Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

Галузь знань 12 – Інформаційні технології

Спеціальність 126 –Інформаційні системи та технології

на тему: «Метод та веб-орієнтована інформаційна система бібліотеки»

КвРІСТ. 240172.24.01.02 ПЗ

Виконав: студент 2 курсу, група ІСТм-24-1 _____ Олег ВОЛОШИН
Підпис Ініціали, прізвище

Керівник: кандитат техн. наук, доцент _____ Дмитро МЕДЗАТИЙ
Науковий ступінь, вчене звання Підпис Ініціали, прізвище

До захисту допускаю:
Зав. кафедри КІІС,
PhD Ольга ПАВЛОВА
02 12 2025 р.

Хмельницький, 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень магістр

Галузь знань 12 Інформаційні технології

Спеціальність 126 Інформаційні системи та технології

Освітня програма Освітньо-професійна програма «Інформаційні системи та технології»

ЗАТВЕРДЖУЮ
Зав. кафедри Ольга ПАВЛОВА
" 25 " 08 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ МАГІСТРА

Олегу ВОЛОШИНУ
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Метод та веб-орієнтована інформаційна система бібліотеки
Керівник проекту (роботи) Дмитро МЕДЗАТИЙ, к.т.н., доцент.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання
Затверджена наказом ректора університету від 25.08.2025 р. № 65
2. Строк подання студентом проекту (роботи) на кафедру 01.12.2025 р.
3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз відомих методів для оцінювання відповідності інтерфейсу гештальт-принципам
Моделювання процесу оцінювання відповідності інтерфейсу гештальт-принципам
Метод оцінювання відповідності інтерфейсу гештальт-принципам
Інформаційна технологія оцінювання відповідності інтерфейсу гештальт-принципам
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

6. Консультанти розділів кваліфікаційної роботи магістра

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Сергій ЛИСЕНКО, професор кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «_01_»_____09_____2025р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	01.09.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	15.09.2025	виконано
3	Робота над розділом 1 – огляд відомих рішень	01.10.2025	виконано
4	Робота над розділом 2 – проєктування інформаційної системи бібліотеки	15.10.2025	виконано
5	Робота над науковою публікацією	15.10.2025	виконано
6	Робота над розділом 3 – метод та алгоритм реалізації веб-орієнтованої інформаційної системи бібліотеки	01.11.2025	виконано
7	Робота над розділом 4 – інформаційна система бібліотеки та її програмна реалізація	15.11.2025	виконано
8	Оформлення пояснювальної записки згідно вимог	01.12.2025	виконано
9	Попередній захист ВКР	02.12.2025	виконано
10	Захист ВКР на засіданні ЕК	19.12.2025	

Студент _____ Олег ВОЛОШИН
Підпис Ініціали, прізвище

Керівник роботи _____ Дмитро МЕДЗАТИЙ
Підпис Ініціали, прізвище

РЕФЕРАТ

Тема кваліфікаційної роботи магістра: Метод та веб-орієнтована інформаційна система бібліотеки.

Автор роботи: Олег ВОЛОШИН

Керівник роботи: к.т.н., доцент, Дмитро МЕДЗАТИЙ

Пояснювальна записка: 70 с., 21 рис., 2 дод., 84 джерел.

Перелік ключових слів: веб-орієнтована інформаційна система, цифрова бібліотека, пошуковий модуль, алгоритм кешування, повнотекстовий пошук, модуль позик і бронювання, NestJS, PostgreSQL, Elasticsearch, Redis, React.

Об’єктом дослідження є процес опрацювання, збереження та пошуку інформації в автоматизованих бібліотечних системах.

Предметом дослідження є метод та веб-орієнтована інформаційна система бібліотеки, що забезпечує автоматизацію процесів керування фондами, позиками, бронюваннями, формуванням статистики та забезпеченням зручного пошуку літератури.

Метою кваліфікаційної роботи магістра є розроблення методу та веб-орієнтованої інформаційної системи бібліотеки, яка забезпечує ефективне управління бібліотечними ресурсами, швидкий доступ до електронного каталогу, повнотекстовий пошук із використанням індексованих структур даних, а також автоматизацію ключових операцій бібліотечного обслуговування.

Для розв’язання поставлених задач використовувались положення системного аналізу, методи структурного та об’єктно-орієнтованого моделювання, теорія побудови інформаційних систем, алгоритмічні засади оптимізації запитів, моделі взаємодії клієнт–сервер та принципи побудови розподілених систем. Під час моделювання логіки роботи системи застосовано UML, BPMN-нотації, методи концептуального та логічного проектування баз даних, принципи кешування, індексування та ранжування пошукових результатів.

Наукова новизна отриманих результатів полягає в удосконаленому методі побудови веб-орієнтованої бібліотечної системи, який відрізняється поєднанням

алгоритмів повнотекстового пошуку, кешування даних і динамічного ранжування результатів для підвищення точності й швидкодії. Набула подальшого розвитку інформаційна технологія автоматизованого управління бібліотечним фондом, що забезпечує інтеграцію пошукового модуля з індексованим сховищем даних та механізмами оперативного оновлення інформації в реальному часі.

На основі проведених досліджень розроблена архітектура і компоненти програмного забезпечення веб-орієнтованої інформаційної системи бібліотеки, що включають серверну підсистему на основі NestJS, клієнтську частину на React, реляційне сховище PostgreSQL, модуль повнотекстового пошуку Elasticsearch та кешуючу підсистему Redis, інтегровані між собою в єдину функціональну структуру для забезпечення швидкого доступу, надійності, масштабованості та узгодженості даних.

Практична значимість отриманих результатів полягає у розробленні веб-орієнтованої інформаційної системи бібліотеки, що може бути використана навчальними закладами, публічними бібліотеками, корпоративними центрами знань та електронними архівами. Система забезпечує пошук за індексованими даними, автоматизацію процесів позики та бронювання, формування статистичних звітів, збереження бібліографічної інформації, рольову модель доступу та сучасний інтерфейс взаємодії з користувачами. Ключовими перевагами є зручність використання, можливість масштабування, підтримка різних сценаріїв доступу, висока швидкодія та адаптивність до збільшення обсягів даних.

ЗМІСТ

ВСТУП 6

1 ОГЛЯД ВІДОМИХ РІШЕНЬ 9

1.1 Еволюція бібліотечних інформаційних систем 9

1.2 Open-source системи для бібліотек та їх розвиток 10

1.3 Сучасні технології веб-орієнтованих інформаційних систем бібліотек 11

1.4 Smart Libraries та інтелектуальні сервіси 12

1.5 Архітектурні підходи до створення веб-орієнтованих бібліотечних інформаційних систем 13

1.6 Структура модулів веб-орієнтованої бібліотечної інформаційної системи .. 15

1.7 Висновки 16

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ БІБЛІОТЕКИ 18

2.1 Постановка задачі та аналіз вимог 18

2.2 UML-моделювання 20

2.3 Моделювання та структура даних 22

2.4 Архітектура інформаційної системи 25

2.5 Бізнес-процеси системи 28

2.6 Обґрунтування вибору технологій 31

2.7 Висновки 33

3 МЕТОД ТА АЛГОРИТМ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ БІБЛІОТЕКИ 35

3.1 Метод побудови серверної частини веб-орієнтованої інформаційної системи бібліотеки 35

3.2 Метод організації доступу до даних 36

3.3 Алгоритм роботи пошукового модуля 38
3.4 Алгоритм кешування даних 41
3.5 Алгоритм обробки клієнтських запитів 44
3.6 Алгоритм автентифікації та авторизації 46
3.7 Алгоритм взаємодії підсистем 48
3.8 Висновки 50
4 ІНФОРМАЦІЙНА СИСТЕМА БІБЛІОТЕКИ ТА ЇЇ ПРОГРАМНА РЕАЛІЗАЦІЯ 52
4.1 Програмна реалізація серверної частини 52
4.2 Програмна реалізація клієнтської частини 55
4.3 Реалізація функціональних можливостей системи 58
4.4 Тестування інформаційної системи 66
4.5 Розгортання інформаційної системи 68
4.6 Оцінка якості роботи системи 70
4.7 Висновки 73
ВИСНОВКИ 75
ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ 77
ДОДАТОК А 83
ДОДАТОК Б 86

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ІС – інформаційна система

СУБД – система управління базами даних

ПЗ – програмне забезпечення

БП – бізнес-процес

БЛ – бізнес-логіка

ORM – об’єктно-реляційне відображення (Object-Relational Mapping)

EC – Elasticsearch

JWT – JSON Web Token

UI – інтерфейс користувача

REST – Representational State Transfer

ВСТУП

У процесі розробки сучасних інформаційних систем важливим завданням є забезпечення їх якості, надійності та функціональної повноти. Незавершеність або некоректність реалізації окремих програмних компонентів може призвести до численних проблем під час експлуатації системи, включаючи втрату даних, збої в роботі, зниження продуктивності та дисфункцію ключових процесів. Особливо це стосується веб-орієнтованих систем, що функціонують у режимі багатокористувацької взаємодії та мають значну кількість бізнес-процесів, які необхідно узгодити між собою.

