МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Острозька академія»
Навчально-науковий інститут інформаційних технологій та бізнесу
Кафедра інформаційних технологій та аналітики даних

КВАЛІФІКАЦІЙНА РОБОТА
на здобуття освітнього ступеня бакалавра

на тему: «**Розробка адмін панелі для системи управління бібліотекою з онлайн каталогом та резервуванням книг»**

Виконав: студент 4 курсу, групи КН-41
першого (бакалаврського) рівня вищої освіти
спеціальності 122 Комп'ютерні науки
освітньо-професійної програми «Комп'ютерні науки»
Григорський Мартін Андрійович

Керівник: *викладач кафедри інформаційних технологій та аналітики даних*
Мацевич Денис Володимирович

Рецензент: *Розробник програмного забезпечення в ТОВ ОБНОВА-ЄВРОШОП, викладач кафедри інформаційних технологій та аналітики даних НаУОА*
Шведюк Володимир Володимирович

РОБОТА ДОПУЩЕНА ДО ЗАХИСТУ
Завідувач кафедри інформаційних технологій та аналітики даних _____
(проф., д.е.н. Кривицька О.Р.)
Протокол №10 від 28 травня 2025 р.

Острог, 2025

АНОТАЦІЯ
кваліфікаційної роботи
на здобуття освітнього ступеня бакалавра

Тема: *Розробка адмін панелі для системи управління бібліотекою з онлайн каталогом та резервуванням книг*

Автор: *Григорський Мартін Андрійович*

Науковий керівник: *Мацевич Денис Володимирович*

Захищена «……»…… 20__ року.

Пояснювальна записка до кваліфікаційної роботи: ***72*** *(кількість сторінок роботи) с., 15 (кількість рисунків) рис., 0 (кількість таблиць) табл., 0 (кількість додатків) додатків, 11 (кількість джерел) джерел.*

Ключові слова: *АДМІН ПАНЕЛЬ, БІБЛІОТЕКА, ОНЛАЙН КАТАЛОГ, REST API, CRUD, POSTMAN, CSV, СТАТИСТИКА, NODE, VUE*

Короткий зміст праці:

Кваліфікаційна робота присвячена розробці адміністративної панелі для системи управління бібліотекою з онлайн каталогом та резервуванням книг. У роботі проведено аналіз предметної області та огляд аналогічних рішень. Спроектовано архітектуру бекенд-частини з визначенням API-маршрутів і реалізовано CRUD-операції для користувачів, книг та бронювань. Впроваджено механізми імпорту/експорту даних через CSV, надсилання електронних сповіщень за допомогою Mailgun та візуалізацію статистики з використанням графіків. Проведено тестування API інструментом Postman і ручне тестування веб-інтерфейсу, що підтвердило коректність та стабільність системи.

Keywords: *ADMIN PANEL, LIBRARY MANAGEMENT SYSTEM, ONLINE CATALOG, REST API, CRUD OPERATIONS, POSTMAN, CSV, STATISTICS, NODE, VUE*

Summary of the work:

The bachelor’s qualification thesis is devoted to the development of an administrative panel for a library management system featuring an online catalog and book reservation functionality. The study conducts an analysis of the problem domain and reviews existing similar solutions. The backend architecture was designed, specifying API routes, and CRUD operations for users, books, and reservations were implemented. Data import/export via CSV, email notifications using Mailgun, and statistical visualization with charts were incorporated. API testing was performed using Postman, and manual testing of the web interface confirmed the system’s correctness and stability.

(підпис автора)

ЗМІСТ

ВСТУП

Актуальність. У сучасних умовах швидкого розвитку інформаційних технологій більшість сфер людської діяльності зазнають значних змін. Це стимулює компанії активно впроваджувати цифрові рішення, щоб зберігати конкурентоспроможність і відповідати темпам інновацій. Цифровізація сприяє зменшенню кількості помилок, які є характерними для роботи з паперовими носіями, а також дозволяє істотно скоротити витрати ресурсів і часу на обробку даних.

Серед таких компаній не винятком є сфера діяльності бібліотеки, яка традиційно вважається цінним ресурсом надання інформації користувачам, а це, у свою чергу, підкреслює неабияку важливість впровадження технологій у дану галузь, з метою надання структурованої основи для категоризації даних, також, враховуючи їхню значну кількість, забезпечення швидкого й точного пошуку. Все це свідчить про актуальність створення бази даних із відкритим вихідним кодом, що зможе значно спростити процес управління інформацією. Важливою складовою цього проєкту є розробка бекенд-частини програмного застосунку, адже саме вона відповідає за функціональність системи та забезпечує реалізацію ключових можливостей кінцевого продукту.

Мета дослідження – розробка бекенд частини інформаційної системи персональної бібліотеки.

Для досягнення поставленої мети потрібно розв’язати такі задачі дослідження:

– провести аналіз сфери діяльності бібліотеки;

– спроєктувати бекенд частину інформаційної системи;

– обґрунтувати вибір інструментів та методи реалізації програмного застосунку;

– реалізувати адміністративну панель для управління всіма даними системи;

– провести тестування розробленої системи.

Об‘єкт дослідження – бекенд частина системи електронної бібліотеки.

Предмет дослідження – методи й засоби створення інформаційної системи для бібліотеки, з акцентом на проєктуванні та реалізації її бекенд-частини.

Практична цінність створеної системи полягає в наступному функціоналі для користувачів та адміністратора:

- можливість реєстрації та входу до особистого кабінету з розширеним доступом до функціоналу користування послугами бібліотечного фонду;
- перегляд та фільтрація ресурсів за різними параметрами;
- бронювання книг та подальше управління цим процесом;
- автоматичне ведення історії видач літератури;
- додавання улюблених книг до персонального списку;
- отримання сповіщень про необхідність повернення книг;
- доступ адміністратора до керування всіма даними через спеціалізовану панель управління.

РОЗДІЛ 1

ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1. Визначення мети та завдань роботи

Основною метою даного проєкту - це розробка бекенд частини інформаційної системи бібліотеки, надійної та ефективної серверної структури, яка забезпечуватиме зберігання значних обсягів даних, їх обробку та масштабованість системи. Бекенд має надавати доступ до інформації про книги, користувачів та дії, що з ними пов’язані – зокрема, бронювання, видалення або додавання до списку вподобань тощо. Виокремлення бекенду як самостійної складової програмного забезпечення обґрунтоване його ключовим функціональним значенням, а також відмінностями у підходах до проєктування та реалізації порівняно з фронтендом – зокрема в аспектах архітектури, логіки взаємодії з базою даних і технічного інструментарію.

Ретельне опрацювання фронтенд частини забезпечує створення зручного й візуально привабливого інтерфейсу для користувача. Для цього розробники застосовують набір технологій, відмінний від бекенд-інструментарію:, а саме: мова програмування HyperText Markup Language (HTML) для передачі змісту та структури веб-сторінки, Cascading Style Sheets (CSS) для забезпечення дизайну, а JavaScript відповідальна за логіку сайту та інтерактивність. Проте цього недостатньо для повноцінної роботи платформи. Наприклад, коли користувач виконує пошук через інтерфейс, результати формуються не на самому клієнті, а завдяки обробці запитів на сервері — через алгоритми, реалізовані у бекенді. Саме серверна частина відповідає за доступ до баз даних, логіку взаємодії з ресурсами та реалізацію прикладного програмного інтерфейсу (API). Якщо система має дуже добре пропрацьований бекенд, тоді зникає ймовірність повільного завантаження чи виникнення помилок під час користування нею.[1] Та навіть більше, роль даної прихованої частини коду як елементу досягнення розробки результативної платформи відзначається такими можливостями:

сервер відповідає за збереження та керуванням інформацією, створює безпечні бази даних для захисту користувачів за допомогою шифрування конфіденційних даних, а також, запроваджуючи відповідне керування доступом, бекенд забезпечує інтеграцію з іншими системами й сервісами. Наприклад, це можуть бути соціальні мережі. Для того, щоб досягти таких результатів розробники повинні володіти зовсім іншими навичками, якщо порівнювати з фронтенд розробниками, у даному випадку знадобиться застосування мов програмування: вибір між Java, Python, Ruby і Node.js, а база даних може бути реалізована з допомогою MySQL, PostgreSQL або MongoDB, та зрештою створення API, які і дозволяють веб-додаткам взаємодіяти з іншими сервісами.[2]

Зі сторони бекенд розробки майбутньої системи, для досягнення поставленої мети потрібно буде розв’язати низку завдань:

- Проєктування бази даних із використанням реляційної СУБД, включаючи визначення сутностей (наприклад, книги, користувачі, автори, жанри тощо), встановити їхні взаємозв'язки, а також створити відповідні таблиці й структури даних для зберігання інформації про книги, користувачів, процеси видачі та бронювання літератури. Особливу увагу слід приділити нормалізації бази даних з метою усунення надмірності та забезпечення ефективної роботи системи.
- Розробити архітектуру серверної частини для обробки запитів користувачів, облік та видачу книг, обробку заявок щодо бронювання і тому подібне, забезпечити ефективну взаємодію з базою даних та її високу продуктивність. Визначити принципи обробки запитів, розподілу навантаження та масштабованості серверної інфраструктури.
- Розробити та інтегрувати API, що забезпечуватиме взаємодію між бекендом і фронтендом, а також іншими частинами системи. API надаватиме доступ до різних даних про книги, користувачів, а також дозволятиме виконання таких дій, як реєстрація, авторизація, пошук, бронювання, видача та повернення книг.

- Реалізувати функції адміністрування, зокрема створення адміністративної панелі для керування даними користувачів, CRUD операціями для книг, а також контролю процесів видачі та бронювання книг. Окремо реалізовати функціонал для надання статистики за допомогою графіків, що дозволить моніторити стан бібліотечної системи.
- Забезпечити безпеку даних користувачів, тобто захист від несанкціонованого доступу, а також належний доступ до системи, зокрема реалізація механізмів аутентифікації та авторизації,.
- Провести тестування API та бекенд-системи за допомогою таких інструментів, як Postman з метою перевірки коректності роботи всіх маршрутів і стабільності обробки запитів.

1.2. Короткий опис предметної області

Інноваційні технології з кожним днем охоплюють все більше сфер життєдіяльності людини, і тепер стає досить складно зустріти будь-яку організацію, яка б не використовувала їх для покращення власної діяльності. Проте досі існують заклади, які користуються паперовими документами з ціллю збереження величезної кількості інформації, і сюди відносяться бібліотеки.

Згідно з визначенням, наведеним у Законі України, бібліотекою вважається така установа або ж структурний підрозділ, або інакше організація інформаційного, навчального чи культурного спрямування, яка володіє доступом до джерел різноманітної інформації і має упорядкований фонд документів, мета такого закладу полягає в забезпеченні потреб користувачів у сфері інформаційних, освітніх та наукових, культурних й інших галузей. [3]

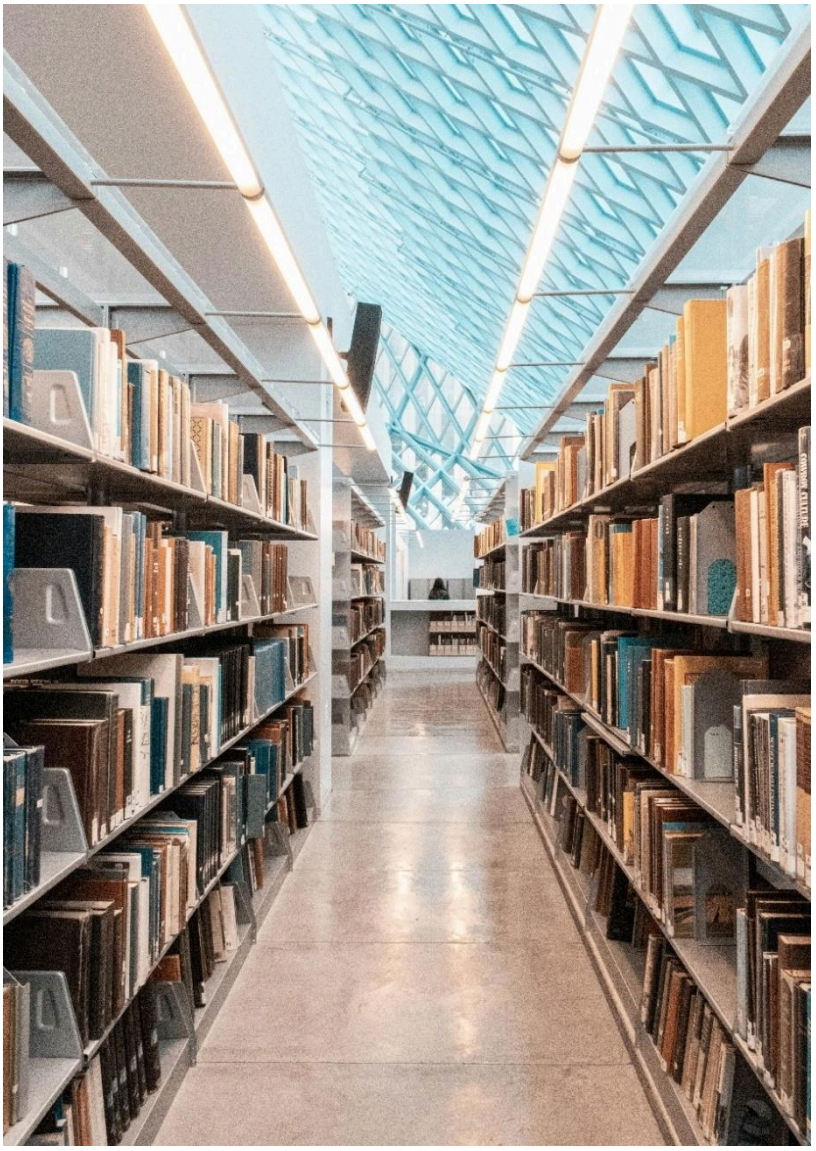


Рис. 1.1. Вигляд бібліотеки

Усі ресурси, що доступні в бібліотеці є безкоштовними, від книжкових фондів і освітніх програм до доступу в Інтернет.. Цікавим є судження, що такі установи не забезпечують належної економіки для держави, хоча насправді все навпаки, адже з усім наданим арсеналом, бібліотеки забезпечують всі верстви населення доступом до знань, різноманітних освітніх ресурсів, таким чином допомагаючи кожному розвивати навички та підвищувати кваліфікацію, знаходити роботу або навіть започатковувати власні справи, що відповідно зміцнює державу, сприяючи її економічному процвітанню.

Право на обслуговування бібліотеки надається таким користувачам: працівникам, учням школи, студентам та їхнім батькам. Діяльність бібліотеки можна умовно поділити на чотири основні напрями: збір, збереження, видача літератури користувачам на визначений термін і подальше її повернення. У контексті освітньої сфери ці процеси організовані наступним чином:: працівником виступає бібліотекар, у його розпорядженні знаходяться документи, у які він заносить дані щодо індивідуального номеру користувача, а також номер книги, дати видачі та повернення книги із бібліотечного фонду, і позначка щодо своєчасності повернення. Також, крім ІД номеру, у кожного користувача повинна бути вказана особиста інформація і контактні дані. Якщо мова йде про учня, то повинен бути присутній код класу.

Кожен користувач бібліотеки має змогу одержати консультацію при потребі щодо пошуку джерел інформації, отримання книг для власного використання надається на проміжок часу не більше п’ятнадцяти днів, а також максимальна кількість для одноразового застосування рівна п’яти книгам, і якщо відсутня конкретна література, то кожен має право отримати інформацію про те, де можна віднайти її в інших бібліотеках, окрім цього, користувачі мають змогу брати активну участь у заходах установи та бути залученими до бібліотечного активу. Основними формами діяльності бібліотеки вважаються абонемент й читацький зал. Останній має деякі особливості щодо обслуговування, наприклад, кожна бібліотека визначає перелік книг, які можуть видаватись у читацькому залі. При цьому кількість видань не є обмеженою. Саме в читальному залі дозволяється працювати з рідкісними або цінними книгами, спеціальними довідковими матеріалами та іншими унікальними ресурсами.. У загальні обов’язки бібліотекаря входять також заповнення звітів про надходження нових книг до бібліотеки і відправлення.[4]

Більшість процесів, пов’язаних з обліком бібліотечного фонду, досі здійснюється вручну та потребує фізичної присутності користувача в установі. Такий підхід суттєво ускладнює й уповільнює обслуговування - наприклад, для

того щоб з’ясувати наявність певної книги, потрібно витратити значний час. Крім того, виникають труднощі зі структуризацією та пошуком літератури. Усе це підтверджує нагальну потребу у впровадженні в бібліотеках автоматизованих інформаційних систем. Вони дозволяють систематизувати книжковий фонд, пришвидшити пошук і значно підвищити ефективність обслуговування користувачів. Зокрема, кожне нове надходження до бібліотеки буде реєструватися в електронній системі з обов’язковим зазначенням основної інформації: автор, назва, рік видання, тематика, індекси, місце зберігання, статус доступності тощо.

