/issue	Реєстрація видачі примірника
POST	/api/borrow/return	Реєстрація повернення примірника
GET	/api/borrow/records	Перегляд історії операцій

Наведений перелік endpoint-ів відображає основні функції системи.

Отже, серверна частина інформаційної системи електронної бібліотеки реалізує основну логіку роботи веб-застосунку. Вона забезпечує обробку запитів, авторизацію користувачів, роботу з каталогом ресурсів, створення бронювань, облік видачі примірників та взаємодію з базою даних PostgreSQL. Використання багаторівневої структури дає змогу відокремити маршрути, бізнес-логіку та доступ до даних, що підвищує зрозумілість, підтримуваність і масштабованість програмного рішення [24].

3.2. Реалізація клієнтської частини та інтерфейсу веб-застосунку

Клієнтська частина інформаційної системи електронної бібліотеки призначена для взаємодії користувача з основними функціями веб-застосунку. Через інтерфейс користувач може виконувати авторизацію, переглядати каталог електронних ресурсів, здійснювати пошук і фільтрацію книг та інші операції, передбачені відповідно до його ролі у системі.

Інтерфейс веб-застосунку побудовано з урахуванням простоти використання та логічної послідовності дій читача. Основна увага приділена тому, щоб користувач міг швидко перейти від пошуку ресурсу до перегляду його опису або створення бронювання. Для цього сторінки системи мають єдину структуру: верхню навігаційну панель, робочу область із вмістом та службові елементи для виконання дій.

Клієнтська частина взаємодіє із серверною частиною через REST API. Після виконання користувачем певної дії, наприклад натискання кнопки пошуку або створення бронювання, веб-інтерфейс формує HTTP-запит до відповідного endpoint. Серверна частина обробляє запит і повертає JSON-відповідь, після чого клієнтська частина оновлює вміст сторінки без прямої взаємодії користувача з базою даних.

Загальна структура клієнтської частини розглядається як сукупність таких основних сторінок:

– сторінка авторизації користувача;
– головна сторінка інформаційної системи;
– сторінка каталогу електронних ресурсів;
– сторінка детальної інформації про книгу;
– особистий кабінет читача;
– сторінка історії бронювань та операцій.

Першим етапом роботи користувача із системою є авторизація. Сторінка авторизації містить форму введення електронної пошти та пароля. Після введення облікових даних користувач надсилає запит на сервер, де виконується перевірка

правильності введеної інформації. У разі успішної авторизації користувач отримує доступ до особистого кабінету та функцій, що потребують ідентифікації. На рис. 3.3 показано сторінку авторизації користувача інформаційної системи електронної бібліотеки.

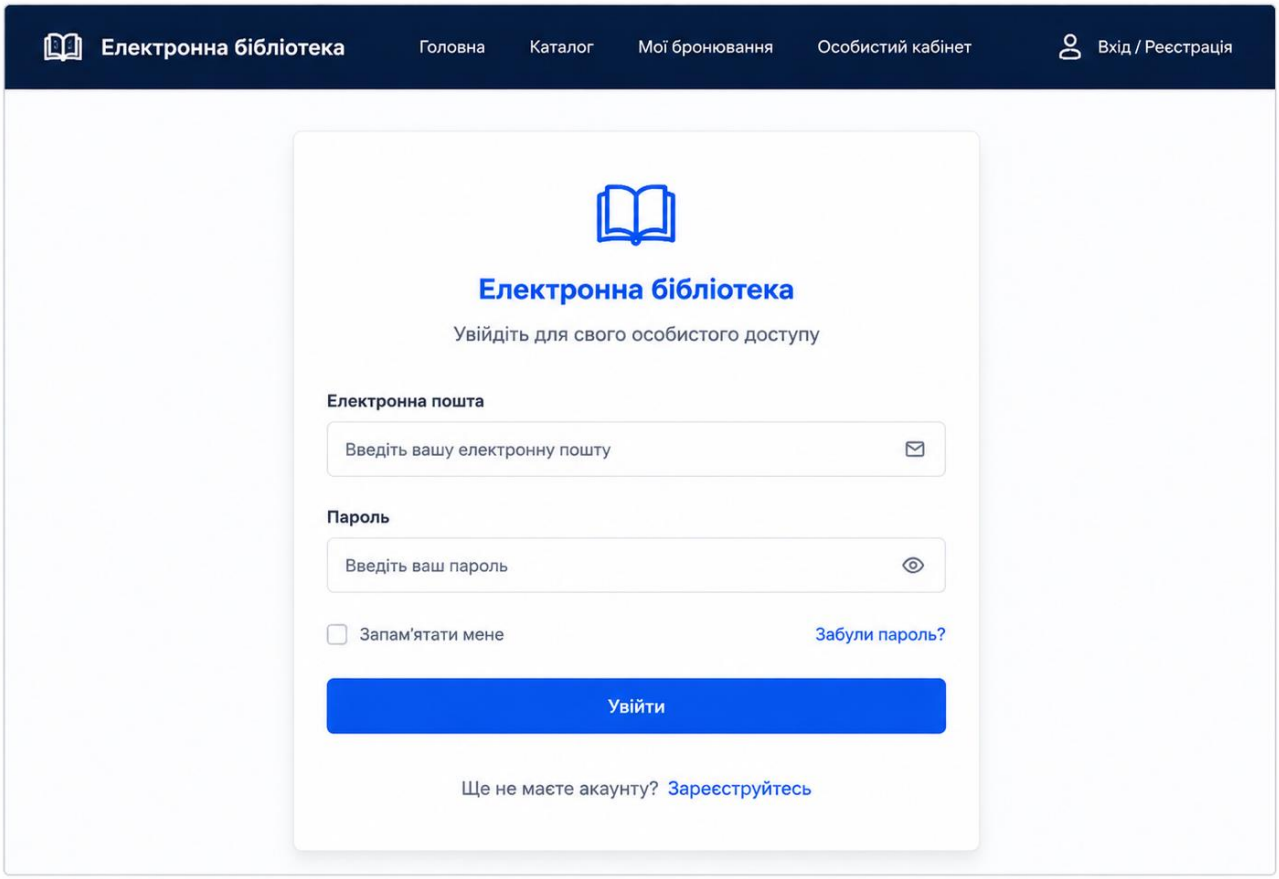

Рисунок 3.3 – Сторінка авторизації користувача

На сторінці авторизації передбачено повідомлення про помилки. Наприклад, якщо користувач вводить неправильний пароль або неіснуючу електронну адресу, система відображає зрозуміле повідомлення і не надає доступ до захищених функцій. Це підвищує зручність користування системою та зменшує кількість помилкових дій.

Після входу в систему користувач переходить на головну сторінку. Вона виконує роль початкового екрана, з якого можна перейти до каталогу ресурсів, особистого кабінету, історії бронювань або форми пошуку. Головна сторінка містить коротку інформацію про призначення системи, елементи навігації та

швидкий доступ до основних функцій. На рис. 3.4 представлено головну сторінку розробленої інформаційної системи.