У сфері автоматизації бібліотечної діяльності ці проблеми мають специфічне значення, оскільки бібліотечні системи повинні не лише забезпечувати доступ до інформаційних ресурсів, але й гарантувати достовірність, актуальність і збереження даних, а також можливість швидкого пошуку, бронювання та обліку літератури. Відсутність ефективної та надійної інформаційної системи може значною мірою ускладнювати роботу бібліотеки та знижувати рівень доступності інформації для користувачів.

Актуальність теми роботи полягає в необхідності створення сучасної веб-орієнтованої інформаційної системи бібліотеки, яка здатна забезпечити автоматизацію внутрішніх процесів, підтримку великих обсягів даних, зручність для користувачів та високий рівень продуктивності. Сьогодні багато бібліотечних установ потребують оновлення своїх інформаційних систем або впровадження нових, оскільки традиційні програмні рішення часто не забезпечують достатнього рівня інтерактивності, масштабованості та швидкодії. Тому дослідження, спрямоване на розробку методу та програмної реалізації веб-орієнтованої інформаційної системи бібліотеки, є важливим і своєчасним.

Метою кваліфікаційної роботи є розроблення методу побудови та програмної реалізації веб-орієнтованої інформаційної системи бібліотеки, здатної автоматизувати ключові функціональні процеси, забезпечити ефективний пошук інформації, підтримку позик та бронювань, а також надійну взаємодію

користувачів із бібліотечними ресурсами. Досягнення поставленої мети вимагає опрацювання сучасних методів моделювання, архітектурних підходів, алгоритмів управління даними та вебтехнологій.

Поставлена мета досягається розв’язанням таких основних задач:

1) необхідно розробити метод побудови серверної частини веб-орієнтованої інформаційної системи бібліотеки;

2) сформувати модель структури даних та алгоритми взаємодії між підсистемами;

3) розробити алгоритми пошуку, кешування та обробки клієнтських запитів;

4) створити архітектуру клієнтської частини для забезпечення зручної та інтуїтивної взаємодії користувача з системою;

5) розробити програмну реалізацію інформаційної системи та оцінити якість її роботи;

6) дослідити можливості масштабування та інтеграції системи в інфраструктуру бібліотеки.

Об’єктом дослідження є процес проєктування та розроблення веб-орієнтованих інформаційних систем для автоматизації бібліотечних процесів.

Предметом дослідження є методи, моделі, алгоритми та програмні засоби, що застосовуються для створення веб-орієнтованої інформаційної системи бібліотеки.

Наукова новизна отриманих результатів полягає в удосконаленому методі побудови веб-орієнтованої бібліотечної системи, який відрізняється поєднанням алгоритмів повнотекстового пошуку, кешування даних і динамічного ранжування результатів для підвищення точності й швидкодії. Набула подальшого розвитку інформаційна технологія автоматизованого управління бібліотечним фондом, що забезпечує інтеграцію пошукового модуля з індексованим сховищем даних та механізмами оперативного оновлення інформації в реальному часі.

1) Набув подальшого розвитку метод побудови програмної архітектури веб-орієнтованих бібліотечних систем, що включає узгоджені моделі даних, алгоритми пошуку та механізми інтеграції підсистем.

2) Набула подальшого розвитку інформаційна технологія автоматизації бібліотечних процесів, яка враховує сучасні вимоги до масштабованості, продуктивності, доступності та багатокористувацької роботи.

Практична цінність отриманих результатів полягає у створенні веб-орієнтованої інформаційної системи бібліотеки, яка може бути використана у бібліотеках різного масштабу для підвищення ефективності роботи, оптимізації внутрішніх процесів, забезпечення якісного доступу до інформаційних ресурсів та автоматизації обліку позик і бронювань. Запропонована система може бути основою для подальших науково-дослідних та практичних робіт у галузі інформаційних технологій.

У даній роботі викладено вимоги до методології розроблення веб-орієнтованих інформаційних систем, описано процеси аналізу, моделювання, побудови архітектури та тестування програмного продукту.

Для розв’язання поставлених задач використовуються основні положення теорії програмної інженерії, баз даних, вебтехнологій, алгоритмів оптимізації та методів моделювання складних програмних систем.

За темою кваліфікаційної роботи опубліковані тези доповіді (додаток А), та взято участь у IV Міжнародна конференція молодих учених та здобувачів вищої освіти «Філософські виміри техніки» (26 листопада 2025 р, м. Тернопіль):

1) Волошин О.В. Веб-орієнтована інформаційна система бібліотеки. *Матеріали IV Міжнародна конференція молодих учених та здобувачів вищої освіти «Філософські виміри техніки»*, м. Тернопіль, 26 листопада 2025 р, С. 94-96.

1 ОГЛЯД ВІДОМИХ РІШЕНЬ

1.1 Еволюція бібліотечних інформаційних систем

Сучасні веб-орієнтовані бібліотечні інформаційні системи пройшли тривалий шлях розвитку – від локальних настільних застосунків до масштабованих інтегрованих веб-рішень. Якщо у 1990-х роках бібліотечні системи були обмежені можливостями програм на кшталт CDS/ISIS чи Aleph, які працювали в офлайн-режимі та не підтримували колективну взаємодію, то на сучасному етапі більшість таких систем повністю перенесено у веб-середовище. Це забезпечило доступ до бібліотечних ресурсів 24/7, інтеграцію між філіями, віддалений доступ і можливість масштабування відповідно до зростання кількості користувачів.

Поява open-source підходу докорінно змінила парадигму бібліотечних інформаційних систем. Однією з перших масштабних реалізацій стала Koha [1], впроваджена у 1999 році. Вона продемонструвала, що бібліотека може працювати на вільному ПЗ, яке піддається гнучкій модифікації, локалізації та інтеграції з національними каталогами. На основі цієї моделі з’явилися інші відомі системи: Greenstone [2], DSpace, VuFind [3]. Кожна з них зробила внесок у розвиток цифрових бібліотек, підтримку метаданих, стандартизацію та доступ до електронних ресурсів.

Перехід до веб-середовища та розвиток відкритого коду сприяли стабільному зростанню бібліотечних консорціумів, які використовують спільні інформаційні системи. У Польщі, Чехії, Індії та Україні open-source рішення почали застосовувати для побудови національних репозиторіїв та університетських мереж [4]. Це дозволило суттєво знизити витрати та підвищити ефективність управління бібліотечними фондами.

Сучасні тенденції в еволюції бібліотечних систем визначаються інтеграцією з веб-фреймворками, такими як Node.js, що забезпечують асинхронність, високу продуктивність і можливість масштабування у хмарних середовищах. Завдяки такому переходу бібліотечні системи перетворилися з автономних рішень у складні

інформаційно-аналітичні комплекси, які підтримують пошук, рекомендації, статистичні модулі та взаємодію з іншими сервісами.

Таким чином, еволюція бібліотечних інформаційних систем демонструє послідовний перехід від локальних програм до сучасних відкритих веб-платформ, які стали основою цифрової трансформації бібліотечно-інформаційної діяльності.

1.2 Open-source системи для бібліотек та їх розвиток

Open-source системи стали рушійною силою автоматизації бібліотек у всьому світі, оскільки вони забезпечують гнучкість, адаптивність та відсутність ліцензійних обмежень. Koha стала першою у світі інтегрованою бібліотечною системою з відкритим кодом, яка довела ефективність спільного розвитку програмного забезпечення. Далі з’явилися Greenstone, DSpace і VuFind, які розширили спектр можливостей бібліотек, додавши підтримку цифрових колекцій, репозиторіїв та інтегрованого пошуку.

Порівняльні дослідження підкреслюють конкурентоспроможність відкритих систем у порівнянні з комерційними рішеннями. Зокрема, у роботі Kumar & Gupta зазначено, що Koha, Evergreen та NewGenLib забезпечують високий рівень функціональності без необхідності постійних ліцензійних витрат. Це робить open-source системи оптимальним вибором для багатьох освітніх та публічних бібліотек.