Таку інформаційну систему зазвичай позначають терміном «цифрова бібліотека». Згідно з визначенням, наведеним на офіційному сайті Digital Library Federation (у перекладі – «Федерація цифрової бібліотеки»), цифрова бібліотека - це колекція електронних матеріалів, доступ до яких здійснюється за допомогою персонального комп’ютера, локальної мережі або Інтернету. Основні принципи функціонування цифрової бібліотеки включають:

1) Дотримання визначених правил і стандартів щодо формування та управління ресурсами бібліотеки;

2) Забезпечення ефективного пошуку інформації в межах колекцій з гарантією безпечного та надійного доступу до матеріалів користувачам.[5]

Особливої ролі відіграє внутрішня бекенд частина бази даних, яка буде відповідати за більшість рутинних справ, що зазвичай виконувались вручну, а тепер можна досягти завдяки сучасному технологічному інструментарію сортування, тегування, групування, тобто йдеться про автоматизацію таких процесів: оновлення статусу книг, а саме, чи вона видана, повернута або зарезервована, ведення історії користування, реалізовуватимуться механізми, які відповідальні за класифікацію книг за темами, галузями знань, віковими категоріями, рівнем складності, типом видання, а також фіксуватимуться загальні аналітичні дані щодо читацьких запитів.

1.3. Постановка задачі

Відповідно до вищенаведеного опису предметної області можна виділити відповідно задачі, які потрібно розв’язати:

- Вибрати оптимальні інструменти розробки. Щодо СУБД, то надійною системою буде PostgreSQL, що є з відкритим кодом і не потребує значних витрат з метою адміністрування, відома своєю простотою у використанні, а також здатністю запускати веб-сайти, які є динамічними, і як варіант стека LAMP запускати відповідні веб-програми. Крім того, дана система є дуже високостійкою щодо відмов, таке судження основане саме на можливості журналювання з випереджальним записом .[6]

- Спроєктувати базу даних, першочергово визначивши сутності та їх зв’язки, і після цього побудувати діаграми «сутність-зв’язок», тобто ER-діаграми, та на її основі за допомогою SQL створити відповідні таблиці бази даних, враховуючи нормалізацію.

- Сформувати логічну структуру, тобто архітектуру бекенд частини системи, визначивши основні модулі, і як відбуватиметься обробка запитів, помилок, авторизації тощо.

- Визначити API маршрутів – шляхи, завдяки яким адміністративна панель буде взаємодіяти напряму з сервером, кожен маршрут буде задавати запити на виконання певної операції щодо винесення даних, створення записів, редагування або видалення. Наприклад, операції:

- Створення логіну та реєстрації

Кожен користувач має змогу логіну та реєстрації в системі, щоб отримати доступ до інтерфейсу облікового запису, де можна буде переглядати особисту інформацію, а також відображатимуться й інші деталі профілю пов’язані з бібліотечним фондом.

- Робота профілю користувача

Залогінений користувач має змогу переглядати та редагувати дані облікового запису, а також є можливість видалення власного профілю за бажанням. Окрім того, тільки для тих користувачів, які успішно ввійшли буде доступний деякий функціонал, а саме: бронювання книг, перегляд особистого кабінету, історія видач, редагування профілю, отримання нагадувань, перегляд статусу книг.

- Пошук, фільтрація та сортування книг

Користувачі мають змогу застосовувати фільтри на головній сторінці для комфортного сортування книг, відповідно до власних уподобань. Проводити таку операцію можна буде у верхній частині інтерфейсу, тобто хедері, і для швидкого пошуку літератури можна буде використовувати тільки часткові назви.

- Управління каталогами книг

Така система забезпечуватиме можливість перегляду книжкового каталогу, додатково буде доступною реалізація створення нових каталогів та видалення раніше вже створених.

- Функціонал щодо реалізації бронювання книг

Кожен користувач здатен виконувати бронювання і з можливістю перегляду оновлення статусу бронювання, а також з видаленням бронювання у разі відміни або помилки.

- Розрахунок штрафів, які надаються тим користувачам, які прострочили термін щодо здачі книг

Інформаційна система буде автоматично обчислювати штраф та повідомляти про нього користувача, здійснюватиметься це за допомогою візуалізації для зручності.

- Збереження вподобаних книг

Користувачі зможуть додавати до списку улюблених книги і така інформація зберігатиметься в особистому кабінеті, а також надаватиметься можливість видаляти у будь-який момент ту чи іншу книгу з даного списку.

Загалом дані вимоги дозволяють краще зрозуміти платформу та чітко описують завдання, які необхідно реалізувати в цій частині проєкту.

Створення адміністративної панелі:

- CRUD (створення, редагування, видалення, перегляд) користувачів.
- Блокування користувача в системі.
- CRUD операції для книг.
- Операції для керування виданих книг: створення на книги, скасування бронювання.
- Автоматичне надсилання сповіщення-нагадування про необхідність повернення книг, через електронну пошту.
- Журнал історії видач, місце, де відображатиметься інформація, хто і коли отримував книги.
- Імпортування та експортування книг через CSV, з метою спрощення масового оновлення бази даних.
- Реалізація статистики через графіки для перегляду книг, які наразі є популярними, кількість бронювань, активність користувачів тощо.

Завершення: перевірка роботи API за допомогою тестування Postman, щоб упевнитися, що система працює коректно, стабільно, кожен маршрут проходить тестування на правильність запитів та відповідей, помилок, обробки даних, авторизації, і якраз-таки Postman дає змогу самостійно налаштовувати автоматизовані запити й створити середовище користувача взагалі з будь-якими змінними, що врешті-решт можна буде легко застосовувати під час розробки потрібних запитів та тестів.[7]

Висновок до розділу 1

У межах першого розділу було здійснено ґрунтовний аналіз мети та завдань, пов’язаних із розробкою бекенд-частини інформаційної системи бібліотеки, що відповідає за зберігання, обробку й передачу даних, а також забезпечує стабільну взаємодію користувача з функціоналом платформи. Проведений огляд предметної області дозволив визначити роль бібліотек як осередків знань і культурного розвитку суспільства, а також окреслити їхню нагальну потребу в цифровізації процесів. Було встановлено, що більшість установ досі використовують застарілі методи роботи з інформацією, переважно у паперовому форматі, що суттєво знижує ефективність обслуговування користувачів. На основі аналізу діяльності бібліотек та актуальних потреб було сформульовано перелік завдань для подальшої реалізації системи. До них віднесено: проєктування бази даних, побудову архітектури серверної частини, розробку та документування API, реалізацію основних функцій, а саме авторизації, пошуку, бронювання, створення списку вподобаних книг, а також створення адміністративної панелі й проведення тестування розробленого програмного забезпечення. Все це є критично важливими для забезпечення надійної та функціональної інформаційної системи бібліотеки.

РОЗДІЛ 2

ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Проєктування та розробка бази даних

2.1.1. Визначення сутностей

Після завершення попередніх кроків розробки можна перейти до однієї з найважливіших частин проєкту – створення бази даних.

База даних – це організований набір структурованої інформації або даних, які зазвичай зберігаються в електронному вигляді в комп'ютерній системі. База даних зазвичай контролюється системою керування базами даних (СУБД).

Для того, щоб перейти до створення БД необхідно визначити, які саме сутності будуть використовуватись в наступних кроках реалізації системи.

Провівши аналіз предметної області, можна визначити наступні сутності для системи з управління бібліотекою: [8]

- Authors
- Books
- Collections
- CollectionsBooks
- Copies
- Fines
- Genres
- Likes
- Loans
- Publishers
- Reservations
- Users

2.1.2. Моделювання даних завдяки ER-діаграмам

Після визначення сутностей наступними кроком буде створення зв’язків між ними завдяки концептуальній діаграмі.

Концептуальна модель не відноситься до того, як буде реалізована реальна база даних. Вона працює на дуже високому рівні, переводячи зв’язки реального світу та бізнес-вимоги в концептуальну структуру, яку легко зрозуміти, і тому її можна навіть представити не експертам. [9]

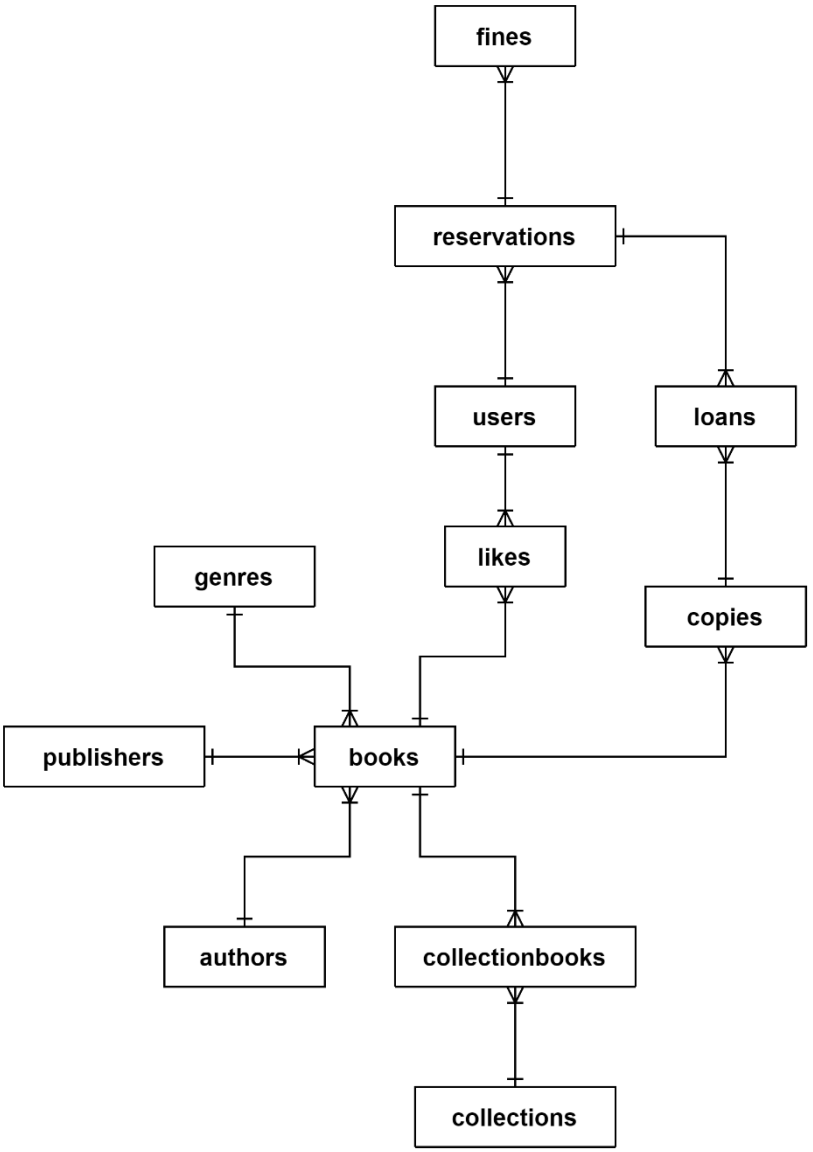


Рис. 2.1. Створена концептуальна діаграма

Оглянувши діаграму, можна побачити, що Книги є центральною таблицею, з якою певним чином зв’язані інші. Наприклад, через інформацію про видавців, авторів, бронювання чи позики.

Далі, щоб деталізувати створену діаграму, потрібно додати до неї поля та перетворити її у логічну модель.

Логічна модель даних є складнішою. Вона все ще не залежить від конкретної використовуваної системи (СУБД), але, починаючи з концептуальної моделі, намагається перевести високорівневі уявлення у формальну абстрактну схему, додаючи наступний рівень деталізації у вигляді полів. [10]

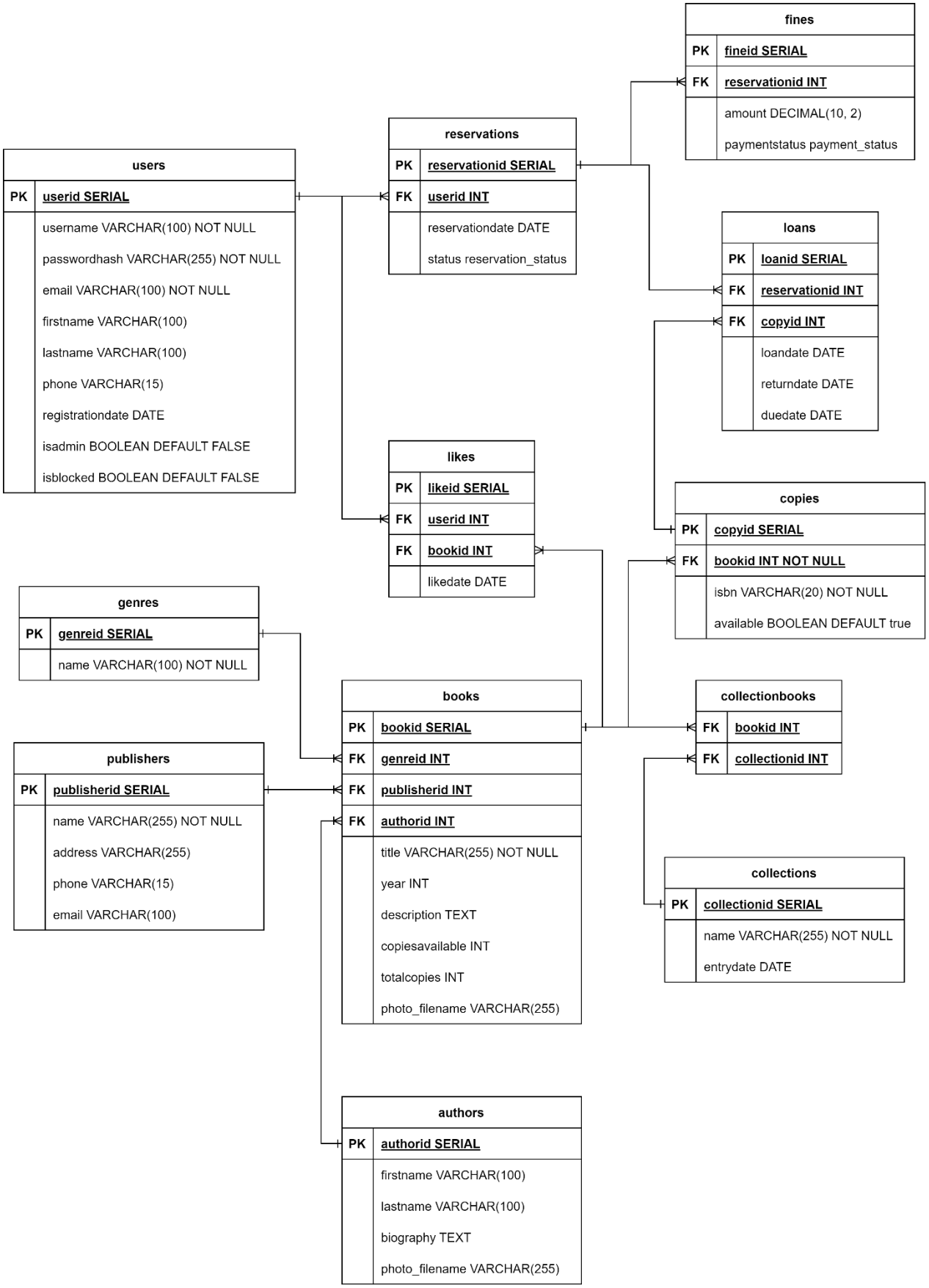


Рис. 2.2. Створена ER-діаграма

2.1.3. Перевірка нормалізації та створення таблиць

Нормалізація бази даних – це структурований набір кроків для оптимального проєктування моделі бази даних.[11]

1NF – перша нормальна форма

- Усунення повторюваних стовпців даних
- Кожен атрибут даних містить одне значення замість кількох значень

2NF – друга нормальна форма

- Дані знаходяться в першій нормальній формі
- Дані повністю функціонально залежать від ідентифікаторів ключів

3NF – третя нормальна форма

- Дані знаходяться в другій нормальній формі
- У даних немає транзитивних залежностей. Це означає, що нам потрібно переконатися, що жоден інший атрибут (крім ідентифікаторів ключів або стовпців ID) не зможе однозначно визначити результат атрибута.

Переглянувши реалізовану структуру, можна побачити, що модель відповідає нормам 3НФ, тому вона повністю готова до практичної реалізації за допомогою обраної СУБД, код реалізації бази даних, разом з даними, знаходиться в додатках.