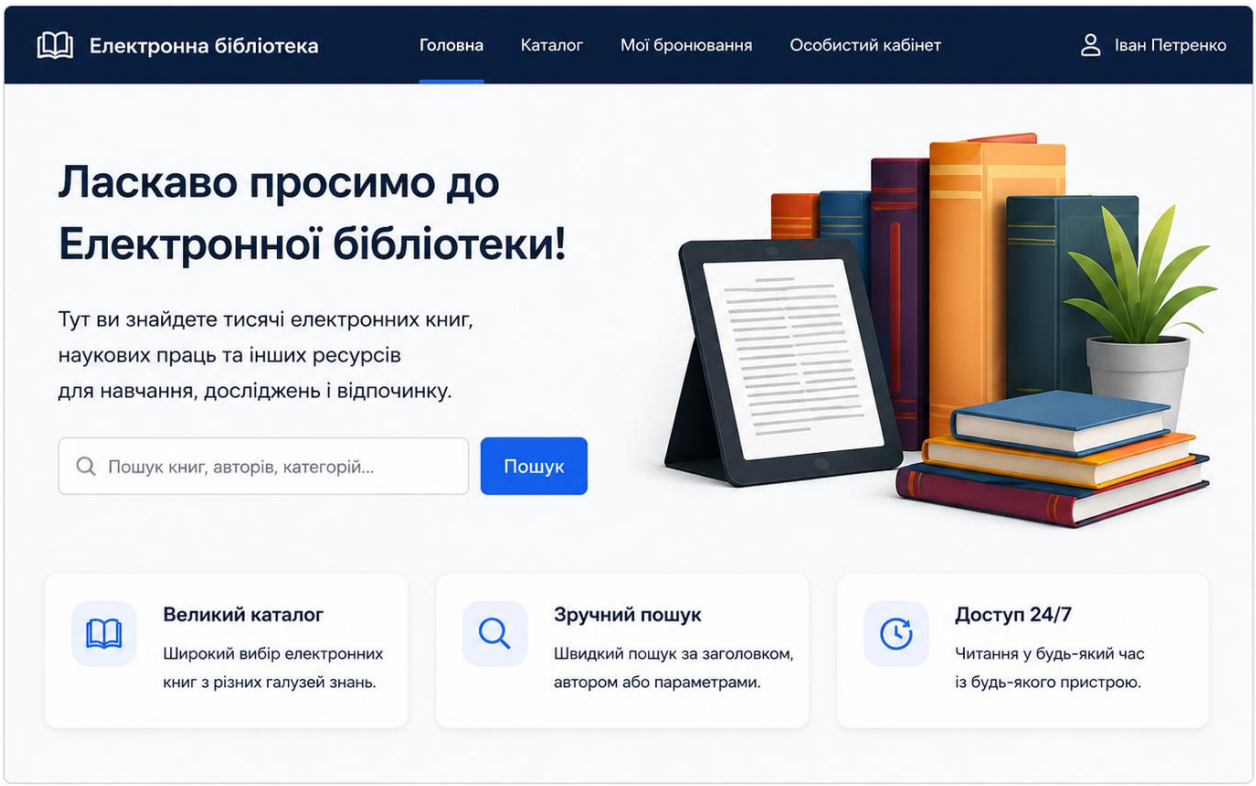

Рисунок 3.4 – Головна сторінка інформаційної системи електронної бібліотеки

Основною сторінкою для читача є каталог електронних ресурсів, який проілюстрований на рис. 3.5. На цій сторінці відображається перелік книг із короткими відомостями: назвою, автором, категорією, роком видання та статусом доступності. Для зручності роботи реалізовано пошук і фільтрацію ресурсів. Користувач може знайти книгу за назвою, автором або категорією, а також звузити перелік ресурсів за додатковими параметрами.

Каталог ресурсів є центральним елементом клієнтської частини, оскільки саме з нього починається більшість користувацьких сценаріїв. Для кожного ресурсу передбачено кнопку переходу до детальної інформації. Це дає змогу не перевантажувати каталог надмірними даними, залишаючи на ньому лише короткий опис ресурсу.