Open-source рішення підтримують широкий спектр професійних стандартів: MARC21, Dublin Core [8], RDF, Z39.50, SRU/SRW, SIP2/NCIP, а також протокол OAI-PMH [9], що забезпечує можливість інтеграції з репозиторіями, національними фондами та агрегаторами метаданих. Наприклад, Greenstone може виступати OAI-PMH сервером і клієнтом, а DSpace [10] – агрегувати та зберігати цифрові колекції. Koha підтримує MARCXML, інтегрується з WorldCat API та Google Books.

Важливою перевагою open-source систем є активність спільнот. Саме спільноти забезпечують оновлення функціональності, багатомовність, локалізацію та безпеку. Це надзвичайно важливо для України, оскільки всі популярні open-

source системи мають українські локалізації або підтримують їх через відкриті переклади.

Open-source підхід створює умови для інновацій: бібліотеки можуть розробляти власні модулі, адаптувати інтерфейс, інтегрувати з локальними реєстрами та національними сервісами [12]. У цьому контексті open-source рішення виступають фундаментом для подальшого розвитку веб-орієнтованих бібліотечних систем.

1.3 Сучасні технології веб-орієнтованих інформаційних систем бібліотек

Сучасні бібліотечні системи поступово переходять на технологічні платформи, які забезпечують високу продуктивність і масштабованість. Однією з ключових технологій є Node.js [11], який працює на основі подієво-орієнтованої моделі та забезпечує неблокуючу обробку запитів. Це дозволяє бібліотечним системам ефективно обробляти одночасні запити користувачів, виконувати асинхронні операції та інтегруватися з зовнішніми сервісами.

Фреймворки Express.js [6] і NestJS [7] забезпечують структуровану серверну архітектуру, підтримку REST API, можливість впровадження мікросервісного підходу та поділ логіки на модулі. Це дозволяє будувати каталогізацію, авторизацію, роботу з видачами, аналітику, модулі звітів і рекомендацій як окремі сервіси.

Бібліотечні системи активно інтегрують професійні стандарти обміну метаданими, такі як Dublin Core [8], OAI-PMH [9], MARCXML, що забезпечує сумісність з міжнародними базами даних і науковими репозитаріями. Системи, побудовані поверх сучасних веб-фреймворків, можуть працювати у хмарних середовищах, підтримувати балансування навантаження та горизонтальне масштабування.

У роботі з даними ключову роль відіграють реляційні бази даних, зокрема PostgreSQL, та інструменти ORM, такі як Prisma. Вони дозволяють чітко

структурувати зв’язки між сутностями, забезпечувати транзакційність та узгодженість даних.

На рівні фронтенда сучасні бібліотечні системи застосовують React, Redux Toolkit, Tailwind CSS та компонентні бібліотеки Shadcn UI. Це забезпечує адаптивність, високу продуктивність клієнтської частини та комфорт взаємодії кінцевого користувача з системою.

Застосування сучасних веб-технологій дозволяє будувати системи нового покоління – інформаційні, аналітичні та інтерактивні, які можуть інтегрувати в собі не лише базові функції, а й модулі рекомендацій, push-сповіщення, статистику в реальному часі та взаємодію з іншими цифровими платформами.

1.4 Smart Libraries та інтелектуальні сервіси

Концепція Smart Libraries формується на основі поєднання традиційних функцій бібліотеки з інтелектуальними, сенсорними та інформаційними технологіями. У межах такого підходу бібліотека розглядається як комплексний цифровий сервіс, який забезпечує персоналізований доступ до ресурсів, автоматизацію рутинних процесів та інтеграцію з зовнішніми інформаційними системами.

Smart Libraries активно використовують RFID для пришвидшення видачі та повернення видань, IoT-пристрої для моніторингу завантаженості читальних залів, кліматичних показників та відвідуваності, а також алгоритми штучного інтелекту для автоматичної класифікації документів, побудови рекомендацій та семантичного аналізу бібліотечних даних. Використання таких підходів дозволяє автоматизувати самообслуговування, оптимізувати роботу персоналу та покращувати взаємодію з читачами.

Дослідження вказують, що Smart Libraries стають осередками інтеграції освітніх платформ, наукових репозитаріїв та мобільних застосунків. Використання відкритих протоколів, таких як OAI-PMH [9], RDF/Linked Data та Dublin Core [8],

дозволяє бібліотекам формувати семантичні зв’язки між матеріалами та розширювати функціональність систем за рахунок сторонніх сервісів.

Node.js [11] і сучасні веб-фреймворки забезпечують можливість реалізації таких інтелектуальних сервісів у режимі реального часу. Наприклад, побудова рекомендаційних систем на основі динамічного аналізу користувацьких запитів [4] або формування дашбордів у реальному часі для адміністраторів [5] стає можливою завдяки асинхронній моделі обробки даних.

Окремим напрямом розвитку Smart Libraries є впровадження адаптивних рекомендаційних механізмів, що аналізують історію взаємодії користувача з бібліотекою та формують персоналізовані добірки літератури. Такі сервіси використовують підходи машинного навчання – колаборативну фільтрацію, контентну класифікацію та гібридні моделі – дозволяючи підвищити релевантність пошуку та рівень задоволеності читачів. Крім того, інтелектуальні аналізатори потоків даних дають змогу відстежувати тренди користувацьких запитів і прогнозувати попит на окремі категорії ресурсів, що сприяє ефективному управлінню бібліотечним фондом.

Таким чином, Smart Libraries формують нову модель бібліотечного середовища, яка заснована на сучасних веб-технологіях, інтерактивних сервісах і глибокій інтеграції цифрових інструментів у традиційні бібліотечні процеси.

1.5 Архітектурні підходи до створення веб-орієнтованих бібліотечних інформаційних систем

Архітектура сучасних бібліотечних інформаційних систем визначає їхню надійність, продуктивність та здатність до масштабування. Протягом довгого часу більшість систем працювали за монолітною схемою, коли всі модулі – каталогізація, облік видань, управління користувачами, звітність – розгорталися у межах одного застосунку. Такий підхід був характерним для ранніх бібліотечних продуктів, але мав низку недоліків: складність підтримки, високу залежність між

модулями, обмеженість масштабування та складнощі з оновленням окремих компонентів.

У сучасних веб-орієнтованих системах переважає модульний або мікросервісний підхід. Кожен модуль може існувати як окремий сервіс, що спрощує оновлення, додає гнучкості при розширенні функціоналу і забезпечує більш ефективне масштабування. Наприклад, модуль /catalog може працювати як окремий сервіс пошуку, тоді як модуль /users відповідає виключно за авторизацію та профілі читачів, а сервіс /notifications – за розсилку повідомлень. У таких системах використовується REST або GraphQL API для взаємодії між компонентами.

Серверні технології нового покоління забезпечують можливість створення високопродуктивних сервісів. Node.js [11] підтримує неблокуючу обробку запитів, що дає змогу бібліотечній системі витримувати високі навантаження під час пошуку, отримання статистики чи роботи з електронними документами. У комбінації з Redis, який використовується як кеш або черга повідомлень, система може працювати в режимі реального часу, не створюючи затримок для користувача.

Важливим аспектом сучасних архітектур є контейнеризація. Завдяки використанню Docker бібліотека може розгортати кластер сервісів, де кожен модуль працює ізольовано. Оркестрація за допомогою Kubernetes або Docker Swarm забезпечує автоматичне масштабування, відмовостійкість і балансування навантаження. Це особливо важливо для великих університетських бібліотек та національних репозиторіїв.

Хмарні платформи, такі як AWS, Google Cloud, Azure, дозволяють розгортати бібліотечні системи в моделі SaaS або PaaS. Система отримує динамічне масштабування, автоматичні резервні копії, моніторинг та інструменти безпеки. Завдяки цьому бібліотечні ІС можуть працювати стабільно навіть при пікових навантаженнях, наприклад під час періоду активного користування студентами.

Ще одним важливим аспектом архітектурного проєктування є підтримка еволюційного розвитку системи. Завдяки модульній структурі бібліотечні

інформаційні системи можуть поступово адаптуватися до нових функціональних вимог, не потребуючи повного перероблення ядра. Це дозволяє інтегрувати додаткові сервіси – наприклад, аналітичні модулі, системи рекомендацій, інструменти виявлення плагіату чи інтегратори наукових баз – без порушення стабільності роботи системи. Така еволюційність архітектури забезпечує тривалу життєздатність рішення та дає змогу легко модернізувати його відповідно до потреб освітніх закладів та інформаційного середовища.

Такий архітектурний підхід забезпечує високу доступність, швидку розробку нових модулів, адаптивність під вимоги конкретної бібліотеки та можливість інтеграції із зовнішніми сервісами – цифровими бібліотеками, репозиторіями, навчальними платформами та корпоративними системами авторизації.