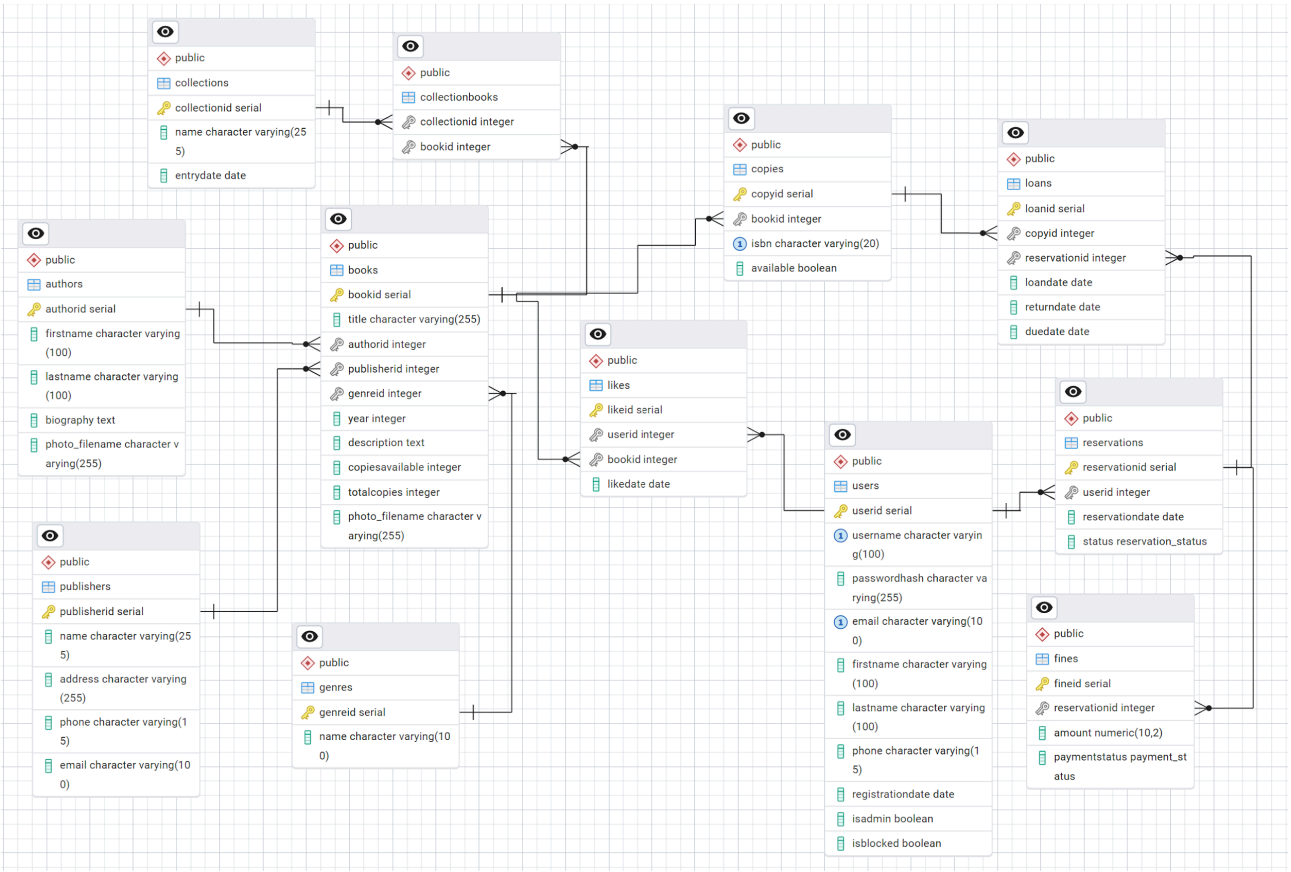


Рис. 2.3. Нормалізована діаграма в середовищі PgAdmin4

2.2. Архітектура бекенду веб-сайту

Розглядаючи підхід для написання серверної частини застосунку, необхідно звернути свою увагу на побудову правильно структурованої архітектури. Вона допоможе коректно керувати, змінювати та ефективно користуватись створеним сервером.

Ця частина проєкту є відповідальною за логіку всього проєкту, взаємодію з базою даних, аутентифікацією та комунікацією з фронтедом, що робить підхід до його структури максимально важливим. Її розробка також враховує саму тему, що розглядається, керування бібліотек з розмежування на логічні блоки у вигляді файлів js.

В сучасній веб-розробці часто використовується підхід з багаторівневою структурою. Її головна задумка в тому, щоб розділяти написаний код на функціональні та логічно відмінні частини, кожна з яких має власну зону

відповідальності, дозволяючи розробляти ці незалежні блоки окремо, що в свою чергу, надає можливість зменшити дублювання. [12]

Загальна архітектура створеного проєкту, виглядає наступним чином:

Рис. 2.4. Структура бекенду

Опис частин проєкту:

.env: реалізований для зберігання змінних середовища, що необхідно приховати (URL, JWT).

app.js: головний файл застосунку, який вміщає в собі підключення всіх зовнішніх файлів, об’єднуючи серверний додаток в одне ціле.

package.json та package-lock.json: конфігурації для проєкту.

Controllers: папка, яка вміщає в собі центральну логіку кожної частини застосунку REST API, розділяється на:

- authController.js: контролер відповідає за здійснення роботи з аутентифікацією користувачів, тобто логіном та реєстрацією.
- booksController.js: працює з CRUD операціями щодо книг, додатково надаючи можливість: пошуку по назві, фільтрування за наданим критерієм та виведення інформації про жанри і авторів.
- collectionsController.js: відповідає за організацію структури для каталогів книг.
- finesController.js: контролер, який обраховує та виводить штрафи користувачів.
- likesController.js: файл описує частини проєкту, що повністю відповідає за керування вподобаннями книг користувачів, включаючи повне їх виведення, додавання нових або видалення.
- loanController.js: контролер, що створений для керування видачами, використовується тільки адміністраторами.
- reservationController.js: відповідає за логіку створення бронювань книг для читачів.
- statisticsController.js: файл для обробки та отримання даних для створення статистики.
- userController.js: контролер, що керує та виводить дані для роботи профілів користувачів.

Middlewares: директорія, яка зберігає код, що відповідає за перевірку типу користувача, тобто чи будуть доступні йому функції адміністратора та адмін-панелі, чи буде відкритий стандартний профіль.

Routes: папка, яка налічує файли для назв маршрутів створених та приєднаних до ендпойнтів контролерів, реалізованих в описаних файлах controllers.

Db: налаштування підключення до бази даних (використовується тільки при локальному запуску), допомагає проводити всі запити, що запускаються на сервері.

Public: папка для статичних файлів зображень, використовується для обкладинок книг та фото авторів.

node_modules: директорія, що зберігає залежності, встановлені за допомогою npm.

Загалом можна впевнено сказати, що обраний підхід до реалізації архітектури застосунку надає чіткий, швидкий та гнучкий додаток для системи бібліотеки, забезпечуючи чітку та зрозумілу структуру.

2.3. Визначення API-маршрутів

Після реалізації контролерів необхідно задати кожному ендпойнту власний маршрут. Ці посилання нададуть можливість фронтенду або іншому серверу запускати на виконання бажані котролери, які керують командами CRUD.

У цьому проєкті було використано принципи розробки RESTful API, які дозволяють ендпойнтам надсилати спеціальну команду HTTP, що поділяється на GET, POST, PUT, DELETE. [13]

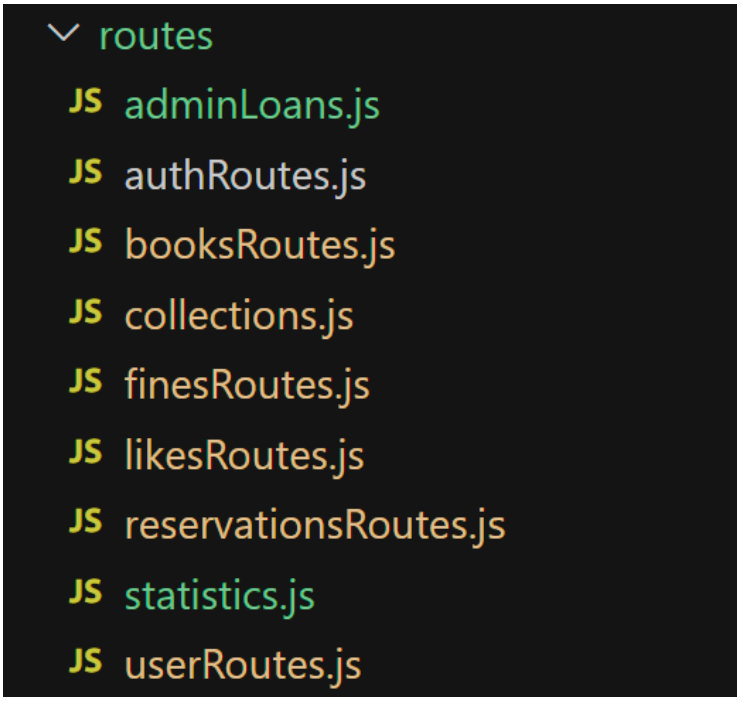


Рис. 2.5 Файли з реалізованими маршрутами

Описуючи кожен файл окремо, в них містяться наступні ендпойнти:

Табл.2.1. Маршрути adminLoans.js

Метод	Назва ендпойнту	Опис
GET	/api/loans/active	Виведення позик, що на даний час є актуальними
GET	/api/loans/history	Виведення всієї історії позик книг
POST	/api/loans/remind	Запит для надсилання сповіщення про повернення книги на пошту
POST	/api/loans/reservations	Додавання бронювання книг для адміністратора

Табл.2.2. Маршрути authRoutes.js

Метод	Назва ендпойнту	Опис
POST	/api/auth/register	Реєстрація користувача в системі

POST	/api/auth/login	Логін в додатку для отримання доступу до профілю

Табл.2.3. Маршрути booksRoutes.js

Метод	Назва ендпойнту	Опис
GET	/api/books	Виведення всіх книг
GET	/api/books/id/:id	Виведення книги з обраним ідентифікатором
POST	/api/books	Додавання нової книги
PUT	/api/books/:id	Редагування даних про книгу
DELETE	/api/books/id/:id	Видалення книги по ідентифікатору
GET	/api/books/search	Пошук книг за їхніми назвами
GET	/api/books/filters	Виведення книг з використанням обраних фільтрів
GET	/api/books/authors	Пошук всіх авторів
GET	/api/books/publishers	Видавці, що були внесені в систему
GET	/api/books/genres	Виведення всіх жанрів

Табл.2.4. Маршрути collections.js

Метод	Назва ендпойнту	Опис
GET	/api/collections	Виведення всіх колекцій
GET	/api/collections /:id/books	Пошук колекції за ідентифікатором

Табл.2.5. Маршрути finesRoutes.js

Метод	Назва ендпойнту	Опис
GET	/api/fines	Запит для знаходження всіх боргів читача

GET	/api/fines/total	Загальний борг

Табл.2.6. Маршрути likesRoutes.js

Метод	Назва ендпойнту	Опис
GET	/api/likes	Виведення всіх вподобаних книг користувача
POST	/api/likes	Додання нового вподобання
DELETE	/api/likes/:likeid	Видалення книги з вподобаних

Табл.2.7. Маршрути reservationsRoutes.js

Метод	Назва ендпойнту	Опис
GET	/api/reservations	Пошук всіх бронювань користувача
POST	/api/reservations	Створення бронювання для обраної книги
PUT	/api/reservations /status	Редагування статусу бронювання
DELETE	/api/reservations /:reservationId	Видалення бронювання з історії

Табл.2.8. Маршрути statistics.js

Метод	Назва ендпойнту	Опис
GET	/api/statistics/top-books	Пошук книг, які є найпопулярнішими
GET	/api/statistics/top-users	Знаходження найактивніших користувачів
GET	/api/statistics/top-genres	Найпопулярніші жанри

GET	/api/statistics/export-books	Експортування всіх книг системи в CSV
POST	/api/statistics/import-books	Імпортування книг з CSV
GET	/api/statistics /monthly-loans	Підрахунок позик книг з групуванням по місяцям
GET	/api/statistics/fewest-copies	Пошук книг, що мають найменше копій
GET	/api/statistics/restock-books	Виведення книг, які потребують поповнень
GET	/api/statistics /average-copies	Підрахунок середньої кількості копій на кожну книгу

Табл.2.9. Маршрути userRoutes.js

Метод	Назва ендпойнту	Опис
GET	/api/user/get-users	Виведення даних про всіх користувачів
PUT	/api/user/update-profile	Оновлення профілю користувача
PUT	/api/user/admin-update-user	Оновлення профілю користувача через адмін-панель
DELETE	/api/user/delete-user	Видалення профілю користувача через адмін-панель
GET	/api/user/me	Виведення особистих даних користувача

Висновок до розділу 2

На даному етапі було розроблено базу даних на основі виокремлених дванадцяти головних сутностей, після чого можна було побудувати зв’язки між ними за допомогою концептуальної ER-діаграми, а далі деталізувати її і розробити логічну діаграму. На наступному кроці проводилась нормалізація БД, завдяки якій системі було забезпечено відсутність надлишкових та транзитивних залежностей. Таким чином можна зберігати дані не дублюючи їх. Оскільки модель відповідала нормам третьої нормальної форми, вона була готовою до практичної реалізації. Далі розроблялась серверна частина системи, попередньо враховуючи принципи багаторівневої структури. На завершальному етапі було реалізовано RESTful API, з метою визначення маршрутів для кожного контролера, та ефективної комунікації між клієнтською частиною та сервером, додатково, виконуючи всі необхідні CRUD-операції для повноцінного функціонування бібліотечної системи.

РОЗДІЛ 3

ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1. Засоби розробки

Серед інструментів реалізації, які варті згадки у проєкті, було використано: Node.js та Express для реалізації серверної частини; PostgreSQL для зберігання даних.[14]

Node.js є середовищем розробки, що дозволяє запускати розроблений застосунок, він успадкував кілька функцій від JavaScript, що зробило його одним з найпопулярніших застосунків для реалізації бекенду. Незалежно від типу застосунку — веб-сервер, онлайн-чат або інтеграція з IoT-пристроями — Node.js здатен ефективно відповідати широкому спектру вимог сучасної розробки. [15]

Express є одним з найпопулярніших фреймворків, що працює разом з Node.js. Його простота у використанні, підтримка маршрутизації, middleware-функцій і висока розширюваність роблять Express одним із найпоширеніших інструментів для побудови серверної логіки. Завдяки Express розробники можуть швидко створювати масштабовані RESTful API з чіткою структурою та ефективною обробкою запитів.

PostgreSQL – це об’єктно-реляційна система керування базами даних, що поєднує в собі комбінацію інструментів реляційних БД, надаючи гнучкість розробки з об’єктно-орієнтованих систем. Ця СУБД використовується для різноманітних застосунків, поєднуючи зберігання та керування даних, створення запитів, маніпуляцію великих об’ємів даних та аналітику щодо використовуваної інформації. PostgreSQL обирають розробники через її зручність, швидкість та переваги в функціоналі в порівнянні з конкурентами. [16]

3.2. Проєктування функціональних модулів системи

Загалом створений API розділений на спеціальні файли, що зберігають в собі необхідні функції, які надають користувачу можливість взаємодіяти з системою: оглядати дані, вносити їх, проводити зміни та видаляти.

Для цього реалізується загальний файл бекенду app.js, що зберігає в собі проміжні файли з маршрутами. [17]

```
const collectionsRoutes = require("./routes/collections");
const booksRoutes = require("./routes/booksRoutes");
const authRoutes = require("./routes/authRoutes");
const userRoutes = require("./routes/userRoutes");
const reservationsRoutes = require("./routes/reservationsRoutes");
const loansRoutes = require("./routes/adminLoans");
const finesRoutes = require("./routes/finesRoutes");
const likesRoutes = require("./routes/likesRoutes");
const statisticsRoutes = require("./routes/statistics");
const app = express();
const PORT = 3000;

app.use(express.json());
app.use(express.static(path.join(__dirname, "public")));
app.use(cors());
app.use("/images", express.static(path.join(__dirname, "public")));

// Маршрути
app.use("/api/collections", collectionsRoutes);
app.use("/api/books", booksRoutes);
app.use("/api/auth", authRoutes);
app.use("/api/user", userRoutes);
app.use("/api/reservations", reservationsRoutes);
app.use("/api/loans", loansRoutes);
app.use("/api/fines", finesRoutes);
app.use("/api/likes", likesRoutes);
app.use("/api/statistics", statisticsRoutes);

// Запуск сервера
app.listen(PORT, () => {
  console.log(`Сервер працює на http://localhost:${PORT}`);
});
```

Рис. 3.1. Маршрути головного файлу

Далі кожен з цих маршрутів має мати два файли: router для створення конкретних маршрутів та controllers, що зберігатиме бажаний функціонал в згрупованих файлах. [18]

Для прикладу, можна подати функціонал, що відповідає за виведення книг, що були вподобані користувачем.

```
router.get("/", getUserLikes);
router.post("/", addLike);
router.delete("/:likeid", removeLike);
```

```
exports.getUserLikes = async (req, res) => {
  try {
    const userId = getUserIdFromToken(req);

    const { rows } = await client.query(
      `SELECT l.likeid, b.bookid, b.title AS book_title, a.firstname AS author_firstname, a.lastname AS author_lastname,
              p.name AS publisher_name, g.name AS genre_name, b.year, b.description AS book_description,
              b.copiesavailable, b.totalcopies, b.photo_filename AS book_photo
       FROM books b
       JOIN authors a ON b.authorid = a.authorid
       JOIN publishers p ON b.publisherid = p.publisherid
       JOIN genres g ON b.genreid = g.genreid
       JOIN likes l ON b.bookid = l.bookid
       WHERE l.userid = $1`,
      [userId]
    );
    if (rows.length === 0) {
      console.log("Вподобання не знайдено для цього користувача");
      return res.status(404).json({ message: "Вподобання не знайдено" });
    }
    return res.json(rows);
  } catch (error) {···
  }
};
```

Рис. 3.2-3. Приклад реалізації модулів системи

Більше детальний аналіз створених маршрутів наведений в наступному підрозділі.