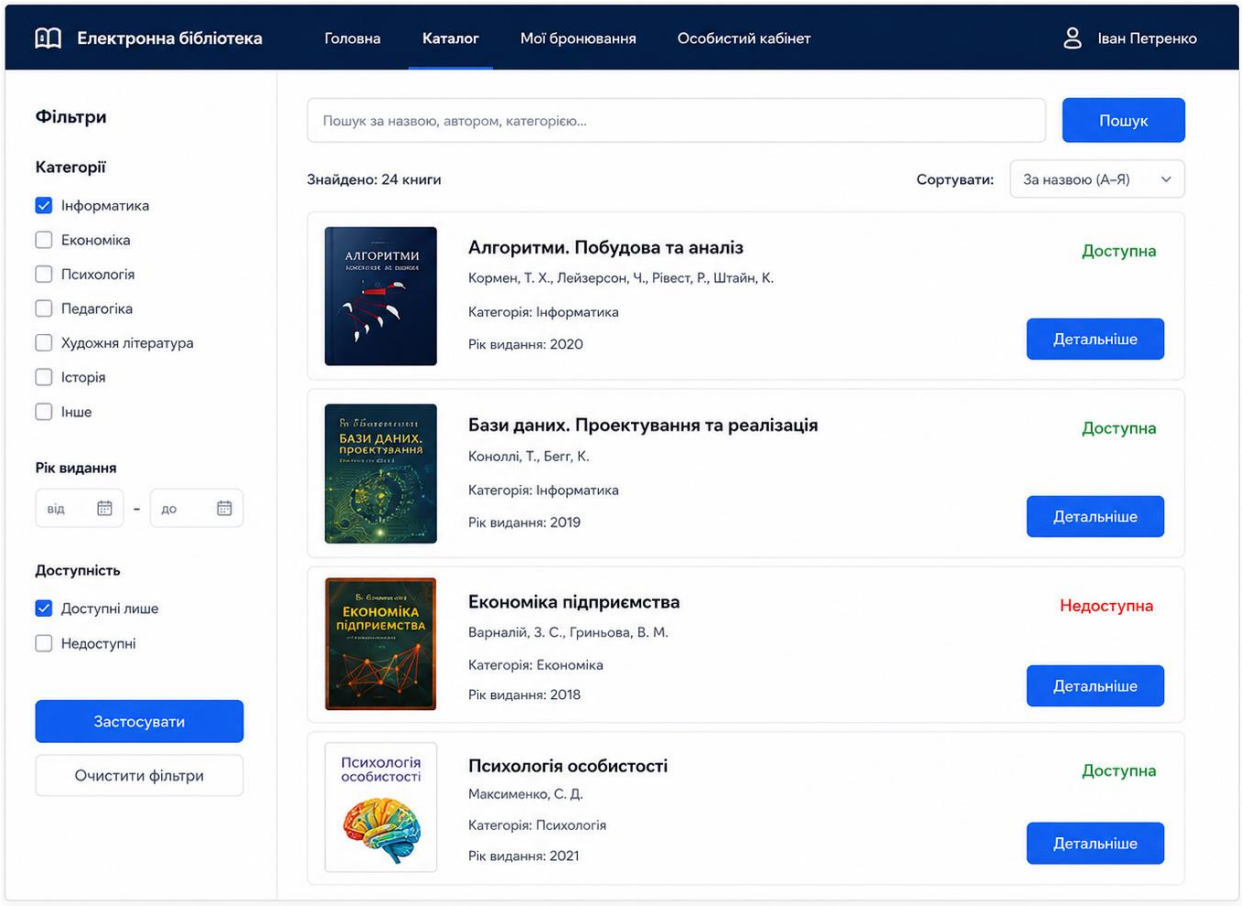

Рисунок 3.5 – Сторінка каталогу електронних ресурсів

Сторінка детальної інформації про ресурс містить повні відомості про книгу. На ній відображаються назва, автори, категорії, опис, рік видання, видавництво, мова, доступні електронні файли та інформація про наявність примірників. Якщо користувач авторизований, йому доступні додаткові дії: перегляд електронного файлу або створення бронювання. Сторінку з детальною інформацією про книгу представлено на рис. 3.6.

Особливістю цієї сторінки є те, що вона поєднує інформаційну та функціональну частини. З одного боку, користувач отримує повний опис ресурсу, а з іншого – може одразу виконати дію, пов’язану з цим ресурсом. Якщо електронний файл має обмежений рівень доступу, система перевіряє права користувача перед його відкриттям.

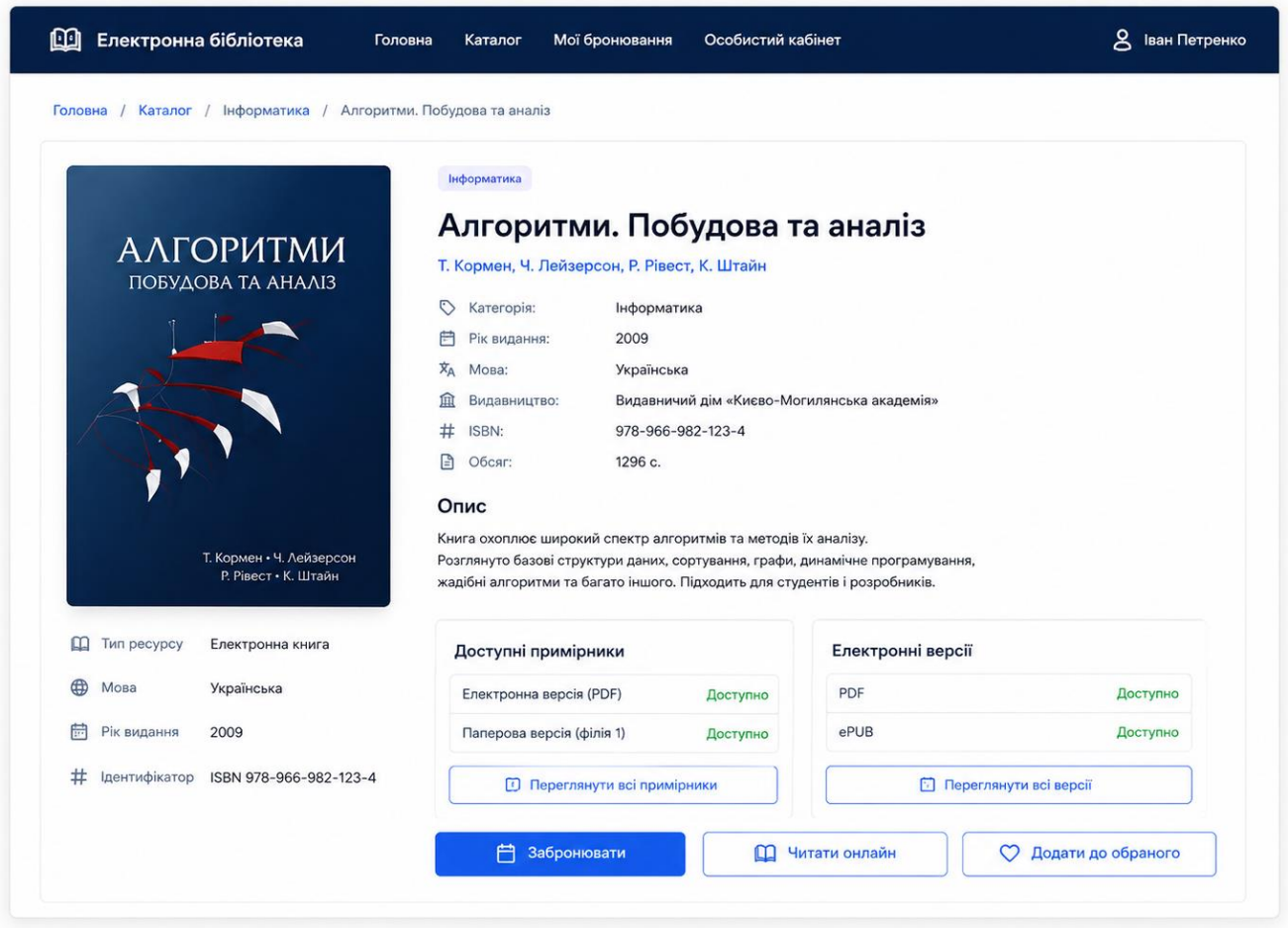

Рисунок 3.6 – Сторінка детальної інформації про електронний ресурс

Особистий кабінет читача призначений для перегляду та редагування персональних даних користувача. На цій сторінці відображається повне ім’я, електронна пошта, номер телефону, статус облікового запису та роль користувача. Також у кабінеті передбачено швидкі посилання на активні бронювання та історію операцій. На рис. 3.7 продемонстровано вигляд особистого кабінету читача.

Реалізація особистого кабінету дає змогу користувачу контролювати власні дані та переглядати дії, виконані в системі. Редагування профілю виконується через окрему форму. Після внесення змін клієнтська частина надсилає запит до серверної частини, де оновлені дані перевіряються та зберігаються у базі даних [24].

Для контролю виконаних операцій у системі передбачено сторінку історії бронювань та операцій. На ній користувач може переглянути активні та завершені бронювання, статуси заявок, дати створення, строки дії бронювання, а також інформацію про видачу та повернення примірників.