1.6 Структура модулів веб-орієнтованої бібліотечної інформаційної системи

Типова веб-орієнтована бібліотечна система складається з набору модулів, кожен з яких виконує окрему функцію та може бути реалізований незалежно від інших. Такий підхід не лише забезпечує високу гнучкість, а й дозволяє бібліотеці адаптувати або розширювати систему відповідно до власних потреб.

Найважливішим модулем є каталогізація. Він забезпечує внесення, редагування та обробку бібліографічних записів, підтримує формати MARC21, MARCXML та Dublin Core. Каталогізаційний модуль відповідає за зв’язки між авторами, виданнями, темами, рубриками, класифікаційними індексами, а також за імпорт і експорт бібліографічних метаданих.

Другим ключовим модулем є модуль абонементу. У цьому модулі ведеться облік читачів, історія видачі та повернення книг, реєстрація штрафів, формування нагадувань та push-сповіщень. інтеграція з RFID-системами дає змогу автоматизувати процес видачі, а також створити станції самообслуговування.

Окреме місце займає модуль пошуку. Він використовує індексування, повнотекстовий пошук, фільтри, сортування та підказки. Завдяки Elasticsearch система може виконувати пошук за назвою, автором, жанром, анотацією та змістом

документу. Це значно підвищує зручність користування та швидкість доступу до інформації.

Модуль управління електронними ресурсами забезпечує роботу з PDF-документами, електронними книгами, аудіоматеріалами, науковими статтями, а також інтеграцію з репозиторіями. Він відповідає за завантаження, зберігання та обмеження доступу до електронних файлів.

Модуль статистики та аналітики забезпечує перегляд даних у режимі реального часу. Він може формувати звіти про кількість виданих книг, рейтинги популярних авторів, активність користувачів, зайнятість фонду та інші показники, що дають можливість приймати управлінські рішення.

До структури типової бібліотечної інформаційної системи також входять модулі інтеграції. Вони забезпечують взаємодію з зовнішніми системами, такими як репозитарії, навчальні платформи, електронні каталоги інших бібліотек, а також сучасні сервіси типу Google Books або Amazon API. Модулі інтеграції використовують протоколи Z39.50, SRU/SRW, OAI-PMH, REST або SOAP.

У сукупності такі модулі формують єдину інформаційну систему, здатну обслуговувати навчальні заклади, публічні бібліотеки та університетські мережі. Модульний підхід забезпечує масштабованість, простоту оновлення, гнучкість та можливість швидкої адаптації під конкретні потреби установи.

1.7 Висновки

У першому розділі проведено огляд еволюції бібліотечних інформаційних систем та сучасних технологій, які формують основу веб-орієнтованих рішень для автоматизації бібліотек. Було визначено ключові етапи розвитку – від локальних автономних систем до відкритих веб-платформ із розширеним функціоналом, інтеграційними можливостями та підтримкою міжнародних стандартів опису метаданих.

Розглянуті open-source системи, такі як Koha, Greenstone, DSpace та VuFind, сформували базові принципи побудови сучасних бібліотечних рішень. Саме вони

продемонстрували можливість забезпечення високого рівня функціональності без прив’язки до комерційних обмежень, що сприяло формуванню широких спільнот розробників і поширенню бібліотечного ПЗ у світі.

Сучасні веб-технології, включаючи Node.js, Express.js, NestJS, PostgreSQL, Redis, Elasticsearch, React та Redux Toolkit, створили підґрунтя для побудови високопродуктивних, масштабованих та інтегрованих бібліотечних систем. Особливу роль відіграє архітектура, яка переходить від монолітних рішень до модульного або мікросервісного підходу, що забезпечує гнучкість, стійкість до навантажень та можливість швидкого розширення функціоналу.

У результаті розгляду концепції Smart Libraries визначено ключову тенденцію розвитку бібліотек: перехід до інтелектуальних, інтерактивних систем, які поєднують цифрові, мобільні та сенсорні технології, а також застосовують алгоритми штучного інтелекту для покращення якості обслуговування користувачів.

Підсумовуючи, веб-орієнтовані бібліотечні інформаційні системи формують нову парадигму взаємодії між читачами, персоналом та інформаційними ресурсами. Це забезпечує підвищення ефективності роботи бібліотек, розширення доступу до знань та створення цифрового середовища, яке відповідає сучасним вимогам університетів, наукових установ та публічних бібліотек.

2 ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ БІБЛІОТЕКИ

2.1 Постановка задачі та аналіз вимог

У процесі цифровізації бібліотечних установ одним із ключових напрямів розвитку стає створення веб-орієнтованих інформаційних систем, які забезпечують зручний доступ до електронних ресурсів, автоматизацію обслуговування користувачів, інтеграцію з зовнішніми бібліотеками та ефективне управління фондами. Сучасні рекомендації з побудови програмних систем свідчать, що розроблення подібних платформ повинно ґрунтуватися на системному аналізі предметної області, визначенні вимог та формуванні моделі майбутнього програмного продукту [13]. Тому першим етапом проєктування створюваної системи бібліотеки є постановка задачі та формальне визначення функціональних і нефункціональних вимог.

Постановка задачі полягає у визначенні мети, цілей, контексту функціонування, обмежень та основних користувацьких сценаріїв. Метою розробки є створення веб-орієнтованої системи бібліотеки, яка дозволяє користувачам здійснювати пошук видань, перегляд детальної інформації про книги, керування власними позиками, перегляд статистики та роботу з персональним профілем у режимі онлайн. Основою постановки задачі виступають методологічні принципи інженерії вимог, які включають виявлення потреб користувачів, аналіз існуючих процесів бібліотеки, визначення функціоналу та побудову високорівневої моделі системи [14].

У межах аналізу вимог ключовим етапом є визначення функціональних вимог. Функціональні вимоги описують конкретні дії, які система повинна підтримувати. Для бібліотечної системи такими вимогами є: можливість пошуку книг за різними критеріями; перегляд детальної інформації про обрані книги; можливість авторизації та автентифікації; управління позиками; робота з персональним профілем; формування статистичних даних. Методології опису функціональних вимог базуються на сценаріях використання, діаграмах прецедентів та формальних описах задач [15]. Відповідно, у процесі моделювання

функціональності використовуються підходи структурованого аналізу та об’єктно-орієнтовані принципи.

Нефункціональні вимоги встановлюють характеристики, яких система має дотримуватися, незалежно від конкретної реалізації. До нефункціональних вимог веб-орієнтованої бібліотечної системи належать продуктивність, масштабованість, відмовостійкість, зручність використання, безпека та сумісність із сучасними вебтехнологіями. У галузі інженерії програмного забезпечення виділяють ці вимоги у спеціальну групу як такі, що визначають якість програмного продукту [16]. Для створення ефективної бібліотечної системи важливо забезпечити швидкий пошук у великому каталозі, захищеність персональних даних, стабільність роботи під навантаженнями та можливість адаптувати інтерфейс до різних пристроїв.

Також під час аналізу вимог враховуються технологічні, організаційні та правові обмеження. До технологічних обмежень належать обрані інструменти розробки, серед яких важливу роль відіграють серверний фреймворк, база даних, ORM-інструментарій та засоби пошуку. Організаційні обмеження пов’язані з внутрішньою структурою бібліотеки, її політиками та специфікою процесів. Окремо виділяються правові вимоги, що регламентують захист персональних даних, зберігання інформації та відкритий доступ до електронних ресурсів. Дослідження сучасних інформаційних систем підкреслюють важливість врахування всіх типів обмежень на початковому етапі проєктування, оскільки вони значно впливають на подальшу архітектуру системи [17].

Аналіз вимог дає можливість сформувати цілісну модель системи, яка стає вихідною точкою для подальшого проєктування. Цей етап дозволяє визначити ключові сутності, потокові процеси, сценарії взаємодії та межі системи. На основі визначених вимог формується специфікація, яка деталізує поведінку системи та стає основою для UML-моделювання, побудови архітектури та підготовки алгоритмічних рішень. Досвід побудови веб-орієнтованих платформ свідчить, що якісний аналіз вимог забезпечує зниження ризиків та підвищує точність реалізації функціоналу [18].

Таким чином, постановка задачі та аналіз вимог є фундаментальним етапом розроблення інформаційної системи бібліотеки. Вони дозволяють визначити очікувану функціональність, якісні характеристики, межі застосування та обмеження, а також сформувати основу для подальших етапів проєктування. Грамотно проведений аналіз вимог значно підвищує якість та надійність системи, забезпечуючи відповідність між потребами користувачів і технічною реалізацією.