3.3. Реалізація адміністративної панелі

Головною частиною розробки дипломної роботи є реалізація адмін-панелі, що дозволить зручно та швидко керувати системою, без потреби внесення змін через СУБД або серверно, надаючи цей доступ відразу на вебсайті.

Панель адміністратора – це централізований інтерфейс для користувачів, у якому є розширені права доступу, що надає можливість керувати даними застосунку. Наступні підрозділи будуть описувати процес розробки цих компонентів.

3.3.1. Керування даними користувачів

При роботі системи онлайн бібліотеки часто виникає потреба в перегляді та зміні даних користувачів застосунку.

Щоб це реалізувати, необхідно створити кілька додаткових ендпойнтів, що допоможуть в реалізації адмін-панелі сторінки користувачів. Створення їх буде відбуватись за структурою, побудованою у попередніх частинах роботи.

Першим кроком є розробка назв маршрутів, що будуть застосовуватись у цій частині застосунку, в окремому файлі userRoutes.js.

backend > routes > JS userRoutes.js > ...

```
1   const express = require("express");
2   const router = express.Router();
3   const {
4     getUsers,
5     updateProfile,
6     updateUserByAdmin,
7     deleteUser,
8     getUserByToken,
9   } = require("../controllers/userController");
10  const { authenticateToken, checkIsAdmin } = require("../middlewares/auth");
11
12  router.get("/get-users", getUsers);
13  router.put("/update-profile", updateProfile);
14  router.put(
15    "/admin-update-user",
16    authenticateToken,
17    checkIsAdmin,
18    updateUserByAdmin
19  );
20  router.delete("/delete-user", deleteUser);
21  router.get("/me", getUserByToken);
22
23  module.exports = router;
24
```

Рис. 3.4. Маршрути для керування користувачами

При детальнішому огляді ендпойнтів, можна дійти висновку, що кожен з них відповідає за певну частину роботи застосунку.

getUsers – надає інформацію про всіх користувачів. Це використовується для того, щоб вивести та переглянути всі дані у вигляді таблиці, що буде

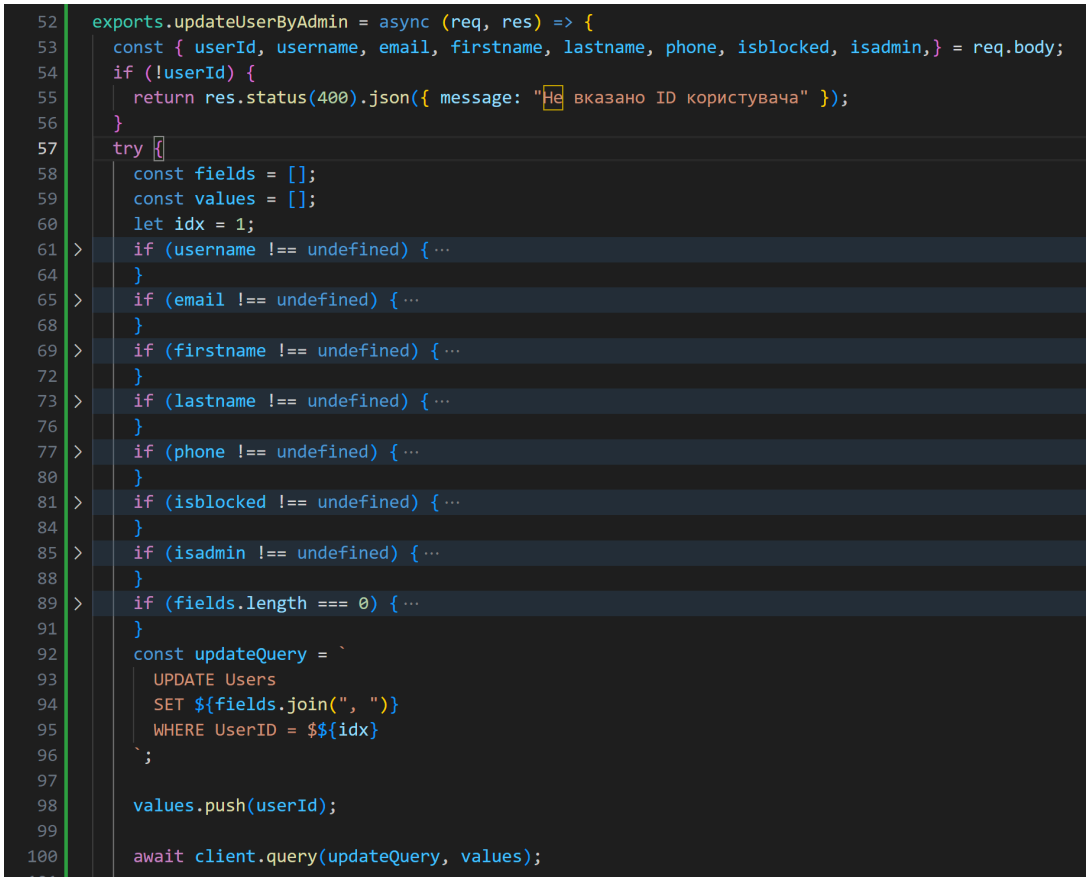
зручним для швидкого перегляду бажаної інформації, а це допоможе, при потребі, вносити дані до обраного користувача. [19]

```
41  exports.getUsers = async (req, res) => {
42    try {
43      const query = "SELECT * FROM Users";
44      const result = await client.query(query);
45      res.json(result.rows);
46    } catch (err) {
47      console.error(err);
48      res.status(500).json({ message: "Помилка сервера" });
49    }
50  };
```

Рис. 3.5. Функція getUsers

updateUserByAdmin – функція, що відповідає за оновлення особистих даних користувача, саме через адмін-панель, тому що звичайне оновлення доступне тільки для користувачів з валідацією в особистому кабінеті.

Тут йде додаткова перевірка, які саме поля адміністратор обрав для оновлення, чи можуть бути звичайні дані з форми реєстрації, так і зміни прав доступу до системи. Конкретніше, адміністратор може заблокувати користувача або навпаки – надати доступ до повноцінної роботи в системі. [20]



```
exports.updateUserByAdmin = async (req, res) => {
  const { userId, username, email, firstname, lastname, phone, isblocked, isadmin,} = req.body;
  if (!userId) {
    return res.status(400).json({ message: "Не вказано ID користувача" });
  }
  try {
    const fields = [];
    const values = [];
    let idx = 1;
    if (username !== undefined) {···
    }
    if (email !== undefined) {···
    }
    if (firstname !== undefined) {···
    }
    if (lastname !== undefined) {···
    }
    if (phone !== undefined) {···
    }
    if (isblocked !== undefined) {···
    }
    if (isadmin !== undefined) {···
    }
    if (fields.length === 0) {···
    }
    const updateQuery = `
      UPDATE Users
      SET ${fields.join(", ")}
      WHERE UserID = $${idx}
    `;

    values.push(userId);

    await client.query(updateQuery, values);
```

Рис. 3.6. Функція updateUserByAdmin

Останньою функцією для керування користувачами є deleteUser.

По своїй суті вона не блокує користувача, що робиться тимчасово і завжди може бути виправлено, а повністю видаляє дані та прогрес читача в системі. Її застосування є найбільш обґрунтованим при помилковій реєстрації або при тривалій відсутності активності профілю.

```
exports.deleteUser = async (req, res) => {
  const { userId } = req.body;

  try {
    const deleteQuery = "DELETE FROM Users WHERE UserID = $1";
    await client.query(deleteQuery, [userId]);

    res.json({ message: "Користувача видалено успішно" });
  } catch (err) {
    console.error(err);
    res.status(500).json({ message: "Помилка сервера" });
  }
};
```

Рис. 3.7. Функція deleteUser

3.3.2. CRUD операції для книг

Центральною частиною кожної бібліотеки є книги, інформація щодо яких завжди повинна бути актуальною та при потребі швидко відредагованою. Тому наступні ендпойнти були створені для повноцінної роботи системи.

Як і в попередньому кроці, необхідно продумати маршрути, що будуть реалізовані в цій частині завдання. Для цього необхідно продовжити роботу в booksRoutes.js. [22]

Рис. 3.8. Маршрути, що використовуватимуться для керування книгами в адмін-панелі

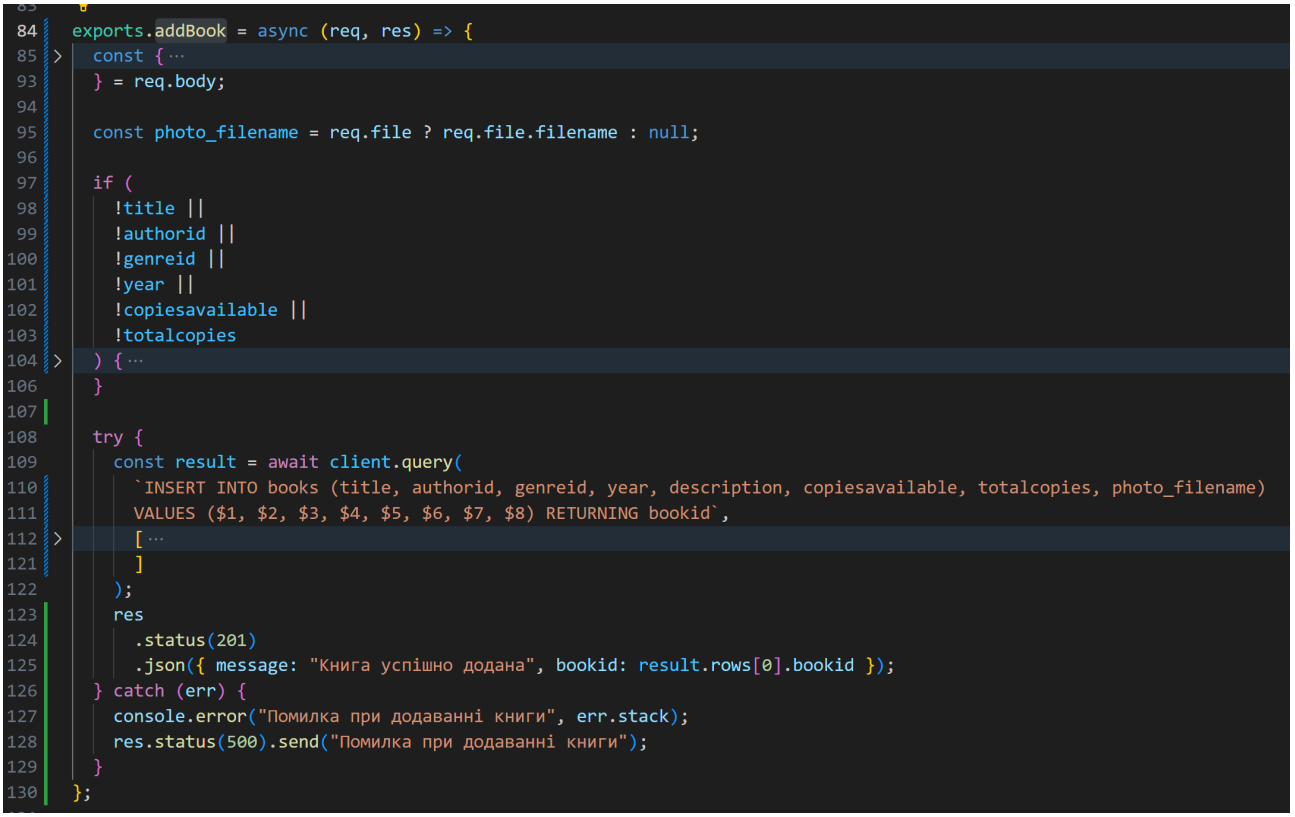
getBooks – функція, яка використовується для знаходження всіх книг.

Тут використовуються дані з кількох таблиць бази даних для перегляду повного об’єму інформації, що крім даних про книги, налічує авторів, видавництва та жанри.

```
exports.getBooks = async (req, res) => {
  const {
    authorid,
    genreid,
    publisherid,
    sortBy = "title",
    sortOrder = "asc",
  } = req.query;
  try {
    let query = `
      SELECT b.bookid, b.title AS book_title, a.authorid, a.firstname AS author_firstname, a.lastname AS author_lastname,
             p.name AS publisher_name, g.genreid, g.name AS genre_name, b.year, b.description AS book_description,
             b.copiesavailable, b.totalcopies, b.photo_filename AS book_photo
      FROM books b
      JOIN authors a ON b.authorid = a.authorid
      JOIN publishers p ON b.publisherid = p.publisherid
      JOIN genres g ON b.genreid = g.genreid
      WHERE 1 = 1`;
    const params = [];
    if (authorid) query += ` AND b.authorid = $${params.push(authorid)}`;
    if (genreid) query += ` AND b.genreid = $${params.push(genreid)}`;
    if (publisherid)
      query += ` AND b.publisherid = $${params.push(publisherid)}`;
    query += ` ORDER BY ${sortBy} ${sortOrder.toUpperCase()}`;
    res.json((await client.query(query, params)).rows);
  } catch (err) {
    console.error("Помилка при виконанні запиту", err.stack);
    res.status(500).send("Помилка при виконанні запиту до БД");
  }
};
```

Рис. 3.9. Функція getBooks

addBook наступний ендпойнт відповідає за внесення нових книг до системи бібліотеки. Ця функція записує дані тільки до таблиці books і, використовуючи зовнішні зв’язки до інших таблиць, прив’язує їх до бажаних авторів та жанрів.



```
exports.addBook = async (req, res) => {
  const {···
  } = req.body;

  const photo_filename = req.file ? req.file.filename : null;

  if (
    !title ||
    !authorid ||
    !genreid ||
    !year ||
    !copiesavailable ||
    !totalcopies
  ) {···
  }

  try {
    const result = await client.query(
      `INSERT INTO books (title, authorid, genreid, year, description, copiesavailable, totalcopies, photo_filename)
      VALUES ($1, $2, $3, $4, $5, $6, $7, $8) RETURNING bookid`,
      [···
      ]
    );
    res
      .status(201)
      .json({ message: "Книга успішно додана", bookid: result.rows[0].bookid });
  } catch (err) {
    console.error("Помилка при додаванні книги", err.stack);
    res.status(500).send("Помилка при додаванні книги");
  }
};
```

Рис. 3.10. Функція addBook

editBook має схожу логіку до занесення нових книг, але тут оновлюються вже готові дані.

Для того, щоб використати цей ендпойнт, необхідно вивести дані про книгу через особистий ідентифікатор, для цього застосовується вже реалізований getBookById, та далі, якщо книга зазнала змін, відправити їх на сервер для зберігання.

```
exports.editBook = async (req, res) => {
  const { id } = req.params;
  const {···
  } = req.body;

  const photo_filename = req.file ? req.file.filename : null;

  if ( !title || !authorid || !genreid || !year || !copiesavailable || !totalcopies
  ) {···
  }

  try {
    let query = `
      UPDATE books SET title = $1, authorid = $2, genreid = $3, year = $4, description = $5,
          copiesavailable = $6, totalcopies = $7`;

    const values = [···
    ];

    if (photo_filename) {···
    } else {···
    }

    const result = await client.query(query, values);

    if (result.rows.length === 0) {···
    }

    res.status(200).json({
      message: "Книга успішно відредагована",
      bookid: result.rows[0].bookid,
    });
  } catch (err) {···
  }
};
```

Рис. 3.11. Функція editBook

deleteBookById – це ендпойнт, що застосовується, якщо книги не існує в бібліотеці або потреба в її зберігання в системі не є актуальною.

```
exports.deleteBookById = async (req, res) => {
  const { id } = req.params;
  try {
    const result = await client.query(
      `DELETE FROM books WHERE bookid = $1 RETURNING bookid`,
      [id]
    );
    if (result.rows.length === 0) {
      return res.status(404).send("Книга не знайдена для видалення");
    }
    res.status(200).send(`Книга з id ${id} успішно видалена`);
  } catch (err) {
    console.error("Помилка при виконанні запиту", err.stack);
    res.status(500).send("Помилка при виконанні запиту до БД");
  }
};
```

Рис. 3.12. Функція deleteBookById

Для того, щоб повноцінно реалізувати addBook та editBook, необхідно надати можливість бачити додаткові таблиці адміністратору, бо внесення цих даних через ідентифікатори є не зручним та може нести за собою велику кількість помилок, що можна уникнути.

Це відбувається завдяки виведенню цих таблиць в окремих ендпойнтах. Тобто при заповненні або редагуванні даних буде виведено select з усіма можливими варіантами заповнень, далі з виведених назв обираються тільки ID та відправляються на відповідні маршрути звичайним INSERT.