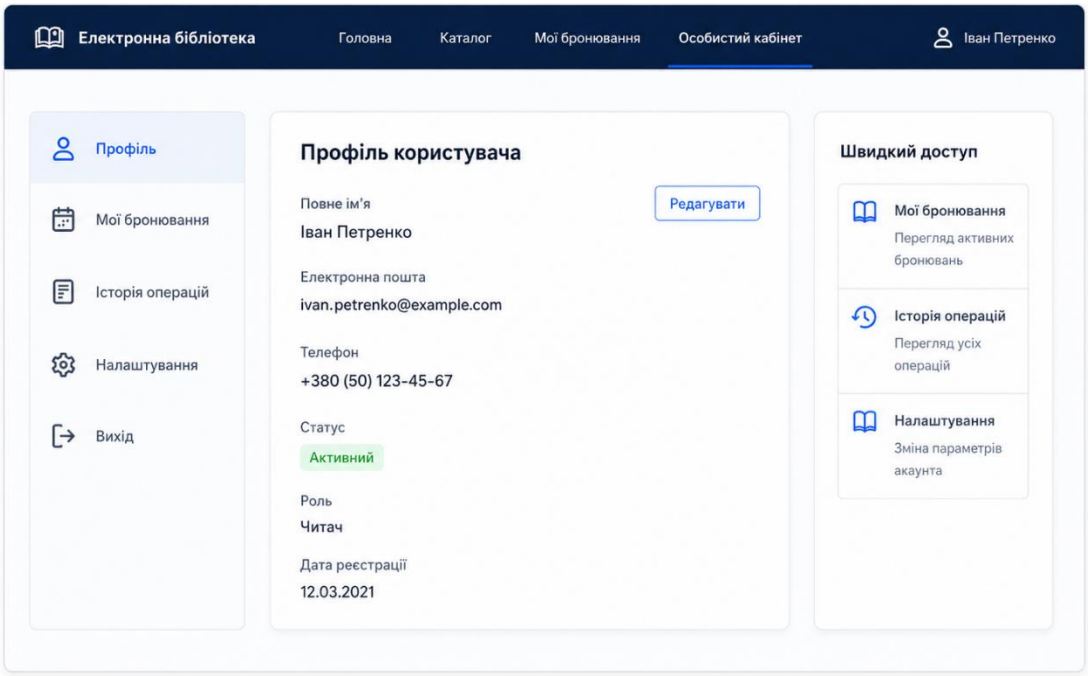

Рисунок 3.7 – Особистий кабінет читача

Сторінка історії операцій є важливою для прозорості роботи системи і показана на рис. 3.8. Вона дає змогу користувачу бачити, які ресурси були заброньовані, які операції вже завершено, а які ще перебувають в активному стані. Це зменшує потребу у зверненні до адміністратора та підвищує самостійність роботи читача із системою.

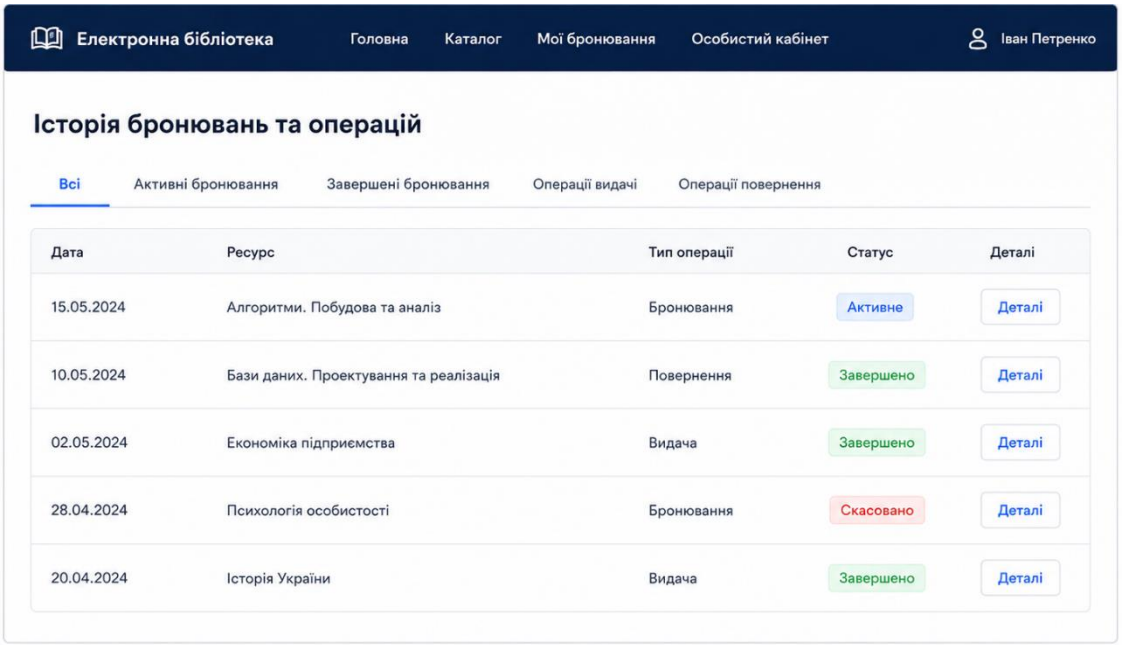

Рисунок 3.8 – Сторінка історії бронювань та операцій

Для адміністратора або бібліотекара передбачена окрема сторінка керування ресурсами. Через неї можна додавати нові книги, редагувати дані про авторів і категорії, завантажувати електронні файли, змінювати статуси примірників та підтверджувати бронювання. На рис. 3.9 показано інтерфейс адміністратора та бібліотекара.