2.2 UML-моделювання

UML-моделювання є ключовим інструментом формального опису структури та поведінки інформаційної системи. У процесі створення веб-орієнтованої бібліотечної системи застосування UML дозволяє визначити взаємодію користувачів із системою, внутрішню логіку програмних компонентів та схему обробки даних. Стандарти UML встановлюють формальний набір діаграм, що забезпечують опис програмної системи на різних рівнях абстракції, включаючи структурний, поведінковий і функціональний аспекти [19].

Моделювання починається з побудови діаграми прецедентів, що визначає основні сценарії взаємодії користувачів із системою (рис 2.1). Для веб-орієнтованої бібліотеки ключовими акторами виступають користувач, адміністратор і гість. Діаграма прецедентів дає змогу формалізувати такі сценарії, як пошук книг, перегляд детальної інформації про видання, керування власними позиками, авторизація, робота з профілем та взаємодія з адміністративним функціоналом. У світовій практиці діаграми прецедентів застосовуються як засіб комунікації між розробниками та представниками замовника, дозволяючи формувати узгоджене бачення функціональності [20].

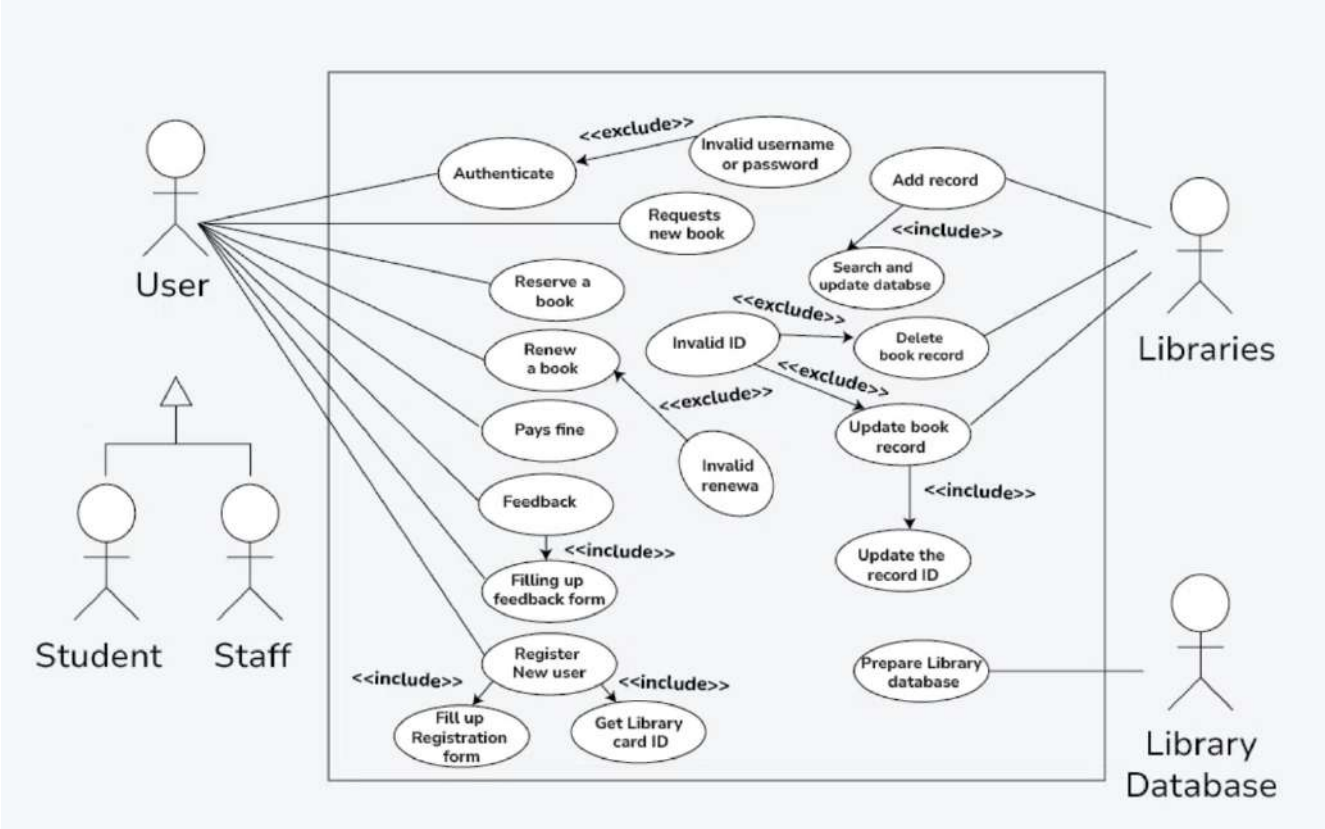


Рисунок 2.1 – UML Use Case Diagram для системи бібліотеки

Наступним важливим кроком є побудова діаграми класів, яка відображає логічну структуру системи. У бібліотечній системі ключовими сутностями є книга, автор, користувач, позика, категорія та статистичний запис. Діаграма класів показує атрибути сутностей, їх взаємозв’язки та напрями асоціацій. Використання об’єктно-орієнтованого моделювання дає можливість визначити багаторівневу структуру класів, виділити спільні характеристики та забезпечити повторне використання компонентів [21]. Діаграма класів також дозволяє на ранньому етапі виявити суперечності в структурі даних і усунути логічні помилки до початку впровадження.

Ще одним важливим інструментом моделювання є діаграми діяльності, що описують алгоритмічні процеси всередині системи. У бібліотечній системі такими процесами можуть бути пошук книги, авторизація, обробка позики чи формування статистики. Діаграми діяльності забезпечують наочне подання логічних переходів між станами системи, що дає можливість описати складну поведінку простими блоками та виявити залежності між діями [22]. Цей тип моделювання особливо

ефективний для складних сценаріїв, які залежать від умовних переходів і взаємодії з різними підсистемами.

Важливою частиною моделювання є також визначення меж системи та взаємодії між її компонентами. Діаграми компонентів та діаграми розгортання дозволяють описати фізичне розташування елементів системи, їх зв’язки та канали взаємодії. Для веб-орієнтованої бібліотечної системи це включає серверну інфраструктуру, базу даних, пошуковий рушій, кеш-сервіс та клієнтську частину, що працює в браузері. Міжнародні стандарти рекомендують використовувати діаграми розгортання для опису архітектурного рівня застосунку, оскільки вони спрощують документування фізичної структури програмного забезпечення [23].

Завершальним етапом UML-моделювання є інтеграція всіх діаграм у єдину модель системи. Така модель дозволяє сформувати цілісне бачення архітектури та функціонування програмної системи, що є основою для подальшої розробки. UML-моделі забезпечують точність опису системи, зменшують ризики невідповідності між проєктною документацією та реалізацією, а також формують фундамент для автоматизації, повторного використання коду та подальшого вдосконалення структури [24]. У практиці проєктування інформаційних систем UML виступає універсальною мовою опису, яку підтримують сучасні CASE-засоби, що значно підвищує ефективність роботи розробників.

Таким чином, UML-моделювання є основою для побудови структурного та поведінкового опису веб-орієнтованої інформаційної системи бібліотеки. Воно дозволяє систематизувати функціональність, визначити взаємозв’язки між компонентами та сформувати архітектурну модель, яка стане основою для подальшої реалізації.

2.3 Моделювання та структура даних

Моделювання даних є одним з ключових етапів проєктування веб-орієнтованої інформаційної системи бібліотеки, оскільки воно визначає логіку зберігання, структурування та доступу до інформації, яка забезпечує

функціональність програмного продукту. У процесі проєктування системи необхідно визначити набір сутностей, їх атрибути та взаємозв'язки, що забезпечують семантичну цілісність та оптимальність архітектури даних. Методології моделювання даних рекомендують починати з концептуального рівня, у межах якого будується високорівнева ER-модель, що відображає основні сутності та їх зв'язки без урахування фізичних обмежень конкретної бази даних [25].

Для бібліотечної системи базовими сутностями є книга, автор, користувач, позика, категорія, видавництво та статистика користувача. ER-модель дозволяє визначити структуру кожної сутності, встановити ключові атрибути та сформувати зв'язки між елементами системи. Наприклад, сутність «Книга» має атрибути назви, опису, року видання, ISBN-ідентифікатора та відношення до автора; сутність «Автор» має атрибути імені, біографії та країн походження, а сутність «Позика» містить інформацію про користувача, книгу, дату видачі та дату повернення. Під час створення ER-моделі особливу увагу приділяють визначенню типів зв'язків: один-до-багатьох, багато-до-багатьох або один-до-одного. Класичні підручники з моделювання даних підкреслюють, що правильний вибір типів зв'язків запобігає появі надлишковості та аномалій оновлення, що суттєво підвищує якість бази даних [26].