У цьому випадку реалізуються getAuthors та getGenres, що виводять дані про всіх авторів та жанри, які доступні для заповнення в системі.

```
250  // Отримання авторів
251  exports.getAuthors = async (req, res) => {
252    try {
253      const result = await client.query(
254        "SELECT authorid, firstname, lastname FROM authors"
255      );
256      res.json(result.rows);
257    } catch (err) {
258      console.error("Помилка при виконанні запиту", err.stack);
259      res.status(500).send("Помилка при виконанні запиту до БД");
260    }
261  };
262
263  // Отримання жанрів
264  exports.getGenres = async (req, res) => {
265    try {
266      const result = await client.query("SELECT genreid, name FROM genres");
267      res.json(result.rows);
268    } catch (err) {
269      console.error("Помилка при виконанні запиту", err.stack);
270      res.status(500).send("Помилка при виконанні запиту до БД");
271    }
272  };
```

Рис. 3.13. Функції getAuthors та getGenres

3.3.3. Керування виданими та забронюваними книгами

Виведення та редагування даних щодо книг в бібліотеці є вагомим, але не центральним процесом. До головних елементів відносяться видача та бронювання видань, що й робить цю систему важливою та повноцінною.

Рис. 3.14. Маршрути, що використовуватимуться для керування резервуванням в адмін-панелі

getLoanHistory – це ендпойнт, що надає інформацію про всі операції в системі, які буди здійснені при видачі та поверненні книг бібліотеки. Тут використовуються дані з кількох таблиць та вся інформація сортується за датою видачі.

```
exports.getLoanHistory = async (req, res) => {
  try {
    const { rows } = await client.query(`
      SELECT
        l.loanid, l.loandate, l.duedate, l.returndate,
        u.lastname,
        b.title AS book_title
      FROM loans l
      JOIN reservations r ON l.reservationid = r.reservationid
      JOIN users u ON r.userid = u.userid
      JOIN copies c ON l.copyid = c.copyid
      JOIN books b ON c.bookid = b.bookid
      ORDER BY l.loandate DESC;
    `);
    res.json(rows);
  } catch (err) {
    console.error(err);
    res.status(500).send("Server Error");
  }
};
```

Рис. 3.15. Функція getLoanHistory

getActiveLoans є схожим до попереднього ендпойнту, виводивши інформацію про видачі книг, але з важливою відмінністю, тут надається інформація тільки про видачі, які є актуальними, тобто не були повернуті на даний момент.

Також цей ендпойнт застосовується при роботі з усіма видачами, надаючи додатковий функціонал для роботи адміністратора, що допоможе змінювати їх статус, видаляти їх та проводити інші дії.

```
exports.getActiveLoans = async (req, res) => {
  try {
    const { rows } = await client.query(`
        SELECT
          l.loanid, l.loandate, l.duedate, l.returndate,
          u.userid, u.email, u.lastname, u.email,
          b.bookid, b.title AS book_title, b.photo_filename,
          c.copyid, r.status, r.reservationId
        FROM loans l
        JOIN reservations r ON l.reservationid = r.reservationid
        JOIN users u ON r.userid = u.userid
        JOIN copies c ON l.copyid = c.copyid
        JOIN books b ON c.bookid = b.bookid
        WHERE l.returndate IS NULL;
      `);

    res.json(rows);
  } catch (err) {
    console.error(err);
    res.status(500).send("Server Error");
  }
};
```

Рис. 3.16. Функція getActiveLoans

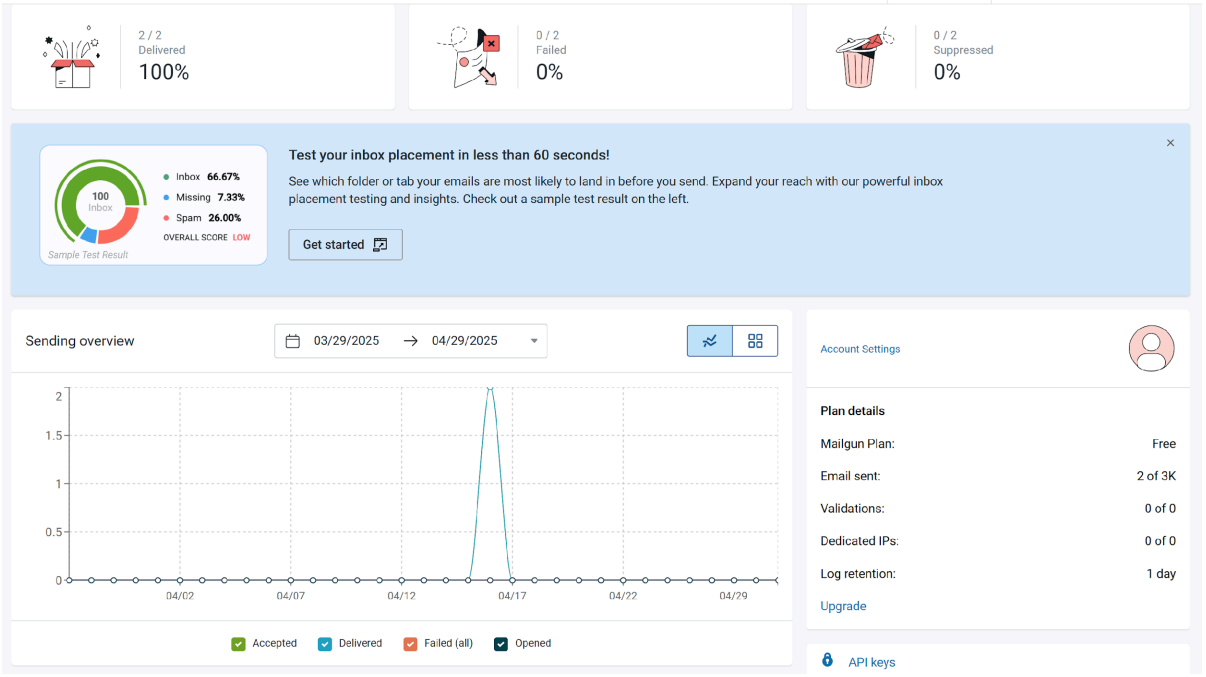
sendReturnReminders – це одна з таких функцій, які можна застосувати для активних бронювань. Коли термін здачі книги підходить до кінця, користувачу можна нагадати про це, відправивши листа на пошту з описом цього питання.

У коді задаються деталі повідомлення та дані для відправлення за допомогою спеціального сервісу.

```
exports.sendReturnReminders = async (req, res) => {
  let { email, book_title, duedate } = req.body;

  email = email.trim();

  const mailgun = new Mailgun(FormData);
  const mg = mailgun.client({
    username: "api",
    key: process.env.MAILGUN_API_KEY,
  });

  try {

    const msg = {
      from: "Library Reminder <postmaster@sandbox5631f76a62df4ba9a9b55584263af83c.mailgun.org>",
      to: [email],
      subject: "Нагадування про повернення книги",
      text: `Нагадуємо, що книгу "${book_title}" потрібно повернути до ${new Date(
        duedate
      ).toLocaleDateString("uk-UA")}.\n\nДякуємо!`,
    };

    const response = await mg.messages.create(
      "sandbox5631f76a62df4ba9a9b55584263af83c.mailgun.org",
      msg
    );

    console.log("Mailgun response:", response);
    res.json({ message: "Нагадування надіслано успішно." });
  } catch (error) {
    console.error("Mailgun error:", error);
    res.status(500).json({ message: "Помилка при надсиланні нагадування." });
  }
};
```

Рис. 3.17. Функція sendReturnReminders

Це організовується завдяки системі Mailgun, що дозволяє формувати спеціальний домен, який надає можливість відправляти ці повідомлення на бажані електронні пошти. [23]



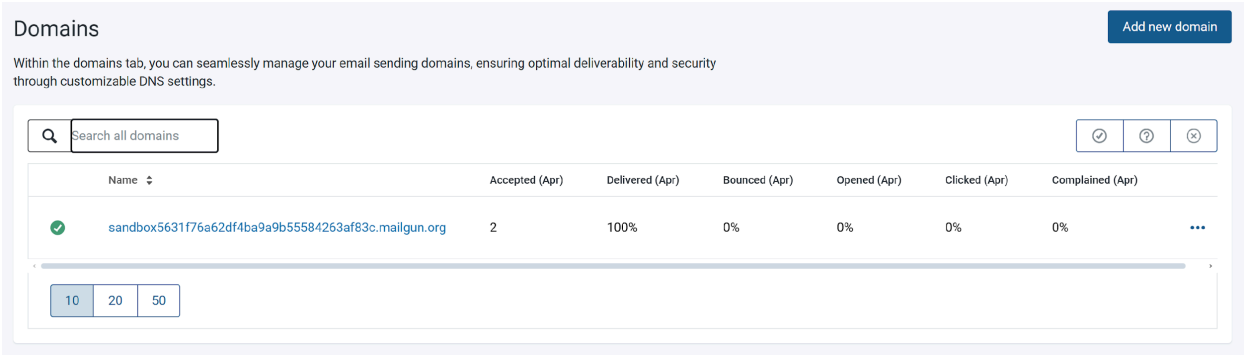


Рис. 3.18-19. Вигляд дашбордів системи Mailgun

adminCreateReservation – це останній реалізований ендпонт з файлу loanContoller, що надає можливість адміністратору вносити бронювання та видачі, одночасно оновлюючи кількість доступних копій книг для видачі. [24]

```
exports.adminCreateReservation = async (req, res) => {
  const { userId, bookId, dueDate } = req.body;
  try {
    const { rows: availableCopies } = await client.query(
      `SELECT copyid FROM copies WHERE bookid = $1 AND available = true LIMIT 1;`,
      [bookId]
    );
    if (availableCopies.length === 0) {···
    }
    const copyId = availableCopies[0].copyid;

    await client.query("BEGIN");
    const { rows } = await client.query(
      `INSERT INTO reservations (userid, reservationdate, status)
       VALUES ($1, CURRENT_DATE, 'active') RETURNING reservationid;`,
      [userId]
    );
    const reservationId = rows[0].reservationid;
    await client.query(
      `INSERT INTO loans (copyid, reservationid, loandate, duedate)
       VALUES ($1, $2, CURRENT_DATE, $3);`,
      [copyId, reservationId, dueDate]
    );

    await client.query(···
    );
    await client.query("COMMIT");
    res
      .status(201)
      .json({ message: "Бронювання створено адміністратором успішно" });
  } catch (err) {···
  }
};
```

Рис. 3.20. Функція adminCreateReservation

3.3.4. Реалізація статистики системи за допомогою графіків

Однією з характеристик, що найчастіше повторюється при створенні адмін-панелі є побудова статистики завдяки різним характеристикам системи.

Створені маршрути матимуть наступні найменування:

Рис. 3.21. Маршрути, що використовуватимуться для керування статистики в адмін-панелі

Для створення таких характеристик задаються запити, які виводять ці дані якраз-таки для бажаних характеристик.

Функції: getTopBooks, getTopUsers та getTopGenres, знаходять найпопулярніші книги, користувачів та жанри відповідно, сортуючи їх за необхідними параметрами.

```
exports.getTopBooks = async (req, res) => {
  try {
    const query = `
      SELECT b.title, COUNT(*) AS loan_count
      FROM loans l
      JOIN copies c ON l.copyid = c.copyid
      JOIN books b ON c.bookid = b.bookid
      GROUP BY b.title
      ORDER BY loan_count DESC
      LIMIT 10;
    `;
    const result = await client.query(query);
    res.json(result.rows);
  } catch (err) {···
  }
};
```

Рис. 3.22. Код функції getTopBooks

getBooksWithFewestCopies та getBooksToRestock знаходять книги, що мають найменші кількості копій, відповідно, їх перегляд слугуватиме поштовхом для того, щоб замовити їх у першу чергу.

```
exports.getBooksWithFewestCopies = async (req, res) => {
  try {
    const query = `
      SELECT b.title, COUNT(c.copyid) AS copies
      FROM books b
      LEFT JOIN copies c ON b.bookid  = c.bookid
      GROUP BY b.title
      ORDER BY copies ASC
      LIMIT 5;
    `;
    const result = await client.query(query);
    res.json(result.rows);
  } catch (err) {···
  }
};
```

Рис. 3.23. Код функції getBooksWithFewestCopies

Двома останніми функціями, що були реалізовані для панелі адміністратора є exportBooksToCSV та importBooksFromCSV, що відповідають за реалізацію наступних речей.

Перша з них є експортом всіх книг, які були занесені в систему, а друга навпаки є імпортом, що за допомогою CSV дозволить заносити книги до бібліотеки. [25]

```
137  exports.exportBooksToCSV = async (req, res) => {
138    try {
139      const query = `
140        SELECT b.bookid, b.title, b.year, a.firstname AS author_firstname, a.lastname AS author_lastname,
141               p.name AS publisher_name, g.name AS genre_name, b.totalcopies, b.copiesavailable FROM books b
142        JOIN authors a ON b.authorid = a.authorid
143        JOIN publishers p ON b.publisherid = p.publisherid
144        JOIN genres g ON b.genreid = g.genreid;
145      `;
146      const result = await client.query(query);
147      const parser = new Parser();
148      const csvData = "\uFEFF" + parser.parse(result.rows);
149
150      res.header("Content-Type", "text/csv");
151      res.attachment("books_export.csv");
152      return res.send(csvData);
153    } catch (err) {
154      console.error("Помилка при експорті CSV:", err);
155      res.status(500).send("Помилка сервера");
156    }
157  };
```

```
exports.importBooksFromCSV = async (req, res) => {
  const filePath = req.file.path;
  const books = [];

  try {
    fs.createReadStream(filePath)
      .pipe(csv())
      .on("data", (row) => {
        books.push(row);
      })
      .on("end", async () => {
        for (const book of books) {
          await client.query(
            `INSERT INTO books (title, year, authorid, publisherid, genreid, totalcopies, copiesavailable)
             VALUES ($1, $2, $3, $4, $5, $6, $7)
             ON CONFLICT (bookid) DO NOTHING`,
            [
              book.title,
              parseInt(book.year),
              parseInt(book.authorid),
              parseInt(book.publisherid),
              parseInt(book.genreid),
              parseInt(book.totalcopies),
              parseInt(book.copiesavailable),
            ]
          );
        }

        fs.unlinkSync(filePath);
        res.status(200).json({ message: "Імпорт завершено успішно" });
      });
  } catch (err) {···
  }
};
```

Рис. 3.24-25. Код функцій exportBooksToCSV та importBooksFromCSV

3.4. Тестування API

Для перевірки створеного коду буде використовуватись Postman та ручне тестування на самому вебсайті.

Щоб використати Postman, необхідно вибрати маршрут та налаштувати дані для його тестування. Далі наводяться кілька ендпойнтів, що були реалізовані для загальної роботи сайту. [26]

Найперше необхідна перевірка коректного отримання інформації про користувача за id, це буде використовуватись в профілі для виведення загальної інформації.

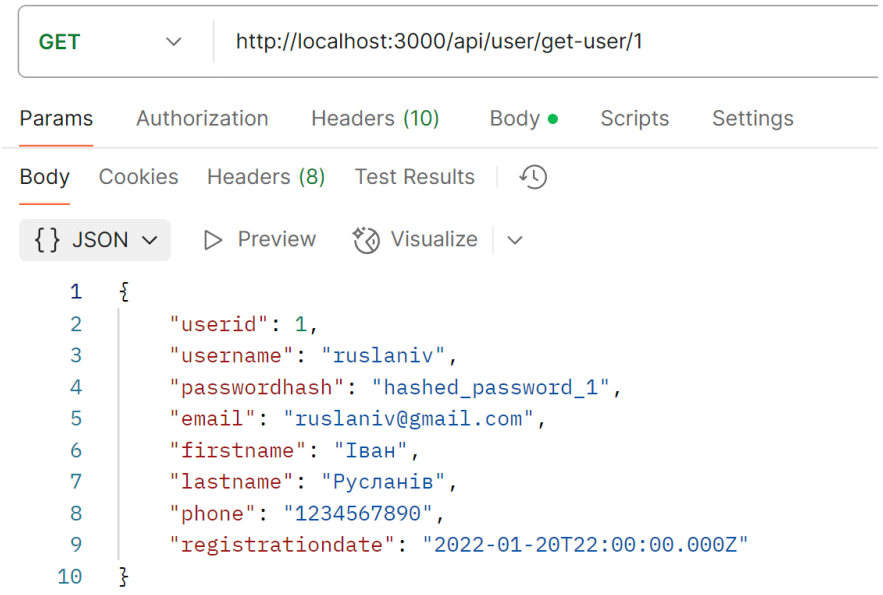


Рис. 3.26. Пошук даних користувача

Після перевірок, які відносяться до користувача та його даних, демонструється пошук детальної інформації про книгу за наданим їй ідентифікатором.