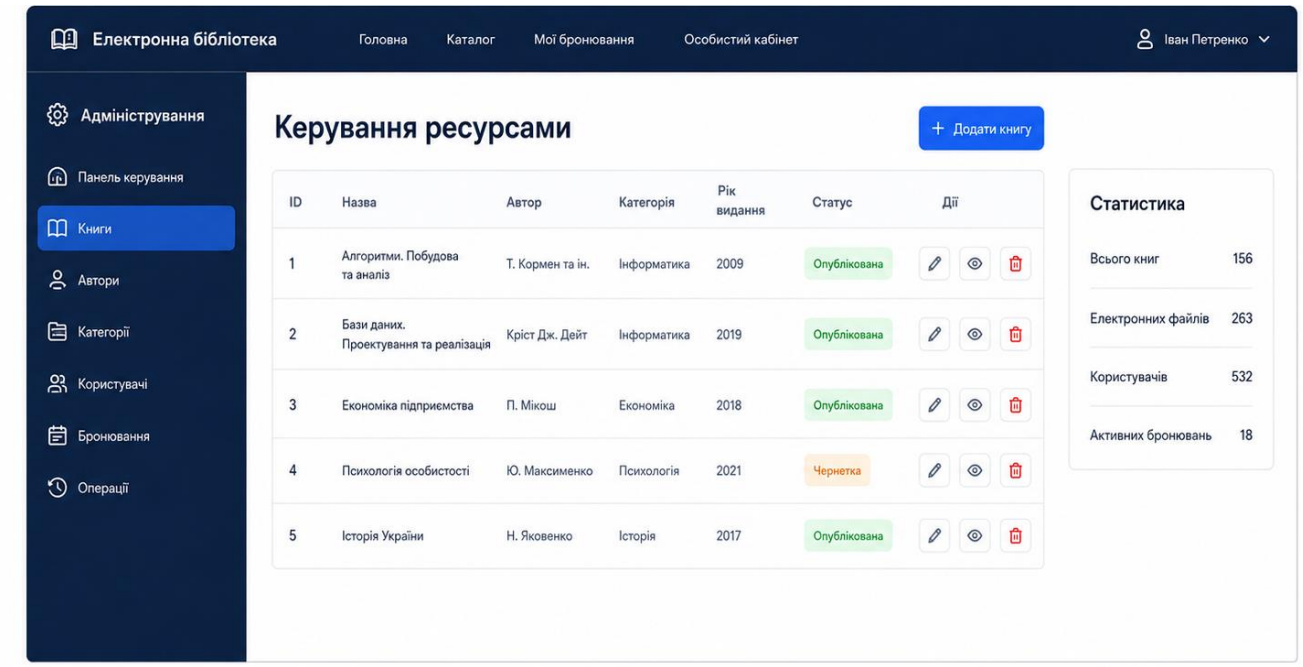

Рисунок 3.9 – Сторінка керування ресурсами електронної бібліотеки

Клієнтська частина повинна також забезпечувати базову перевірку введених даних. Наприклад, під час авторизації перевіряється заповнення обов’язкових полів, під час редагування профілю – коректність електронної пошти та номера телефону, а під час створення бронювання – наявність вибраного ресурсу. Первинна перевірка на рівні інтерфейсу зменшує кількість некоректних запитів до сервера, однак не замінює серверну валідацію.

З погляду зручності використання інтерфейс побудований так, що основні дії доступні за мінімальну кількість переходів. Для цього в навігаційній панелі розміщено посилання на каталог, особистий кабінет, історію операцій та вихід із системи. Пошуковий рядок зроблено доступним безпосередньо на сторінці каталогу, оскільки пошук є однією з найбільш частих дій користувача.

Основні елементи інтерфейсу веб-застосунку наведено в таблиці 3.3.

Таблиця 3.3 – Основні елементи клієнтського інтерфейсу

Елемент інтерфейсу	**Призначення**
Форма авторизації	Введення електронної пошти та пароля користувача
Навігаційна панель	Перехід між основними сторінками системи
Каталог ресурсів	Перегляд переліку книг та електронних матеріалів
Пошуковий рядок	Пошук ресурсу за назвою, автором або ключовими словами
Фільтри	Відбір ресурсів за категорією, роком видання або доступністю
Картка ресурсу	Коротке представлення книги в каталозі
Сторінка деталей	Відображення повної інформації про ресурс
Кнопка бронювання	Створення заявки на резервування ресурсу
Особистий кабінет	Перегляд і редагування даних користувача
Історія операцій	Перегляд бронювань, видач і повернень

У результаті реалізації клієнтської частини користувач отримує зручний веб-інтерфейс для роботи з ресурсами електронної бібліотеки. Інтерфейс забезпечує доступ до основних функцій системи, приховує технічну складність взаємодії із сервером і базою даних та подає дані у зрозумілому для користувача вигляді. Такий підхід дає змогу організувати повний цикл роботи читача: від пошуку ресурсу до перегляду інформації, створення бронювання та контролю власної історії операцій.

3.3. Тестування функціональних можливостей системи

Тестування інформаційної системи електронної бібліотеки виконано з метою перевірки коректності роботи основних функцій веб-застосунку, взаємодії клієнтської та серверної частин, а також правильності збереження й отримання даних із бази даних PostgreSQL [25]. Основну увагу приділено перевірці сценаріїв, які найчастіше використовуються читачем і бібліотекарем.

Під час тестування перевірялися такі функціональні можливості системи:

– авторизація користувача;
– перегляд каталогу електронних ресурсів;
– пошук і фільтрація книг;
– відкриття сторінки детальної інформації про ресурс;
– створення бронювання;
– перегляд історії бронювань та операцій;
– редагування профілю користувача;
– обробка помилкових дій користувача.

Тестування проводилося за методом функціонального тестування. Для кожної функції було сформовано вхідні дані, очікуваний результат і фактичний результат виконання. Такий підхід дає змогу перевірити, чи відповідає реалізована система поставленим вимогам. Результати проведеного тестування наведено у табл. 3.4.

Таблиця 3.4 – Результати тестування функціональних можливостей системи

№	Тестовий сценарій	Вхідні дані	Очікуваний результат	Фактичний результат
1	**2**	**3**	**4**	**5**
1	Авторизація користувача	Коректні email і пароль	Користувач входить у систему	Виконано
2	Авторизація з неправильним паролем	Коректний email, неправильний пароль	Виведення повідомлення про помилку	Виконано
3	Перегляд каталогу ресурсів	Перехід на сторінку каталогу	Відображення списку книг	Виконано
4	Пошук книги за назвою	Назва книги у пошуковому полі	Відображення відповідних результатів	Виконано
5	Фільтрація за категорією	Вибрана категорія ресурсу	Відображення книг вибраної категорії	Виконано
6	Перегляд деталей книги	Натискання кнопки перегляду	Відкриття сторінки з повною інформацією	Виконано

Продовження таблиці 3.4

1	2	3	4	5
7	Створення бронювання	Авторизований користувач, доступна книга	Створення запису бронювання	Виконано
8	Бронювання без авторизації	Неавторизований користувач	Запит на авторизацію	Виконано
9	Перегляд історії операцій	Авторизований користувач	Відображення списку бронювань і видач	Виконано
10	Редагування профілю	Нові дані користувача	Оновлення інформації у профілі	Виконано

Для перевірки авторизації було протестовано два основні випадки: успішний вхід користувача та спроба входу з неправильними обліковими даними. У першому випадку система надає доступ до особистого кабінету, у другому – відображає повідомлення про помилку. Це підтверджує правильність роботи механізму перевірки користувача.

Окремо перевірено роботу каталогу електронних ресурсів. Під час переходу на сторінку каталогу система коректно отримує дані із серверної частини та відображає список книг. Пошук і фільтрація дають змогу швидко знайти потрібний ресурс за назвою, автором або категорією.

Також було протестовано сценарій створення бронювання. Якщо користувач авторизований і ресурс доступний, система створює новий запис бронювання та відображає його в історії операцій. Якщо користувач не виконав вхід у систему, веб-застосунок не дозволяє створити бронювання і пропонує пройти авторизацію.

Для перевірки коректності роботи серверної частини тестувалися відповіді REST API [26-28]. У разі успішного виконання операції сервер повертає відповідь зі статусом успіху та необхідними даними. У разі помилки формується відповідь із повідомленням, яке може бути відображене у клієнтському інтерфейсі. У табл. 3.5 наведено приклад перевірки REST API.

Таблиця 3.5 – Приклади перевірки REST API

Метод	**Endpoint**	**Очікувана відповідь**
POST	/api/auth/login	Дані користувача та токен доступу
GET	/api/books	Список електронних ресурсів
GET	/api/books/:id	Детальна інформація про книгу
POST	/api/reservations	Дані створеного бронювання
GET	/api/reservations	Список бронювань користувача
PUT	/api/users/profile	Оновлені дані профілю

За результатами тестування встановлено, що основні функціональні можливості інформаційної системи електронної бібліотеки працюють коректно.