Після формування концептуальної моделі здійснюється перехід до логічного моделювання, у межах якого визначаються типи полів, первинні та зовнішні ключі, правила нормалізації та структурні обмеження. Для інформаційної системи бібліотеки особливо важливим є застосування нормалізації до третьої нормальної форми, оскільки це дозволяє уникнути дублювання атрибутів та зменшує ризик суперечностей у даних. Методологія реляційного моделювання наголошує, що нормалізація також сприяє оптимізації запитів та зменшенню обсягу зберігання, хоча у деяких випадках денормалізація може бути доцільною для підвищення продуктивності [27].

Фізична структура даних визначається обраною СУБД – у проекті використовується PostgreSQL, яка забезпечує підтримку реляційної моделі, транзакційність, можливість масштабування та широкий набір інструментів для

роботи зі складними запитами. PostgreSQL підтримує індексацію різних типів, включно з B-Tree, Hash, GIN та GiST-індексами, що дозволяє оптимізувати пошук у великих таблицях. Документація PostgreSQL підкреслює, що правильний вибір типу індексу є критичним елементом оптимізації продуктивності системи [28]. У бібліотечній системі індекси застосовуються для полів ISBN, назв книг, авторів та інших часто використовуваних атрибутів.

Особливу роль відіграє моделювання структур, пов’язаних із пошуковим модулем, який працює у взаємодії з Elasticsearch. Для нього формується окрема структура індексів, що включає токенізацію, аналізатор (рис 2.2), морфологічну обробку та механізми релевантності. Пошукові індекси використовують інший підхід до моделювання даних, базуючись на документно орієнтованому представленні, що дозволяє зберігати структуровані та напівструктуровані об’єкти. Класичне керівництво з побудови пошукових систем наголошує, що моделювання індексу впливає на якість і точність пошуку навіть більше, ніж зміст бази даних [29].

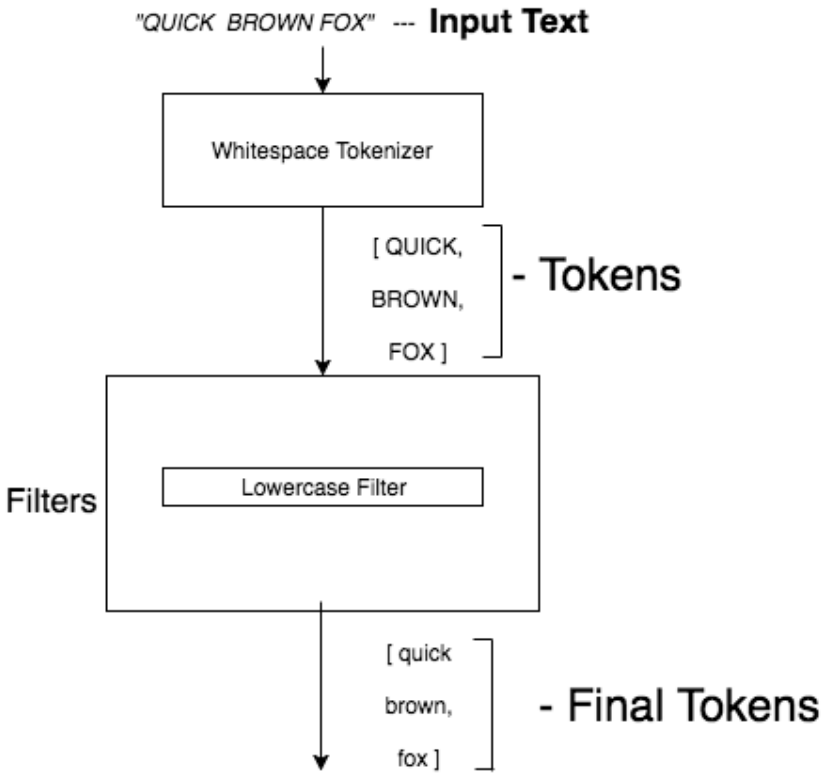


Рисунок 2.2 – Потік даних в аналізаторі Elasticsearch

Іншим важливим елементом структури даних є моделювання таблиць, пов’язаних із користувачами та їхньою взаємодією з бібліотечною системою. Зокрема, таблиця позик містить зв’язок між читачем і книгою, таблиця статистики зберігає інформацію про кількість переглядів, дії користувачів та інші параметри, що дозволяють формувати інтелектуальну аналітику. У сучасній літературі з побудови інформаційних систем наголошується, що виділення таблиць для відстеження поведінкових даних дає можливість підвищити ефективність сервісів рекомендації та персоналізації контенту [30].

Таким чином, моделювання даних є комплексним процесом, який включає концептуальне, логічне та фізичне проєктування структури даних. Правильно побудована модель забезпечує цілісність, узгодженість, оптимальність та масштабованість системи, що є критичними характеристиками для веб-орієнтованої інформаційної системи бібліотеки.

2.4 Архітектура інформаційної системи

Архітектура веб-орієнтованої інформаційної системи бібліотеки визначає структуру її основних компонентів, взаємозв’язки між ними, способи обробки даних та механізми інтеграції. Коректно побудована архітектура забезпечує масштабованість, ефективність, безпеку й узгодженість роботи окремих програмних модулів. Архітектурне проєктування ґрунтується на принципах модульності, розподілення відповідальності, інкапсуляції та багаторівневого поділу системи, що дозволяє уникнути надмірної складності та забезпечити легкість підтримки програмного продукту [31]. Для веб-орієнтованих систем, які мають працювати в умовах значного навантаження та великої кількості одночасних користувачів, архітектурне проєктування є критично важливим елементом створення ПЗ.

Структура системи побудована за багаторівневим підходом, що включає клієнтський рівень, серверний рівень, рівень даних та рівень зовнішніх сервісів. На клієнтському рівні розміщується інтерфейс користувача, реалізований за

допомогою React, який забезпечує динамічний рендеринг компонентів і швидку взаємодію з сервером. Клієнтська частина відповідає за отримання та відображення даних, відправлення запитів до бекенду та реалізацію логіки роботи з профілем користувача, каталогом та інструментами пошуку. У сучасній літературі описано, що розділення клієнтського і серверного рівнів дозволяє створювати високопродуктивні та гнучкі вебзастосунки, які легко масштабуються та адаптуються до вимог користувача [32].

Серверний рівень побудований на основі фреймворку NestJS, який реалізує архітектурні патерни модульності та інверсії залежностей. Такий підхід дозволяє розділити логіку системи на окремі модулі: модуль обробки користувачів, модуль книг, модуль позик, модуль статистики, модуль авторизації, модуль взаємодії з Elasticsearch та модуль роботи з кешем. NestJS використовує контролери для обробки HTTP-запитів, сервіси для реалізації бізнес-логіки й провайдери для взаємодії з базою даних та зовнішніми сервісами. Подібна модульна архітектура підкреслюється в сучасних методологіях побудови серверних застосунків як така, що забезпечує чисту структурованість і полегшує розширення функціоналу [33].

Особливе місце в архітектурі займає рівень даних. У якості основної системи керування базами даних обрано PostgreSQL, яка забезпечує транзакційну цілісність, підтримку складних запитів, індексацію та ефективну роботу з великими масивами інформації. ORM Prisma використовується для взаємодії з базою даних, визначення моделей, управління міграціями та виконання типізованих запитів. Документація Prisma підкреслює, що використання ORM дозволяє зменшити ризик помилок, підвищити читабельність коду та забезпечити чіткий зв’язок між логічною моделлю даних та фізичною структурою БД [34]. Оскільки система включає пошуковий модуль, частина інформації продубльована в Elasticsearch, що дозволяє реалізувати швидкий повнотекстовий пошук і забезпечити релевантність результатів.

Рівень зовнішніх сервісів включає пошуковий рушій Elasticsearch та кеш-систему Redis. Elasticsearch виконує індексацію книг, авторів та описових даних, що дозволяє реалізувати підтримку fuzziness-пошуку, морфологічної обробки

тексту та складних запитів із фільтрацією. Redis застосовується для кешування даних, що значно зменшує навантаження на основну базу даних та прискорює час відповіді сервера. У проєктуванні систем, що працюють із великими обсягами інформації, використання додаткових шарів зберігання та обчислення є фундаментально важливим принципом, що дозволяє забезпечити стабільність та продуктивність під час пікових навантажень [35].