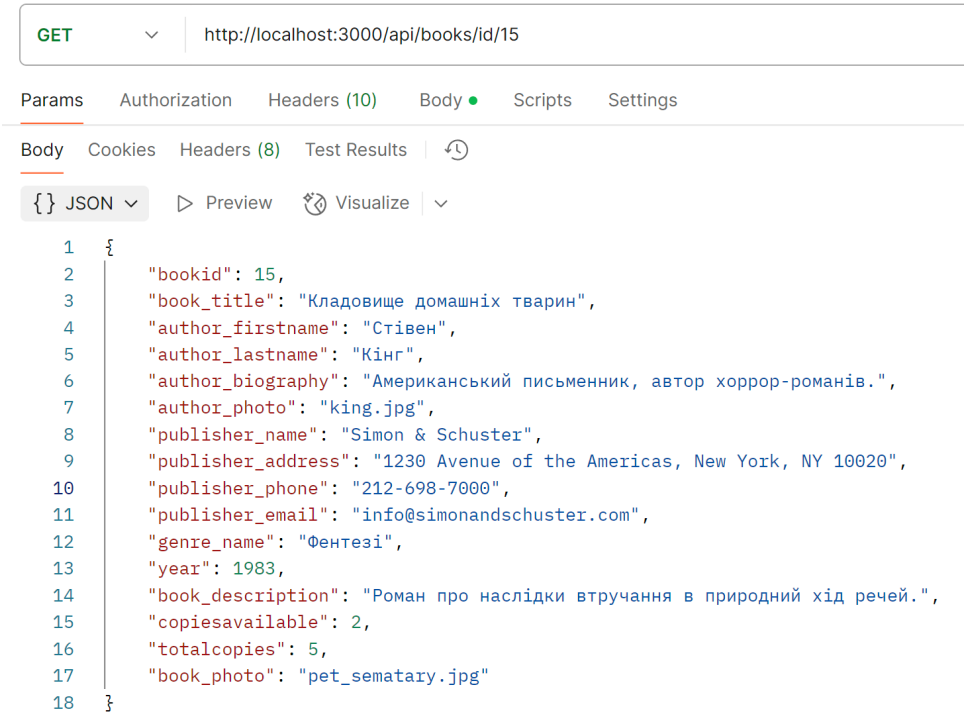


Рис. 3.27. Виведення інформації про книгу за ID

Також важливим елементом під час користування сайту є сортування книг за певними заданими параметрами. У випадку розробки проєкту для фільтрування це буде автор та жанр, а сортування відбувається по значеннях року та назві книги, в різному порядку.

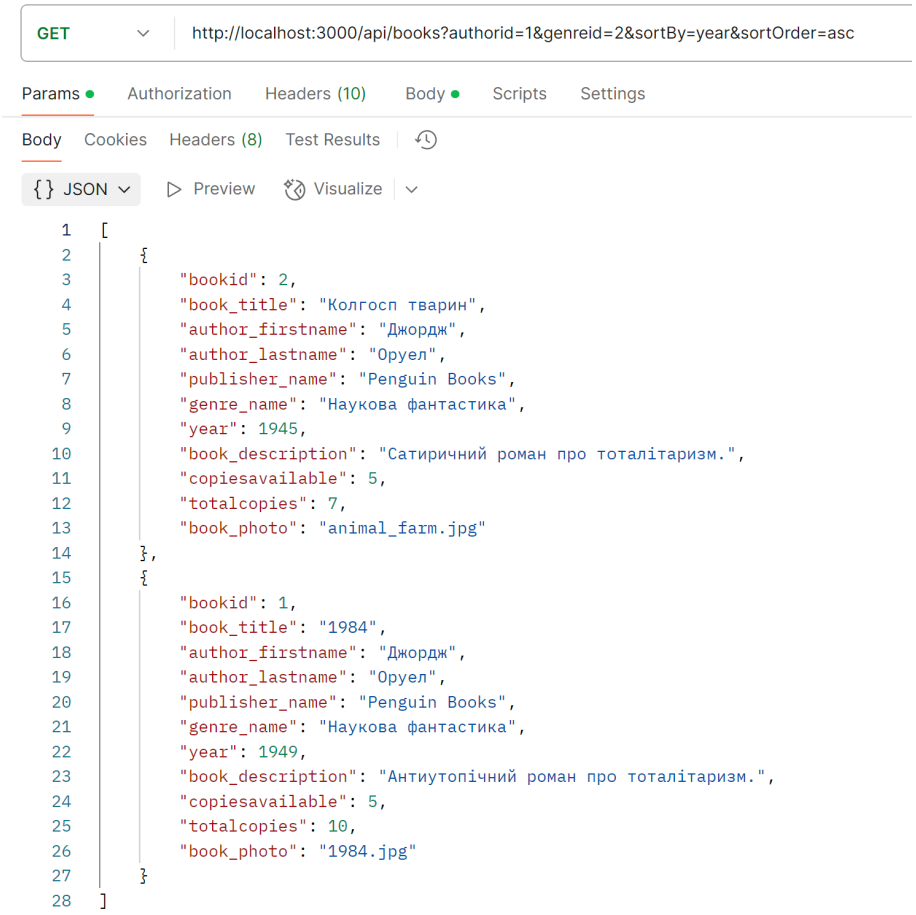


Рис. 3.28. Сортування книг за фільтрами

Для перевірки роботи адмін-панелі необхідно ввійти на сайт, аутентифікуватись адміністратором та перевірити її роботу.

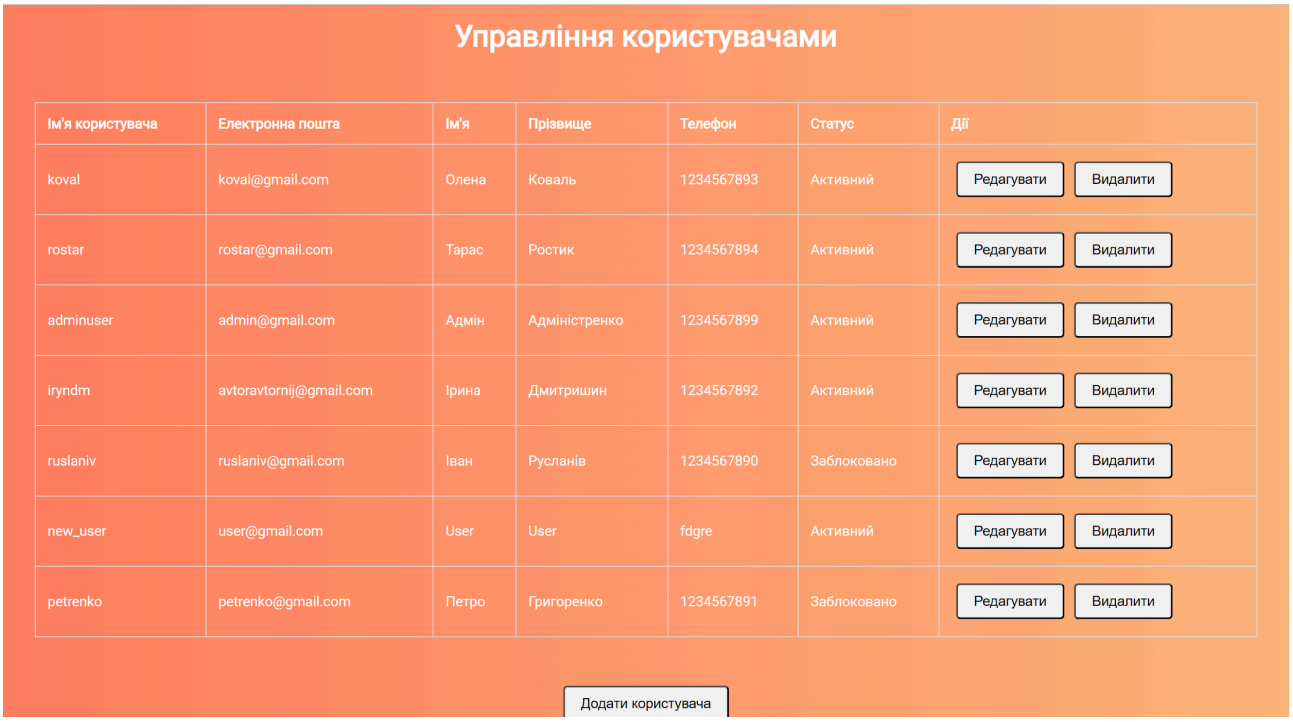


Рис. 3.29. Вигляд всіх користувачів

Тут також можна заблокувати користувача, що при спробі входу до цього профілю, виводитиме наступне сповіщення: [27]

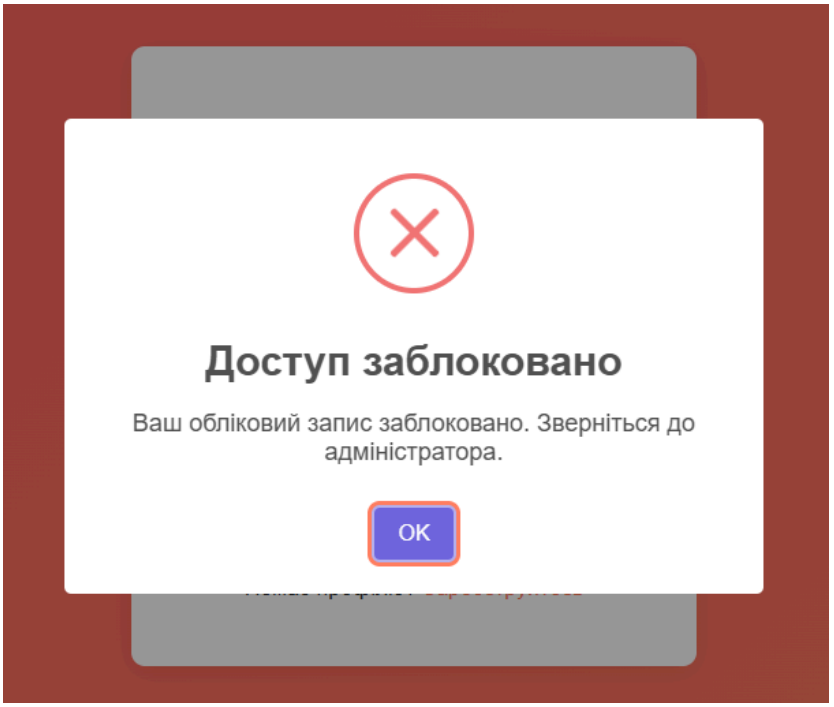


Рис. 3.30. Спроба входу заблокованого користувача

Далі було переглянуто сторінку для керування книгами.

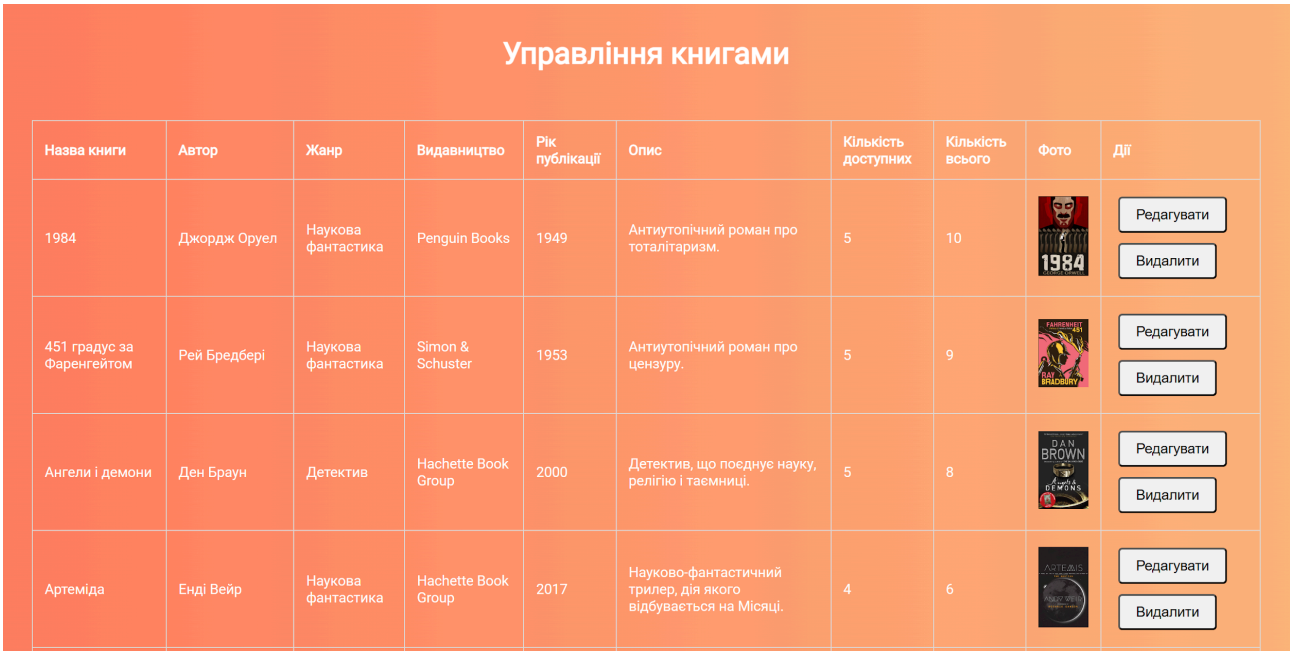
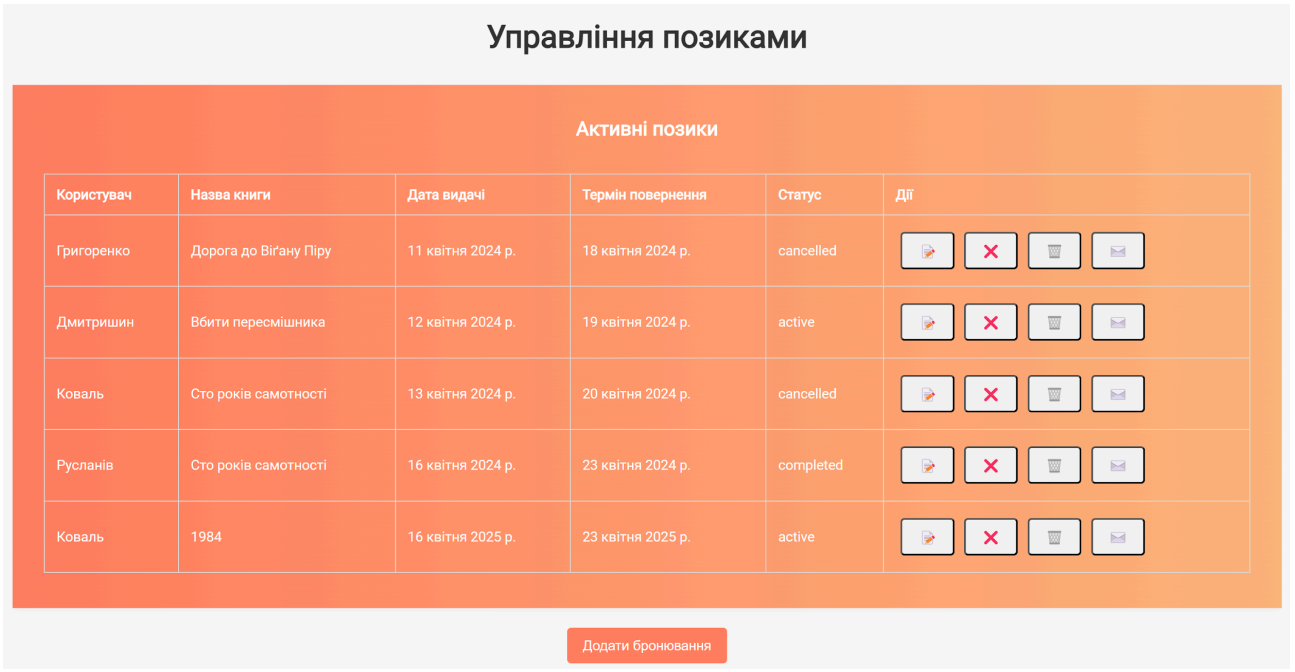


Рис. 3.31. Сторінка керувань книгами системи

Робота з резервуваннями в адмін-панелі виглядає наступним чином: тут можна змінювати статус, видаляти позики повністю та відправляти сповіщення прямо на пошту.