Таким чином, у третьому розділі було розглянуто практичну реалізацію інформаційної системи електронної бібліотеки. Описано побудову серверної частини, організацію REST API, принципи взаємодії з базою даних PostgreSQL та особливості клієнтського інтерфейсу. Також наведено основні екранні форми, які демонструють роботу користувача із системою.

Проведене тестування підтвердило коректність виконання основних сценаріїв. Отримані результати свідчать, що розроблена система є функціонально завершеною, зручною для користувача та придатною для використання в умовах електронної бібліотеки.

4 БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

У даному розділі розглянуто питання безпеки життєдіяльності та основ охорони праці з урахуванням впливу трудової діяльності на функціональний стан людини. Охарактеризовано роль центральної нервової системи у регуляції діяльності організму, сприйнятті та обробці інформації, формуванні адаптаційних реакцій і забезпеченні працездатності. Також проаналізовано шляхи збереження працездатності та підвищення продуктивності праці, зокрема раціональну організацію режиму праці й відпочинку, оптимізацію трудових навантажень, покращення умов праці та зменшення впливу несприятливих виробничих чинників.

4.1. Роль центральної нервової системи в трудовій діяльності людини

Нервова система має найголовніше значення в організмі людини. Вона координує, регулює роботу всіх внутрішніх органів і здійснює зв'язок організму із зовнішнім середовищем [29].

Нервова система людини складається із центральної (ЦНС), яка включає головний і спинний мозок і периферійної (ПНС), яка складається з нервових волокон, що відходять від головного і спинного мозку.

За функціями нервову систему поділяють на соматичну і вегетативну. Соматична нервова система регулює опорно-руховий апарат і всі органи чуття, а вегетативна - процес обміну речовин та роботу всіх внутрішніх органів (серця, нирок, легенів). Найпростіші рухи регулює спинний мозок. Довгастий мозок керує процесами травлення, дихання, кровообігу та іншими життєво важливими функціями. Підкіркова і кіркова частини головного мозку керують усією психічною діяльністю людини [29].

Центральна нервова система виконує рефлекторну, інтегративну та координаційну функції.

Рефлекторна діяльність мозку зумовлена безумовними та умовними рефлексами. Безумовні рефлекси є вродженими, мають велику стійкість і забезпечують пристосування організму до зовнішнього середовища. Умовні рефлекси набуваються залежно від обставин, розширюють діапазон пристосувальницьких можливостей організму і згасають, якщо потреби в них немає.

Стійка і злагоджена система умовних рефлексів формується у процесі навчання і забезпечує виконання певного виробничого завдання. Стійкість системи умовних рефлексів може бути порушена при відхиленні трудової діяльності від програми, а надійність - під впливом несприятливих виробничих чинників. Такі порушення, якщо не вжити належних заходів, можуть призвести до зниження працездатності, травм або нещасних випадків [29].

Виконуючи інтегративну функцію, ЦНС забезпечує злагоджену взаємодію всіх органів і систем організму, підтримує його стійкий внутрішній стан. Несприятливі умови праці можуть призвести до стомлення нервової системи, що послаблює її інтегративну функцію і може спровокувати розлад ряду фізіологічних систем: серцево-судинної, шлунково-кишкової, дихальної тощо або призвести до різних захворювань (інфаркти, інсульти, виразкові хвороби).

Завдяки координаційній функції ЦНС здійснює підпорядкування багатьох рефлексів одному, який має на даний час найважливіше значення для організму.

Усі функції центральної нервової системи реалізуються в кожній конкретній реакції організму, забезпечуючи ефект найбільшого пристосування до мінливих умов зовнішнього середовища і підвищуючи фізіологічну опірність організму шкідливим зовнішнім впливам.

Вища нервова діяльність людини заснована на функціях двох сигнальних систем. Анатомічною основою першої сигнальної системи є аналізатори (зоровий, слуховий). Аналізатор - це система нервових клітин, які сприймають і переробляють інформацію, що надходить до них із зовнішнього та внутрішнього середовища організму.

Анатомічною основою другої сигнальної системи, яка властива тільки людині, є мовно-руховий апарат, тісно пов'язаний із зоровим та слуховим аналізаторами, а її подразником є слово. Мова, в усіх її видах, являє собою найбагатше джерело подразників. За допомогою слова передаються сигнали про конкретні подразники, і в цьому випадку слово служить принциповим подразником - сигналом сигналів, є пусковим механізмом дій і вчинків людей. Мова підвищує здатність мозку відображати дійсність, забезпечує аналіз і синтез, абстрактне мислення, створює можливість для спілкування, використання і передачі життєвого досвіду, досягнень культури і мистецтва. Але в деяких випадках слово може бути негативним подразником і може призвести до розладів нервової системи, порушень функціонування всіх систем організму і, таким чином, стати небезпечним виробничим фактором.

Центральна нервова система бере участь у прийманні, обробці та аналізі будь-якої інформації, що надходить із зовнішнього і внутрішнього середовищ. При виникненні перенавантажень на організм людини нервова система визначає ступінь їхнього впливу і формує адаптаційно-захисну реакцію [29].

4.2. Шляхи збереження працездатності та підвищення продуктивності праці на будівництві

Розумова праця об'єднує роботи пов'язані зі сприйняттям та опрацюванням інформації, необхідністю переважного навантаження сенсорного апарату, уваги, пам'яті, а також активації процесів мислення, емоційної сфери.

Виділяють такі різновиди розумової праці:

– операторська;
– адміністративно-керівна;
– творча;
– праця викладачів і медичних працівників;
– праця учнів і студентів.

Вказані види роботи відрізняються щодо організації трудового процесу, рівномірності навантаження, ступеню емоційного напруження. При розумовій діяльності загострюється сприйняття, увага, пам'ять. Посилюється кровопостачання мозку, підвищується енергетичний обмін нервових клітин, змінюються показники біоелектричної активності мозку.

При інтенсивній інтелектуальній діяльності споживання кисню 100 г кори головного мозку в 5-6 разів більше, ніж споживання скелетного м'язу такої ж ваги при максимальному навантаженні. Розумова праця, а особливо, робота оператора супроводжується деякою нервово-емоційною напругою. Вона призводить до посилення серцево-судинної діяльності, дихання, енергообміну, підвищення м'язового тонусу [29-31].

Після закінчення розумової праці "робоча домінанта" повністю не згасає, зумовлюючи більш тривале втомлення та виснаження ЦНС при розумовій праці, ніж при фізичній.

Практичне значення заходів щодо підвищення працездатності випливає із закономірностей її динаміки і зводиться до:

– збільшення фази стійкого стану в фонді робочого часу;
– прискорення процесу опрацювання;
– віддалення фази розвитку втоми;
– забезпечення високої продуктивності праці за нормальних фізіологічних затрат.

Комплекс заходів щодо підвищення і збереження працездатності працівників на оптимальному рівні реалізується на техніко-організаційному, соціально-економічному, санітарно-гігієнічному, медико-біологічному, психологічному напрямках.