Фізична архітектура системи передбачає розгортання у контейнеризованому середовищі Docker, що забезпечує ізоляцію сервісів, їх повторюваність та можливість масштабування. Кожен компонент – серверна частина, база даних, Redis та Elasticsearch – запускається у власному контейнері, а взаємодія між ними виконується через внутрішню мережу. Підхід контейнеризації є сучасним стандартом у галузі розробки програмного забезпечення, оскільки забезпечує стабільність середовища та спрощує розгортання на будь-яких хост-платформах. Провідні фахівці з архітектури програмних систем підкреслюють, що використання контейнерів дозволяє досягти високого рівня ізоляції та контрольованості середовища виконання [36].

Загалом архітектура веб-орієнтованої інформаційної системи бібліотеки є багаторівневою, модульною та розподіленою. Вона забезпечує чіткий поділ функцій, ефективність роботи, можливість інтеграції з зовнішніми сервісами та високу масштабованість у майбутньому. Архітектурний підхід, покладений в основу системи, відповідає сучасним вимогам до веб-орієнтованих застосунків та забезпечує основу для подальшого розвитку платформи. На рисунку 2.3 зображена діаграма спроєктованої архітектури системи.

Модульна побудова фізичної архітектури дозволяє незалежно оновлювати та масштабувати окремі компоненти без зупинки всієї системи. Використання внутрішньої мережі Docker забезпечує захищений та контрольований обмін даними між сервісами, мінімізуючи ризики несанкціонованого доступу. Такий підхід створює надійну основу для розгортання системи як у локальному середовищі, так і в хмарній інфраструктурі з урахуванням зростання навантаження.

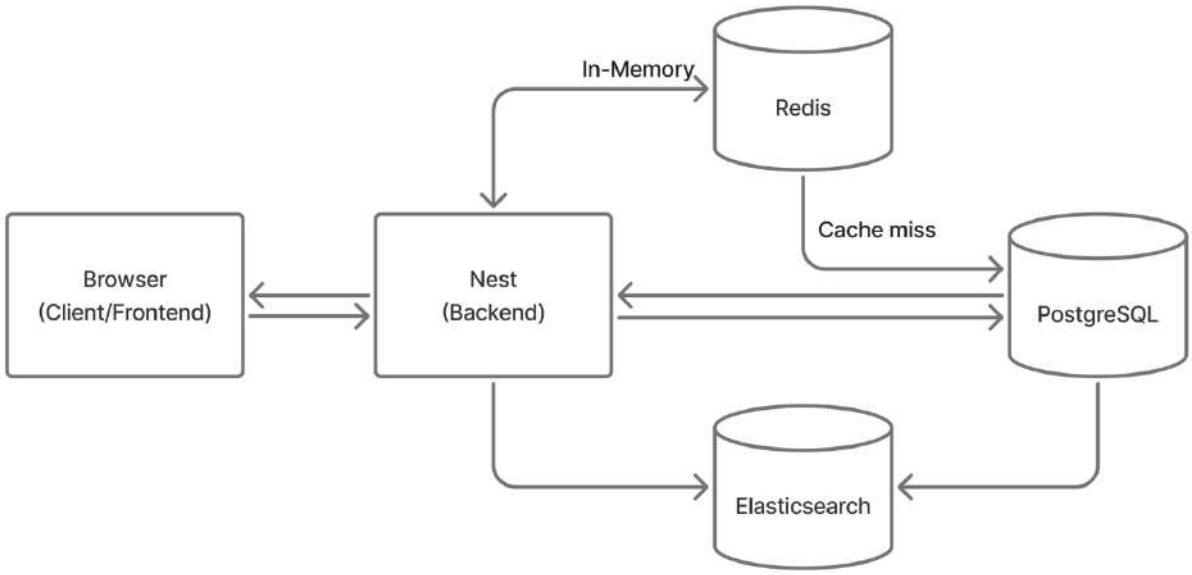


Рисунок 2.3 – Діаграма спроєктованої архітектури системи

2.5 Бізнес-процеси системи

Бізнес-процеси веб-орієнтованої інформаційної системи бібліотеки визначають логіку та послідовність операцій, що забезпечують виконання функціональних можливостей системи. Вони описують рух інформації між компонентами, послідовність дій користувачів та механізми взаємодії серверної та клієнтської частин. Формалізація бізнес-процесів дозволяє визначити раціональну структуру роботи системи, оптимізувати логіку операцій та забезпечити узгодженість між функціональними модулями. У сучасних підходах до проєктування інформаційних систем ключову роль у документуванні бізнес-процесів відіграє мова BPMN, розроблена як стандарт для опису процесів у вигляді блок-схем різного рівня деталізації [37].

Одним із центральних бізнес-процесів системи є процес пошуку книг. Він включає отримання пошукового запиту, попередню обробку даних, звернення до пошукового індексу та формування результатів. Процес починається з того, що користувач вводить ключові слова або фрази у пошукове поле. Після цього система передає запит на сервер, де він перетворюється у формат, придатний для обробки

Elasticsearch. Документація з моделювання бізнес-процесів підкреслює важливість формалізації кожного етапу, оскільки це дозволяє чітко визначити взаємодію між компонентами й оптимізувати логіку обробки запитів [38]. На основі цього процесу створюється BPMN-діаграма, що відображає послідовність дій від моменту введення запиту до отримання результатів.

Іншим ключовим бізнес-процесом є процес авторизації користувача. Він включає введення облікових даних, перевірку відповідності інформації на сервері, генерацію токена доступу та ініціалізацію сесії користувача. Згідно зі стандартами побудови інформаційних систем, авторизація є критичним процесом, оскільки вона забезпечує конфіденційність даних, цілісність профілів користувачів та коректну роботу персональних функцій системи [39]. У BPMN-діаграмі процес авторизації відображається як послідовність дій: отримання логіна й пароля, перевірка наявності користувача в базі даних, звірка пароля, створення JWT-токена та передавання клієнту інформації про успішний вхід.

Також важливим є бізнес-процес управління позиками. Він реалізує логіку видачі та повернення книг, формування історії, контроль термінів та оновлення статистики. У класичних інформаційно-бібліотечних системах ці операції виконуються вручну або частково автоматизовано, але у веб-орієнтованій системі весь процес відбувається інтерактивно. Послідовність дій включає вибір книги, підтвердження видачі, створення запису про позику, контроль терміну та оновлення статусу при поверненні. Теоретичні дослідження бібліотечних процесів підкреслюють, що автоматизація позик значно підвищує ефективність роботи бібліотеки та зменшує ймовірність помилок [40].

Ще одним важливим процесом є формування статистики, яке виконується автоматично на основі інформації про пошукові запити, перегляди книг, кількість позик та активність користувачів. Бізнес-процес включає збір даних, їх агрегацію, аналіз та збереження у відповідній таблиці бази даних. На основі статистичних даних користувач отримує змогу переглядати власну активність, а адміністратор – контролювати загальні тенденції та навантаження. У сучасній аналітичній літературі наголошується, що впровадження автоматизованої статистики є

ключовим компонентом інформаційних систем, оскільки воно підвищує прозорість, керованість та адаптивність системи [41].

Для кожного з бізнес-процесів створюється окрема BPMN-діаграма, що відображає логічний зв’язок між діями та умовами переходу. BPMN дозволяє візуально представити складні процеси, зробити їх зрозумілими для розробників, аналітиків та представників замовника. Міжнародний стандарт BPMN визначає формальні правила побудови діаграм, що дозволяє уникнути неоднозначностей у трактуванні моделей, а також забезпечує можливість автоматичної генерації робочих процесів у рамках систем автоматизації [42]. У результаті бізнес-процеси стають не лише частиною документації, а й важливим інструментом управління логікою програмної системи.

Ще одним допоміжним бізнес-процесом, який відіграє важливу роль у функціонуванні системи, є процес синхронізації даних між різними підсистемами. У веб-орієнтованих бібліотечних системах синхронізація охоплює оновлення інформації в кеші, пошуковому індексі та основній базі даних після виконання будь-якої зміни, пов’язаної з ресурсами бібліотеки. Процес включає виявлення змін, їх фіксацію та ініціювання відповідних оновлень у Redis та Elasticsearch. Такий підхід дає змогу забезпечити узгодженість даних навіть у випадках високої інтенсивності запитів або паралельного виконання операцій різними користувачами. Наявність автоматизованого механізму синхронізації знижує ризик виникнення застарілих даних у пошукових запитах, підвищує точність результатів та гарантує стабільність роботи системи.

Загалом бізнес-процеси веб-орієнтованої бібліотечної системи формують основу функціональної логіки, узгоджуючи дії користувачів із внутрішніми алгоритмами обробки даних. Їх формалізація забезпечує прозорість, керованість та точність функціонування системи й дозволяє створити ефективну модель взаємодії між компонентами, що є необхідною передумовою для подальшої реалізації.