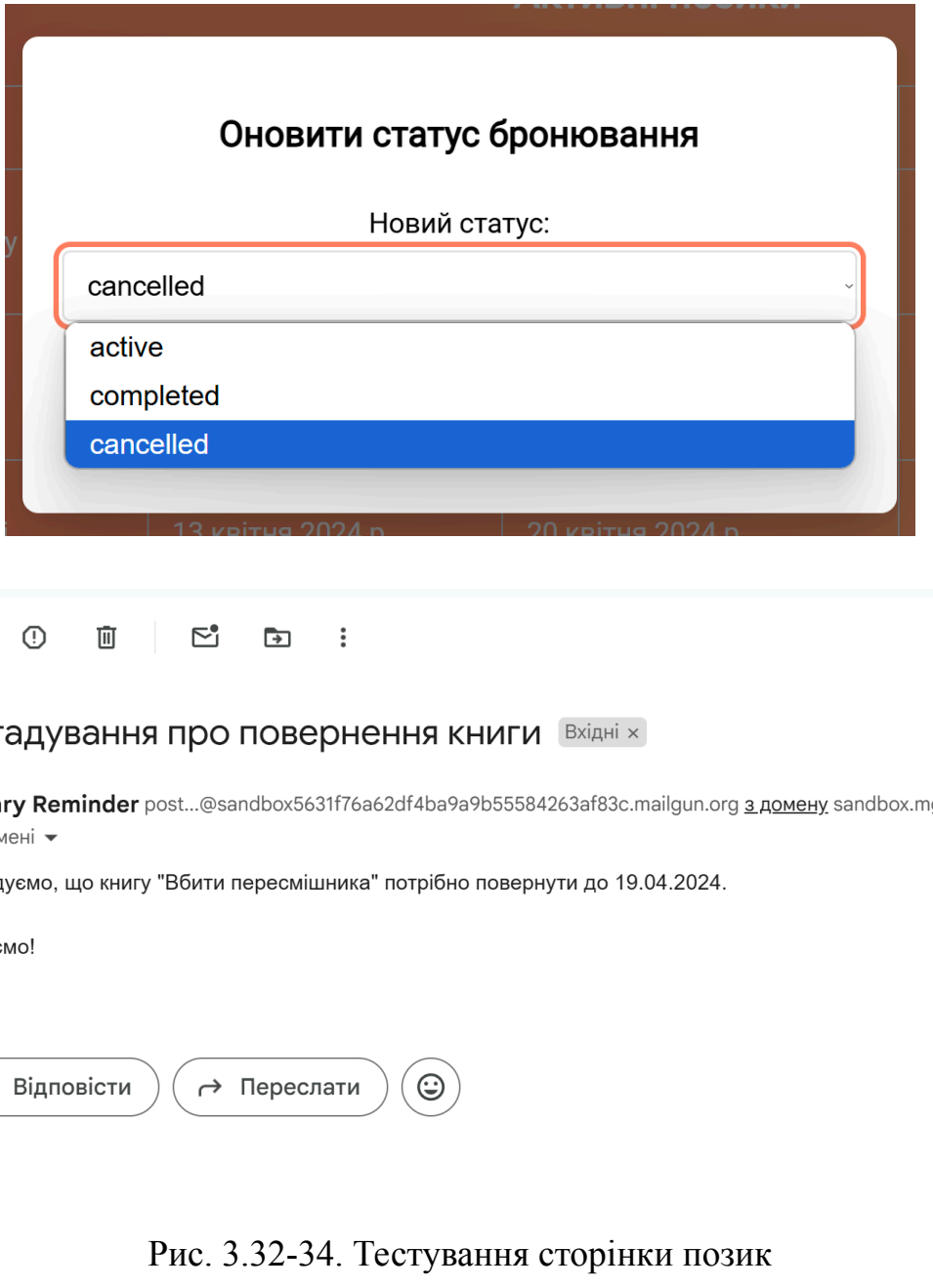


Рис. 3.32-34. Тестування сторінки позик

Останнім елементом панелі є реалізовані графіки, що позначають загальну статистику.

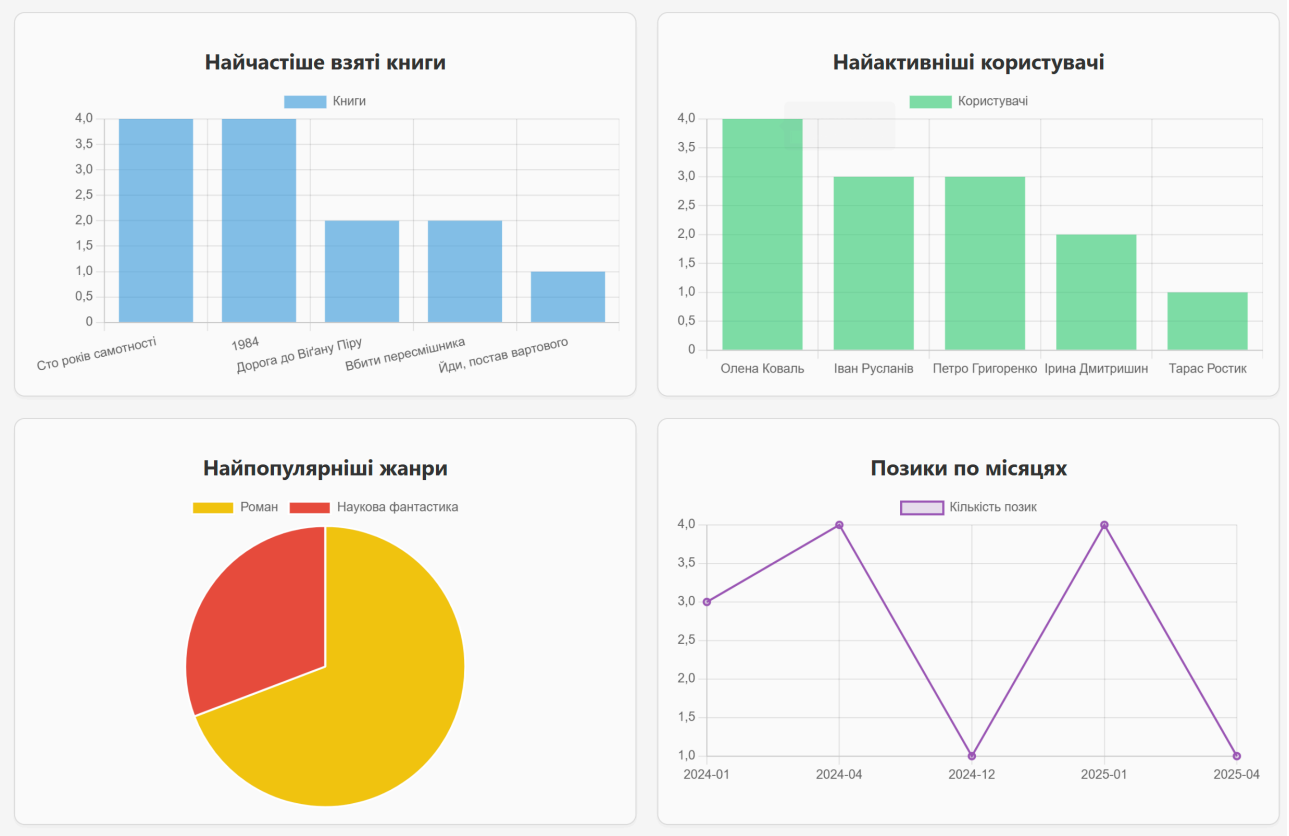


Рис. 3.35 Сторінка статистики системи

Решту створених функцій та додаткових елементів адмін-панелі наведено в додатках.

Висновок до розділу 3

На завершальному етапі розробки було створено одну з найважливіших частин – адміністративну панель, що забезпечує зручне керування всіма елементами цифрової платформи, через вебінтерфейс, без необхідності безпосереднього доступу до бази даних..

Додатково було побудовано ряд таких функцій: редагування даних користувачів, управління книгами, бронювання, надсилання сповіщень на пошту та побудова загальних статистичних графіків, а також, було впроваджено функції імпорту та експорту у форматі CSV.

Для перевірки працездатності системи, було здійснено тестування API через Postman та ручне тестування через вебінтерфейс. За результатами

тестування система продемонструвала стабільну роботу та коректну обробку запитів. Таким чином, поставлену мету цього етапу - створення ефективного інструменту адміністрування - було повністю досягнуто.

ВИСНОВКИ

У межах проєкту було створено веб-систему бібліотеки, яка надає користувачам можливість здійснювати пошук книг, переглядати каталоги, бронювати літературу та зберігати важливу інформацію у власному обліковому записі. Крім того, досягнуто головної мети - реалізовано функціональну адміністративну панель, яка надає можливість ефективного керування даними користувачів, книг, процесами бронювання та перегляду статистичних звітів.

Даних результатів можна було отримати, поступово виконавши етапи проєктування системи:

- Аналіз предметної області, в межах якого було визначено актуальності і формулювання наступних завдань для технічного процесу побудови;

- Виокремлення сутностей та їх зв’язків, і відповідна розробка ER-діаграми на їх основі із здійсненням нормалізації;

- Побудова серверної частини з Node.js, застосовуючи Express, і таким чином визначивши API-маршрути;

- Реалізовано адміністративну панель, яка надає змогу адміністратору опрацьовувати дані користувачів, книг та бронювань, і отримувати графіки загальної статистики, а також CRUD-операції для книг і користувачів, що дозволяє вносити зміни, оновлювати або видаляти інформацію. Модуль керування бронюванням надає можливість не лише бачити активні видачі, а й надсилати нагадування про повернення книг;

- Розроблено функціонал для імпорту та експорту даних у форматі CSV;

- Перевірено систему через інструмент тестування API, а саме Postman, та безпосередньо через вебінтерфейс, звідки можна було переконатись, що платформа працює стабільно та відповідно до функціональних вимог.

Власним внеском цього проєкту є детальна побудова бекенд частини та адміністративної панелі. Результати розробки можуть бути використані для

подальшого вдосконалення подібних інформаційних систем в інших навчальних, культурних або освітніх установах задля покращення доступу до знань та цифровізації бібліотечних процесів.

Отже, дане інноваційне впровадження – це не тільки про покращення управлінських процесів для суттєвого підвищення ефективності в наданні послуг у сфері бібліотечної діяльності, проте найважливіше – це про вагомий внесок щодо збереження та розвитку культурної спадщини поколінь.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Fllstck. Why Do You Need a Backend? [Електронний ресурс]. – Режим доступу: https://dev.to/fllstck/why-do-you-need-a-backend-475d – Дата звернення: 01.05.2025.

2. What’s the Difference Between Frontend and Backend? – CareerFoundry. [Електронний ресурс]. – Режим доступу: https://careerfoundry.com/en/blog/web-development/whats-the-difference-between-frontend-and-backend/ – Дата звернення: 01.05.2025.

3. Про бібліотеки і бібліотечну справу: Закон України № 32/95-ВР від 27 січня 1995 р. // Відомості Верховної Ради України. 1995. № 7. Ст. 45. [Електронний ресурс]. – Режим доступу: http://zakon4.rada.gov.ua/laws/show/32/95-вр/conv – Дата звернення: 01.05.2025.

4. Освітній портал імені Андрея Шептицького. [Електронний ресурс]. – Режим доступу: https://sheptytskyi.osvitportal.in.ua/Zdobuvacham-osvity-4/osv-148 – Дата звернення: 01.05.2025.

5. Digital Library Federation (DLF). [Електронний ресурс]. – Режим доступу: https://www.diglib.org – Дата звернення: 01.05.2025.

6. Introduction to PostgreSQL – Guru99. [Електронний ресурс]. – Режим доступу: https://www.guru99.com/uk/introduction-postgresql.html – Дата звернення: 01.05.2025.

7. Тестування API з Postman – Testmatick. [Електронний ресурс]. – Режим доступу: https://testmatick.com/uk/testuvannya-api-z-postman/ – Дата звернення: 01.05.2025.

8. Gleek. Database entity: How to structure your data. [Електронний ресурс]. – Режим доступу: https://www.gleek.io/blog/database-entity – Дата звернення: 01.05.2025.

9. TutorialsPoint. Conceptual Database Design. [Електронний ресурс]. – Режим доступу: https://www.tutorialspoint.com/conceptual-database-design – Дата звернення: 01.05.2025.

10. Авраменко В. С., Авраменко А. С. Проектування інформаційних систем: навчальний посібник. – Черкаси: Черкаський національний університет ім. Б. Хмельницького, 2017. – 434 с.: іл.

11. JavaRush. Рівень 17, лекція 2 – Робота з Hibernate. [Електронний ресурс]. – Режим доступу: https://javarush.com/ua/quests/lectures/ua.questhibernate.level17.lecture02 – Дата звернення: 01.05.2025.

12. Mozilla Developer Network. Development environment for Express (Node.js). [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/development_environment – Дата звернення: 01.05.2025.

13. Amazon Web Services. What is a RESTful API? [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/what-is/restful-api/ – Дата звернення: 01.05.2025.

14. CRUD REST API with Node.js, Express, and PostgreSQL – LogRocket Blog. [Електронний ресурс]. – Режим доступу: https://blog.logrocket.com/crud-rest-api-node-js-express-postgresql/ – Дата звернення: 01.05.2025.

15. Node.js. About Node.js. [Електронний ресурс]. – Режим доступу: https://nodejs.org/en/about – Дата звернення: 01.05.2025.

16. PostgreSQL Global Development Group. About PostgreSQL. [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/about/ – Дата звернення: 01.05.2025.

17. GeeksforGeeks. Routing in Node.js. [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/routing-in-node-js/ – Дата звернення: 01.05.2025.

18. Mozilla Developer Network. Express routing. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn_web_development/Extensions/Server-side/Express_Nodejs/routes – Дата звернення: 01.05.2025.

19. GeeksforGeeks. Node.js Export Module. [Електронний ресурс]. – Режим доступу: https://www.geeksforgeeks.org/node-js-export-module/ – Дата звернення: 01.05.2025.

20. W3Schools. Node.js MySQL Update. [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/nodejs/nodejs_mysql_update.asp – Дата звернення: 01.05.2025.

21. W3Schools. Node.js MySQL Delete. [Електронний ресурс]. – Режим доступу: https://www.w3schools.com/nodejs/nodejs_mysql_delete.asp – Дата звернення: 01.05.2025.

22. Express.js. Multer middleware for handling multipart/form-data. [Електронний ресурс]. – Режим доступу: https://expressjs.com/en/resources/middleware/multer.html – Дата звернення: 01.05.2025.

23. Mailgun. Email API Service. [Електронний ресурс]. – Режим доступу: https://www.mailgun.com – Дата звернення: 01.05.2025.

24. MySQL 8.4 Reference Manual. COMMIT statement. [Електронний ресурс]. – Режим доступу: https://dev.mysql.com/doc/refman/8.4/en/commit.html – Дата звернення: 01.05.2025.

25. CloudSigma Blog. Managing CSV in Node.js using node-csv. [Електронний ресурс]. – Режим доступу: https://blog.cloudsigma.com/managing-csv-in-node-js-using-node-csv/ – Дата звернення: 01.05.2025.

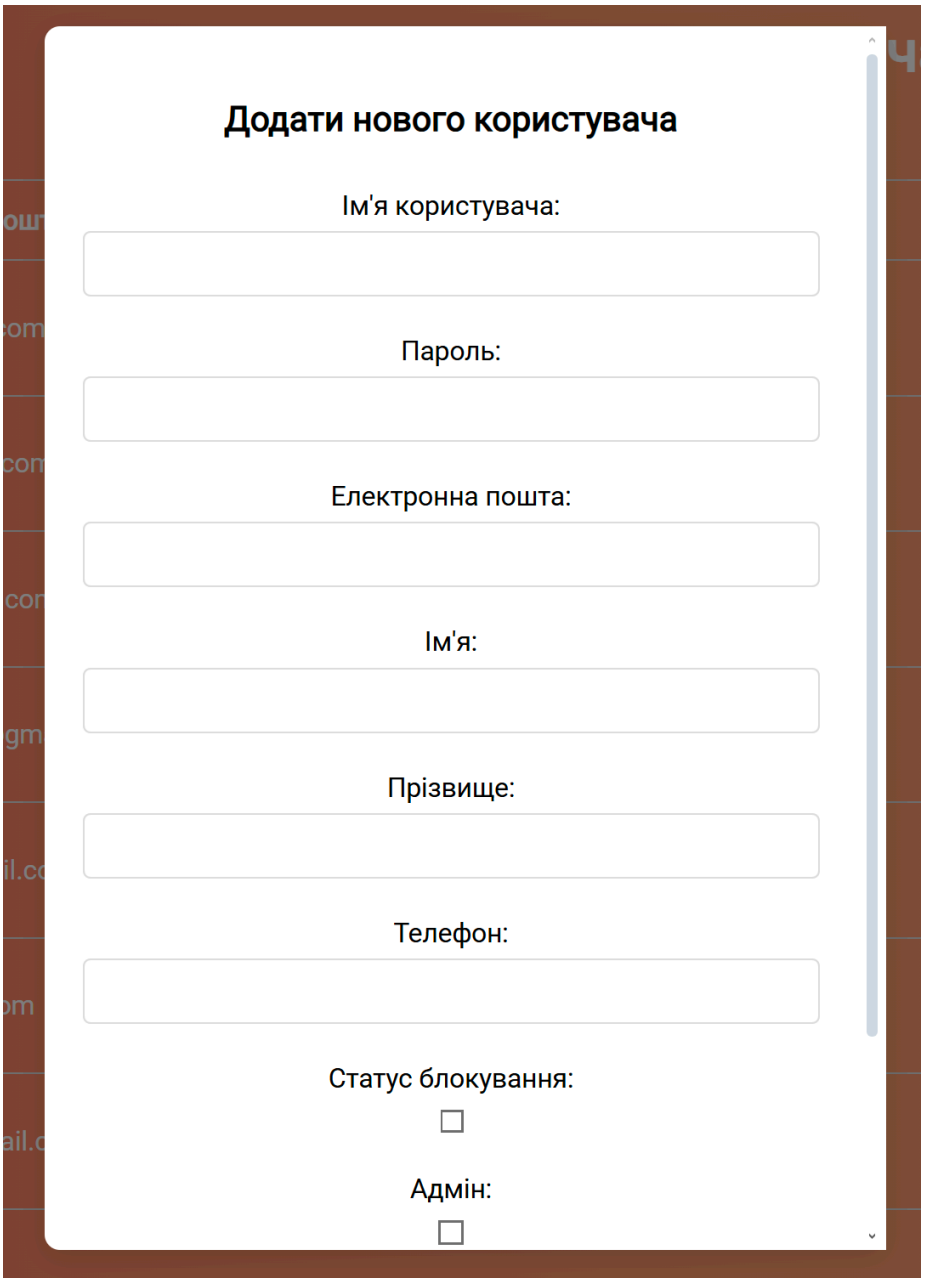
26. Postman. What is Postman? [Електронний ресурс]. – Режим доступу: https://www.postman.com/product/what-is-postman/ – Дата звернення: 01.05.2025.