Могутнім фактором високої працездатності і продуктивності праці є оптимізація трудових навантажень на основі механізації і автоматизації виробничих процесів, удосконалення технології, скорочення і ліквідації важкої ручної праці. Доведено, що при правильній організації праці на легких роботах

спостерігається найбільша тривалість фази стійкого стану, а на важких роботах вона нетривала [30].

Високий рівень працездатності безпосередньо залежить від умов праці, оскільки поліпшення їх супроводжується зменшенням енергетичних затрат організму на подолання несприятливого впливу факторів виробничого середовища.

Важливим напрямком підвищення працездатності працюючих є ритмізація трудових процесів, оптимізація темпу роботи, а також раціоналізація трудових рухів на фізіологічній основі, що сприяє формуванню і закріпленню робочих динамічних стереотипів, а отже зменшенню м'язових і вольових зусиль. Ритмічна робота підвищує функціональні можливості організму, сприяє його тренованості і забезпечує економізацію енергетичних затрат.

Економізація функціональних затрат досягається завдяки стійкій домінанті і автоматизму дій, що виключає зайві рухи, розсіювання уваги тощо.

Особливе значення для підтримання працездатності працівників на високому рівні має раціональний режим праці і відпочинку.

Дослідження показують, що впровадження раціонального режиму праці і відпочинку на підприємствах забезпечує підвищення продуктивності праці на 8 – 10%, сприяє поліпшенню фізіологічного стану працівників (зменшується частота пульсу в процесі роботи, підвищується м'язова витривалість в кінці зміни, покращується координація рухів) [31].

Високій працездатності працівників сприяє і раціоналізація робочих місць на основі врахування антропометричних, біомеханічних і психофізіологічних вимог, що обумовлює раціональну робочу позу, зменшення статичних навантажень, оптимізацію робочої зони та інформаційних потоків.

Висока працездатність забезпечується за рахунок використання факторів естетичного впливу на працюючих. Такими факторами є колір, світло, музика. Слід підкреслити значення функціональної музики, яка впливає на емоційну сферу людини, підвищує збудливість і лабільність центральної нервової системи. На початку роботи вона прискорює процес впрацювання, а в кінці робочого дня зменшує суб'єктивне відчуття стомленості.

Вплив функціональної музики посилюється, якщо вона поєднується з фізичними вправами. Останні підвищують лабільність органів, які безпосередньо беруть участь у виконанні роботи, активізують роботу органів дихання і кровообігу.

Особливе значення в підвищенні працездатності працівників має створення сприятливого соціально-психологічного клімату в організації, високий рівень мотивації праці, ефективна система стимулювання результатів діяльності, рівень життя в цілому і охорона здоров'я населення [30].

Ефективність заходів, спрямованих на підвищення працездатності працівників, можна оцінити приростом продуктивності праці, який досягається за рахунок збільшення фази стійкого стану в загальній тривалості робочої зміни.

ВИСНОВКИ

У кваліфікаційній роботі розроблено інформаційну систему електронної бібліотеки з використанням NodeJS. Система призначена для автоматизації основних процесів роботи з бібліотечними ресурсами, зокрема обліку книг, керування користувачами, пошуку матеріалів, роботи з електронними файлами, бронювання ресурсів та обліку операцій видачі й повернення.

У першому розділі проаналізовано предметну область електронної бібліотеки з точки зору програмної інформаційної системи. Визначено основні процеси її функціонування, ролі користувачів, вимоги до даних, безпеки, надійності та серверної частини. Обґрунтовано доцільність використання веб-орієнтованого підходу та платформи NodeJS для реалізації серверної логіки.

У другому розділі виконано проєктування інформаційної системи електронної бібліотеки. Розроблено клієнт-серверну архітектуру, діаграми варіантів використання, діаграми класів, структуру програмних модулів і модель бази даних. Запропоновано реляційну структуру з таблицями, які описують сутності предметної області та визначено принципи побудови REST API і модульної структури NodeJS-проєкту.

У третьому розділі описано практичну реалізацію серверної та клієнтської частин веб-застосунку. Серверна частина реалізує обробку HTTP-запитів, авторизацію користувачів, роботу з каталогом ресурсів, створення бронювань, облік операцій і взаємодію з базою даних PostgreSQL. Клієнтський інтерфейс забезпечує авторизацію, перегляд каталогу, пошук і фільтрацію книг, перегляд детальної інформації про ресурс, роботу з особистим кабінетом та історією операцій.

Проведене функціональне тестування підтвердило коректність виконання основних сценаріїв роботи системи. Результати тестування показали, що реалізовані функції відповідають визначеним вимогам.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Sommerville I. Software Engineering. 10th ed. Boston: Pearson Education. 2016. 816 p.

2. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner’s Approach. 9th ed. New York: McGraw-Hill Education. 2020. 672 p.

3. Brown E. Web Development with Node and Express: Leveraging the JavaScript Stack. 2nd ed. Sebastopol: O’Reilly Media. 2019.

4. Casciaro M., Mammino L. Node.js Design Patterns: Design and implement production-grade Node.js applications using proven patterns and techniques. 3rd ed. Birmingham: Packt Publishing. 2020.

5. Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. 7th ed. New York: McGraw-Hill Education. 2019.

6. Бородкіна І. Л., Бородкін Г. О. Інженерія програмного забезпечення: посібник для студентів вищих навчальних закладів. Київ: Центр учбової літератури. 2020. 204 с.

7. Левус Є. В., Мельник Н. Б. Вступ до інженерії програмного забезпечення: навчальний посібник. Львів: Видавництво Львівської політехніки. 2018. 248 с.

8. Левус Є. В., Марусенкова Т. А. Вступ до інженерії програмного забезпечення: навчальний посібник. Львів: Видавництво Львівської політехніки. 2021. 188 с.

9. Петрик М. Р. Лабораторний практикум з розділу «Шаблони проєктування» дисципліни «Архітектура та проектування програмного забезпечення»: навчальний посібник. Тернопіль: ТНТУ імені Івана Пулюя. 2016. 36 с.

10. ДСТУ ISO/IEC/IEEE 12207:2018 Інженерія систем і програмних засобів. Процеси життєвого циклу програмних засобів.

11. ISO/IEC/IEEE 29148:2018 Systems and software engineering — Life cycle processes — Requirements engineering.

12. ISO/IEC/IEEE 12207:2017 Systems and software engineering — Software life cycle processes.

13. IEEE Std 1012-2016 IEEE Standard for System, Software, and Hardware Verification and Validation.

14. Souppaya M., Scarfone K., Dodson D. Secure Software Development Framework (SSDF) Version 1.1: Recommendations for Mitigating the Risk of Software Vulnerabilities. NIST Special Publication 800-218. 2022.

15. Singh P. Design and implementation of a virtual library system: An overview. IP Indian Journal of Library Science and Information Technology. 2024.

16. Microsoft. Best practices for RESTful web API design. URL: https://learn.microsoft.com/en-us/azure/architecture/best-practices/api-design (дата звернення: 07.06.2026 р.).

17. Node.js Documentation. URL: https://nodejs.org/docs/latest/api/ (дата звернення: 07.06.2026 р.).

18. Introduction to Node.js. URL: https://nodejs.org/learn/getting-started/introduction-to-nodejs (дата звернення: 08.06.2026 р.).

19. Express.js. Express – Node.js web application framework. URL: https://expressjs.com/ (дата звернення: 08.06.2026 р.).

20. Express package documentation. URL: https://www.npmjs.com/package/express (дата звернення: 11.06.2026 р.).

21. PostgreSQL Global Development Group. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/ (дата звернення: 11.06.2026 р.).

22. PostgreSQL Global Development Group. PostgreSQL 16 Documentation. URL: https://www.postgresql.org/docs/16/ (дата звернення: 11.06.2026 р.).

23. Prisma. Prisma ORM Documentation. URL: https://www.prisma.io/docs (дата звернення: 12.06.2026 р.).

24. MDN Web Docs. Overview of HTTP. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/Overview (дата звернення: 13.06.2026 р.).

25. MDN Web Docs. HTTP request methods. URL: https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Methods (дата звернення: 14.06.2026 р.).

26. OWASP Foundation. OWASP Top 10:2025. URL: https://owasp.org/Top10/2025/en/ (дата звернення: 14.06.2026 р.).

27. OWASP Foundation. OWASP Cheat Sheet Series. URL: https://cheatsheetseries.owasp.org/ (дата звернення: 14.06.2026 р.).

28. Bootstrap Team. Bootstrap Documentation. URL: https://getbootstrap.com/docs/5.3/getting-started/introduction/ (дата звернення: 15.06.2026 р.).

29. Катренко Л.А., Катренко А.В. Охорона праці в галузі комп'ютинґу. Львів: Магнолія-2006, 2012. 544 с.

30. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Київ. 2018. URL: https://zakon.rada.gov.ua/go/z0508-18 (дата звернення: 15.06.2026 р.).

31. Методичні вказівки для написання розділу «Безпека життєдіяльності, основи охорони праці» в кваліфікаційних роботах здобувачів освітнього рівня «бакалавр». Для студентів всіх форм навчання, рівень вищої освіти перший (бакалаврський) / укл.: О. Я. Гурик, І. Б. Окіпний. Тернопіль: ТНТУ імені Івана Пулюя. 2021. 20 с.

ДОДАТКИ

Додаток А – Деталізований перелік ендпоінтів REST API

Таблиця А.1 – Ендпоінти модуля авторизації

Метод	Ендпоінт	Призначення
POST	`/api/auth/register`	Реєстрація нового користувача
POST	`/api/auth/login`	Авторизація користувача в системі
POST	`/api/auth/logout`	Завершення сеансу користувача
GET	`/api/auth/me`	Отримання інформації про поточного користувача

Таблиця А.2 – Ендпоінти модуля користувачів

Метод	Ендпоінт	Призначення
GET	`/api/users`	Отримання списку користувачів
GET	`/api/users/:id`	Отримання інформації про конкретного користувача
GET	`/api/users/profile`	Отримання профілю поточного користувача
PUT	`/api/users/profile`	Оновлення профільних даних поточного користувача
PUT	`/api/users/:id/role`	Зміна ролі користувача
PUT	`/api/users/:id/status`	Зміна статусу активності користувача
DELETE	`/api/users/:id`	Видалення або деактивація облікового запису

Таблиця А.3 – Ендпоінти модуля бібліотечних ресурсів

Метод	Ендпоінт	Призначення
GET	`/api/books`	Отримання списку бібліотечних ресурсів
GET	`/api/books/search`	Пошук ресурсів за параметрами
GET	`/api/books/:id`	Отримання детальної інформації про ресурс
POST	`/api/books`	Додавання нового бібліотечного ресурсу
PUT	`/api/books/:id`	Оновлення інформації про ресурс
DELETE	`/api/books/:id`	Видалення запису про ресурс

Таблиця А.4 – Ендпоінти модуля авторів

Метод	Ендпоінт	Призначення
GET	`/api/authors`	Отримання списку авторів
GET	`/api/authors/:id`	Отримання інформації про автора
POST	`/api/authors`	Додавання нового автора
PUT	`/api/authors/:id`	Оновлення інформації про автора
DELETE	`/api/authors/:id`	Видалення запису про автора

Таблиця А.5 – Ендпоінти модуля категорій

Метод	Ендпоінт	Призначення
GET	`/api/categories`	Отримання списку категорій
GET	`/api/categories/:id`	Отримання інформації про категорію
POST	`/api/categories`	Додавання нової категорії
PUT	`/api/categories/:id`	Оновлення інформації про категорію
DELETE	`/api/categories/:id`	Видалення категорії

Таблиця А.6 – Ендпоінти модуля примірників

Метод	Ендпоінт	Призначення
GET	`/api/copies/book/:bookId`	Отримання списку примірників певної книги
GET	`/api/copies/:id`	Отримання інформації про конкретний примірник
POST	`/api/copies`	Додавання нового фізичного примірника
PUT	`/api/copies/:id`	Оновлення даних про примірник
PUT	`/api/copies/:id/status`	Зміна статусу примірника
DELETE	`/api/copies/:id`	Видалення запису про примірник

Таблиця А.7 – Ендпоінти модуля електронних файлів

Метод	Ендпоінт	Призначення
GET	`/api/files/book/:bookId`	Отримання електронних файлів, пов’язаних із книгою
GET	`/api/files/:id`	Отримання інформації про електронний файл
POST	`/api/files`	Додавання відомостей про електронний файл
PUT	`/api/files/:id`	Оновлення інформації про електронний файл
DELETE	`/api/files/:id`	Видалення запису про електронний файл

Таблиця А.8 – Ендпоінти модуля бронювання

Метод	Ендпоінт	Призначення
POST	`/api/reservations`	Створення заявки на бронювання
GET	`/api/reservations/my`	Перегляд власних бронювань користувача
GET	`/api/reservations`	Отримання списку всіх бронювань
GET	`/api/reservations/:id`	Отримання інформації про конкретне бронювання
PUT	`/api/reservations/:id/confirm`	Підтвердження бронювання
PUT	`/api/reservations/:id/cancel`	Скасування бронювання
PUT	`/api/reservations/:id/complete`	Завершення бронювання

Таблиця А.9 – Ендпоінти модуля видачі та повернення книг

Метод	Ендпоінт	Призначення
POST	`/api/borrow-records/issue`	Реєстрація видачі примірника
PUT	`/api/borrow-records/:id/return`	Реєстрація повернення примірника
GET	`/api/borrow-records`	Отримання загальної історії операцій
GET	`/api/borrow-records/user/:userId`	Отримання історії операцій конкретного користувача
GET	`/api/borrow-records/copy/:copyId`	Отримання історії операцій конкретного примірника
GET	`/api/borrow-records/overdue`	Отримання списку прострочених повернень