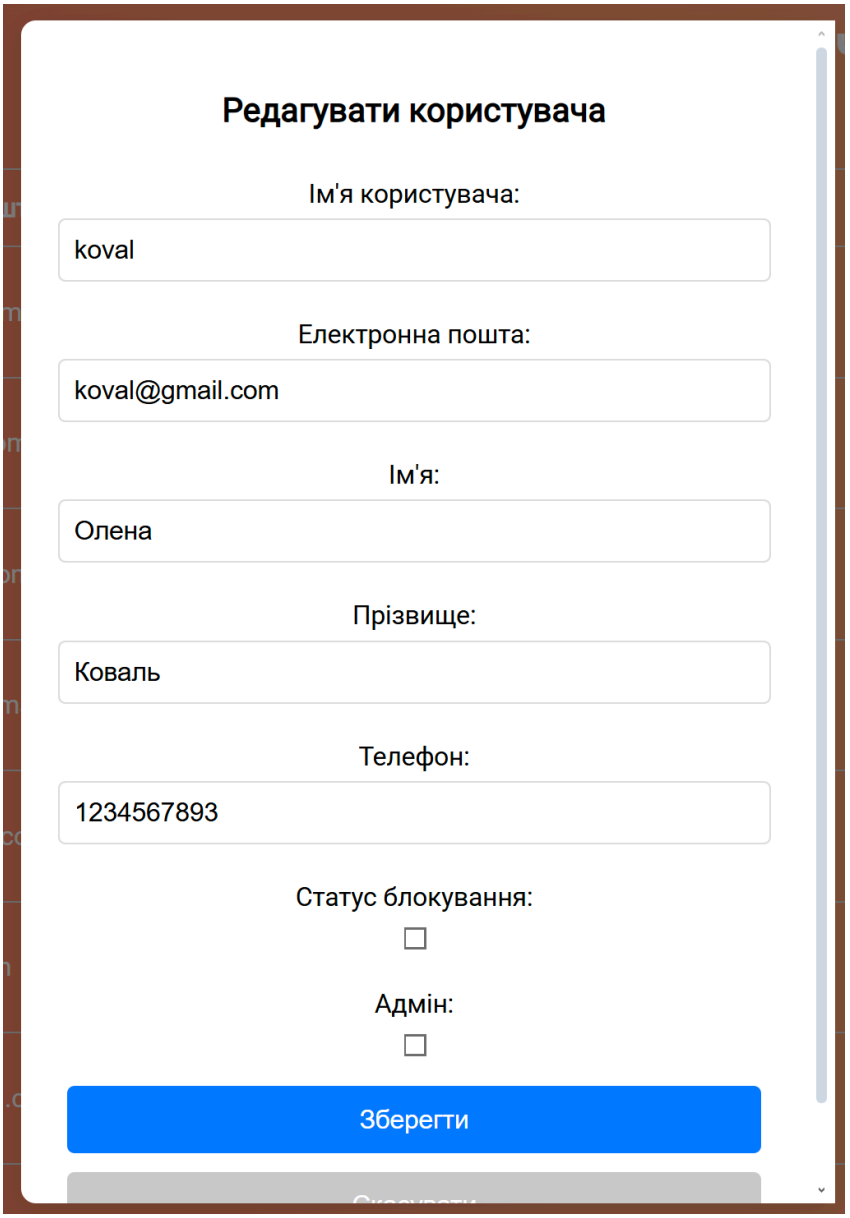
27. Kominfo Kepulauan Riau. SweetAlert CSS resource. [Електронний ресурс]. – Режим доступу: https://kominfo.kepriprov.go.id/beta/assets/admin/css/lib/sweetalert/ – Дата звернення: 01.05.2025.

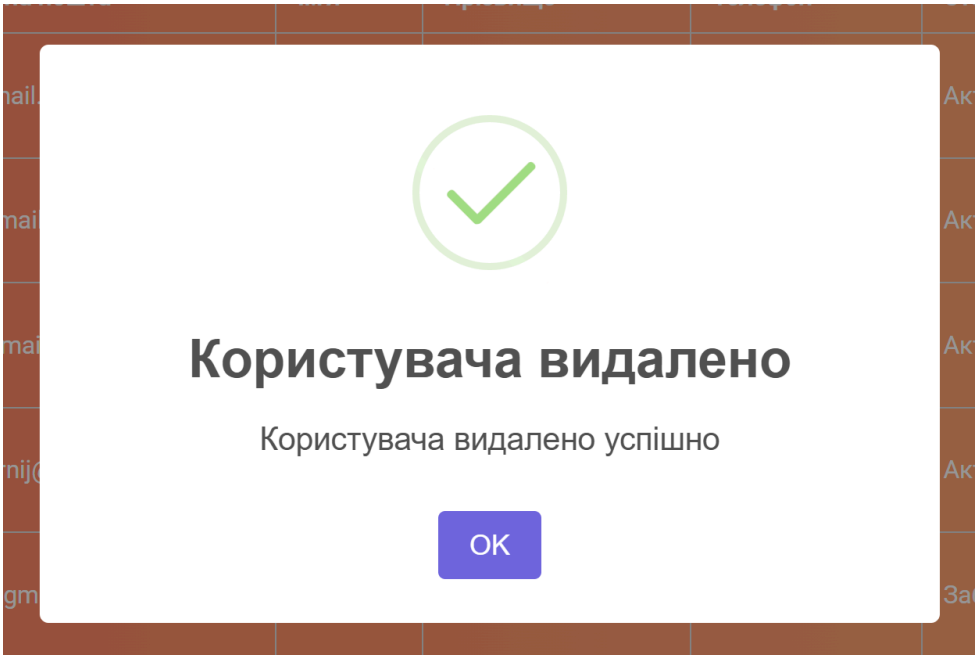
ДОДАТКИ

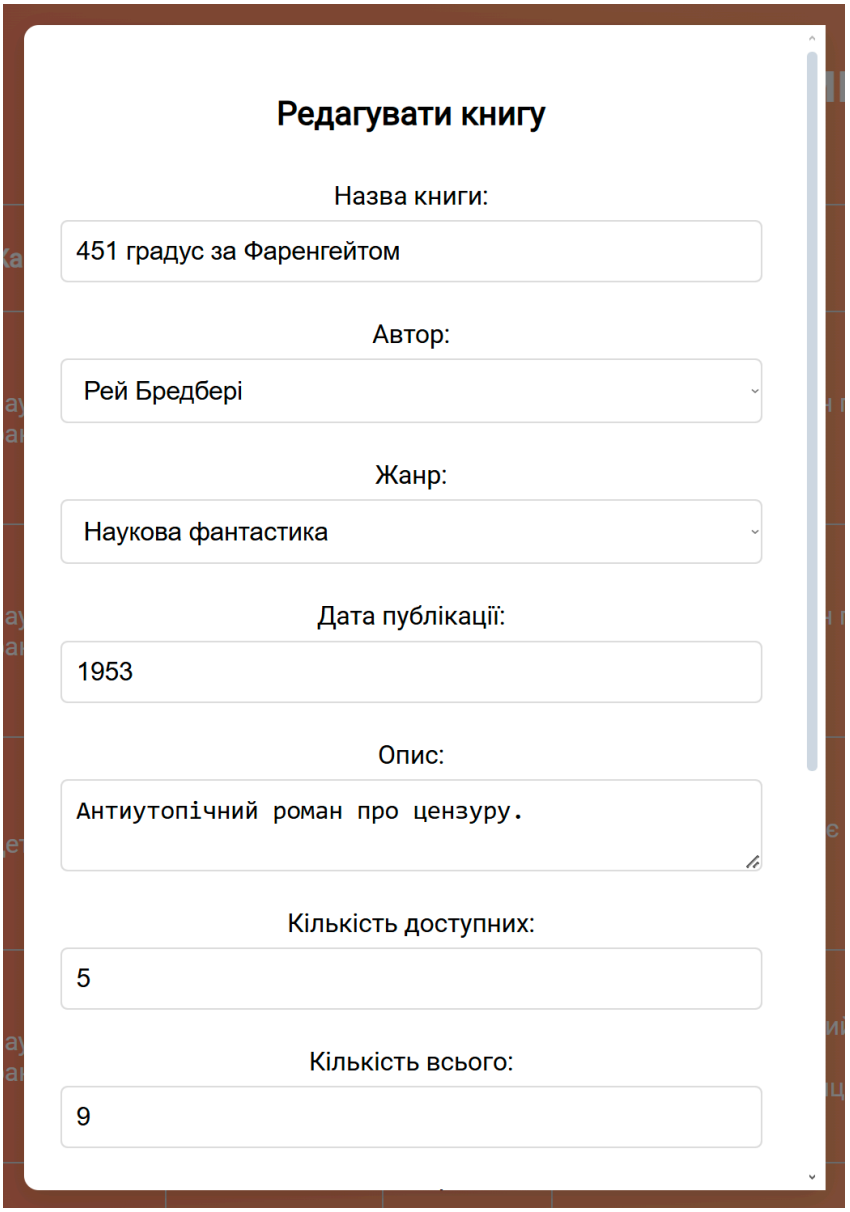
Додаток А

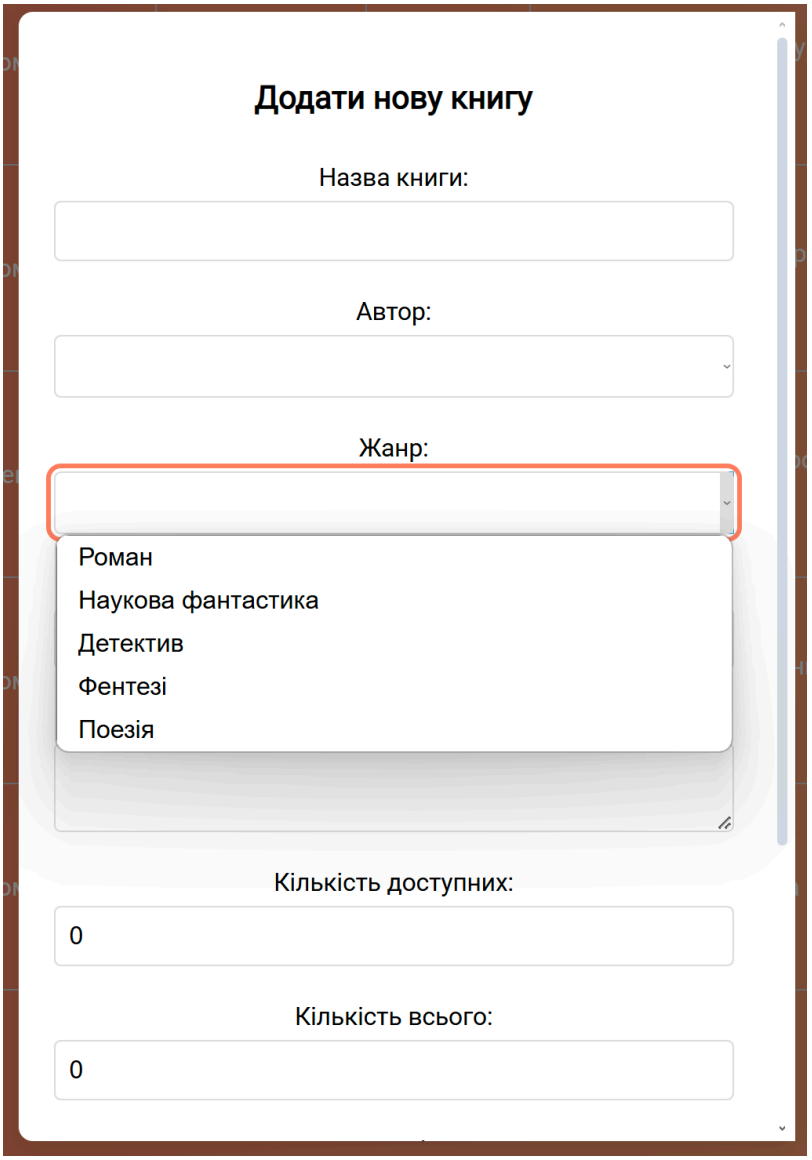
Рисунки додаткових функцій адмін-панелі:

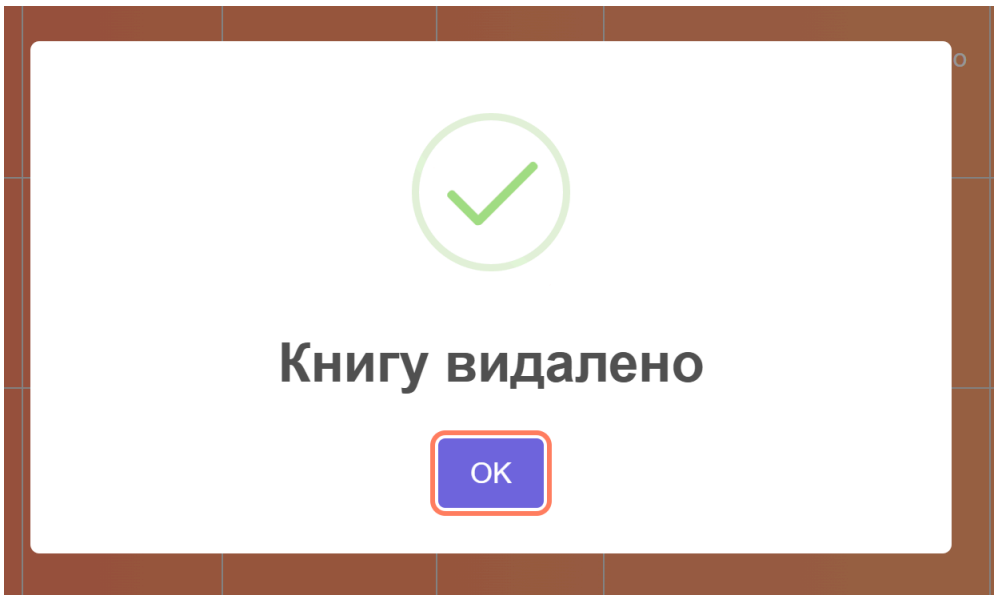



Редагувати користувача
Ім'я користувача:
koval
Електронна пошта:
koval@gmail.com
Ім'я:
Олена
Прізвище:
Коваль
Телефон:
1234567893
Статус блокування:
Адмін:
Зберегти


Користувача видалено
Користувача видалено успішно
OK


Редагувати книгу
Назва книги:
451 градус за Фаренгейтом
Автор:
Рей Бредбері
Жанр:
Наукова фантастика
Дата публікації:
1953
Опис:
Антиутопічний роман про цензуру.
Кількість доступних:
5
Кількість всього:
9


Додати нову книгу
Назва книги:
Автор:
Жанр:
Роман
Наукова фантастика
Детектив
Фентезі
Поезія
Кількість доступних:
0
Кількість всього:
0


Книгу видалено
OK

Історія позик

Користувач	Назва книги	Дата видачі	Дата повернення
Коваль	1984	16 квітня 2025 р.	1 січня 1970 р.
Русланів	1984	19 січня 2025 р.	26 січня 2025 р.
Ростик	1984	18 січня 2025 р.	25 січня 2025 р.
Григоренко	1984	17 січня 2025 р.	24 січня 2025 р.
Коваль	Сто років самотності	16 січня 2025 р.	23 січня 2025 р.
Русланів	Йди, постав вартового	6 грудня 2024 р.	13 грудня 2024 р.
Русланів	Сто років самотності	16 квітня 2024 р.	1 січня 1970 р.
Коваль	Сто років самотності	13 квітня 2024 р.	1 січня 1970 р.
Дмитришин	Вбити пересмішника	12 квітня 2024 р.	1 січня 1970 р.
Григоренко	Дорога до Віґану Піру	11 квітня 2024 р.	1 січня 1970 р.
Коваль	Сто років самотності	18 січня 2024 р.	25 січня 2024 р.
Дмитришин	Вбити пересмішника	17 січня 2024 р.	24 січня 2024 р.
Григоренко	Дорога до Віґану Піру	16 січня 2024 р.	23 січня 2024 р.

📚 Середня кількість копій на одну книгу:

1.52

📉 Книги з найменшою кількістю копій

- Пармський монастир – 0 копій
- Ангели і демони – 0 копій
- Віднесені вітром – 0 копій
- Втрачений символ – 0 копій
- Хороші дружини – 0 копій

📦 Рекомендовані до поповнення

- Пармський монастир – 0 копій
- Проєкт «Радуйся, Маріє» – 0 копій
- Ангели і демони – 0 копій
- Хороші дружини – 0 копій
- Віднесені вітром – 0 копій
- Маленькі жінки – 0 копій
- Скарлетт – 0 копій
- Червоний і чорний – 0 копій
- Марсіанин – 0 копій
- Артеміда – 0 копій
- Втрачений символ – 0 копій
- Любов під час холери – 2 копій
- Дорога до Віґану Піру – 2 копій
- Розум і почуття – 2 копій
- Ніч лагідна – 2 копій