2.6 Обґрунтування вибору технологій

Вибір технологій для розроблення веб-орієнтованої ІС бібліотеки є одним із ключових етапів проєктування, оскільки саме від цього залежить стабільність, масштабованість, продуктивність, безпека та гнучкість системи під час її подальшого використання. Грамотно побудований технологічний стек формує основу для ефективної реалізації функціональних можливостей, полегшує процес розроблення, тестування та розгортання, а також забезпечує можливість подальшого розширення системи без істотних змін архітектури. У сучасній практиці створення веб-орієнтованих систем особлива увага приділяється вибору інструментів, які мають активну спільноту, регулярні оновлення та підтримку промислового масштабу [43].

Основою серверної частини системи обрано фреймворк NestJS, який реалізує принципи модульності, інверсії залежностей та чіткої структурованості коду. NestJS базується на TypeScript, що дозволяє застосовувати статичну типізацію, зменшує кількість помилок під час компіляції та підвищує передбачуваність поведінки системи. Крім того, архітектура NestJS хорошо узгоджується з принципами побудови корпоративних застосунків. В офіційній документації підкреслюється, що використання декораторів, контролерів і сервісів створює зрозумілу структуру проєкту та забезпечує чистий розподіл відповідальностей між компонентами сервера [44]. У контексті бібліотечної ІС це дозволяє реалізувати окремі модулі для книг, користувачів, позик, статистики та пошуку.

Для роботи з БД використовується PostgreSQL, яка належить до класу реляційних СУБД і характеризується високою надійністю, підтримкою транзакцій, потужним механізмом індексації та широким набором функцій для роботи зі складними запитами. PostgreSQL активно використовується в інтегрованих інформаційних системах через свою стабільність, гнучкість і здатність працювати під високим навантаженням. Наукові джерела підкреслюють, що PostgreSQL забезпечує одну з найбільш оптимальних моделей зберігання та опрацювання

структурованих даних, що робить її ефективним рішенням для ІС, де дані зберігаються у глибоко взаємопов’язаній структурі [45].

Додатковим інструментом взаємодії з БД є ОРМ Prisma, яка надає інтерфейс високого рівня для виконання типізованих запитів та спрощує управління міграціями. Prisma забезпечує автоматичну генерацію моделей на основі схеми БД, що покращує узгодженість між логічною моделлю та фізичною структурою даних. У документації вказано, що архітектура Prisma орієнтована на розробників і спрямована на мінімізацію ризиків, пов’язаних із помилками структури запитів, що робить цей інструмент одним з найефективніших для проєктів з акцентом на типобезпечність і масштабованість [46].

Пошукові функції реалізовано на основі пошукового рушія Elasticsearch, який дає можливість працювати з повнотекстовими індексами, аналізаторами, токенізаторами та системами ранжування. На відміну від реляційних СУБД, ЕС працює за принципом індексування документів, що дозволяє значно швидше виконувати пошук за великими корпусами тексту. У спеціалізованих джерелах наголошується, що Elasticsearch є одним з найбільш ефективних сучасних інструментів для реалізації швидкого і релевантного пошуку в системах, які працюють з неструктурованими або напівструктурованими даними [47]. Для бібліотечної ІС це є критичним, оскільки доступ до книг, авторів і описів має бути максимально швидким і точним.

Кешування реалізується за допомогою Redis – високопродуктивного in-memory сховища, що забезпечує надзвичайно швидкий доступ до даних. Redis застосовується як проміжний шар між сервером та БД, дозволяючи зменшити навантаження на PostgreSQL та покращити швидкість реакції системи. У технічній літературі зазначається, що Redis забезпечує затримку на рівні мілісекунд, що робить його ідеальним інструментом для кешування результатів пошукових запитів, авторизаційних токенів та інших даних, які часто використовуються в ІС [48].

Клієнтська частина системи створена на основі React, що дозволяє будувати сучасні UI-компоненти та реалізовувати логіку роботи SPA. React забезпечує

високу реактивність, швидке оновлення інтерфейсу та ефективне управління станом застосунку. Використання React разом з Redux Toolkit дозволяє забезпечити передбачуване управління станом, а також підвищити стабільність роботи інтерфейсу при значній кількості користувацьких взаємодій. Структура SPA відповідає сучасним вимогам до веб-систем, що працюють у реальному часі, та дозволяє досягти високого рівня зручності для користувачів.

Таким чином, вибір технологій для веб-орієнтованої ІС бібліотеки є обґрунтованим з точки зору продуктивності, безпеки, масштабованості та відповідності сучасним стандартам. Кожен елемент технологічного стека виконує власну чітко визначену функцію, а разом вони формують стабільну, надійну та гнучку систему, здатну до подальшого розвитку.

2.7 Висновки

У другому розділі було здійснено повний цикл проєктування веб-орієнтованої інформаційної системи бібліотеки – від постановки задачі та аналізу вимог до формування архітектури, моделі даних, бізнес-процесів та вибору технологічного стека. На основі системного аналізу предметної області визначено цілі створюваної ІС, сформульовано основні функціональні та нефункціональні вимоги, окреслено обмеження й умови експлуатації. Це дозволило задати чіткі рамки для подальших етапів проєктування та забезпечити узгодженість між потребами користувачів і технічною реалізацією.

У процесі UML-моделювання були описані ключові аспекти поведінки та структури системи: виділено основних акторів, сформовано сценарії взаємодії користувачів із системою, визначено базові сутності та зв’язки між ними, описано логіку внутрішніх процесів. Це надало можливість створити цілісну модель функціонування системи на концептуальному рівні й підготувати підґрунтя для реалізації БЛ та модулів.

Моделювання даних дозволило сформувати ДМ, що охоплює сутності «книга», «автор», «користувач», «позика», «категорія», «статистика» та інші

допоміжні структури. На основі ER-підходу й принципів нормалізації побудовано логічну модель БД, оптимізовану для зберігання та обробки великих масивів бібліотечних даних. Вибір PostgreSQL як основної СУБД доповнюється використанням EC для реалізації швидкого повнотекстового пошуку, що забезпечує необхідну продуктивність і гнучкість роботи з текстовою інформацією.

Архітектура системи спроєктована як багаторівнева, модульна й розподілена. Вона включає клієнтську частину на основі React, серверну частину на NestJS із чітко організованими модулями, рівень БД на PostgreSQL з ОРМ Prisma та зовнішні сервіси – EC і Redis. Такий підхід забезпечує масштабованість, можливість розподілу навантаження, простоту розгортання в контейнеризованому середовищі, а також гнучкість подальшого розвитку системи.

Окрему увагу в розділі приділено проєктуванню структури даних та взаємодії між модулями системи. Було визначено основні сутності предметної області бібліотеки, їх атрибути та зв’язки, що відображено у логічній моделі бази даних. Такий підхід дозволив узгодити вимоги бізнес-логіки з архітектурними рішеннями та забезпечити цілісність і узгодженість даних на всіх рівнях системи.

У межах розділу також було формалізовано ключові БП – пошук книг, авторизацію користувача, управління позиками, формування статистики – та представлено їх у вигляді BPMN-діаграм. Це дозволило чітко зафіксувати логіку роботи системи на процесному рівні, забезпечити прозорість взаємодій між компонентами й користувачами, а також створити основу для подальшої реалізації алгоритмів. Обґрунтування вибору технологій показало, що обраний стек (NestJS, PostgreSQL, Prisma, Redis, Elasticsearch, React) відповідає сучасним вимогам до продуктивних веб-орієнтованих ІС, забезпечує достатній рівень надійності, безпеки та масштабованості.

3 МЕТОД ТА АЛГОРИТМ ФУНКЦІОНУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ БІБЛІОТЕКИ

3.1 Метод побудови серверної частини веб-орієнтованої інформаційної системи бібліотеки

Метод побудови серверної частини веб-орієнтованої інформаційної системи бібліотеки ґрунтується на принципах модульної та сервісно-орієнтованої архітектури, що забезпечує узгодженість функціональних компонентів, стійкість до навантажень та можливість масштабування системи. Центральною ідеєю методу є поділ серверної частини на незалежні модулі, кожен з яких виконує чітко визначену функцію, а взаємодія між ними здійснюється за стандартизованими інтерфейсами [49].

Метою методу є створення серверного середовища, здатного забезпечити ефективну обробку клієнтських запитів, узгоджену взаємодію з підсистемами збереження даних, індексації та кешування, а також підтримку високої доступності сервісу. Метод дозволяє зменшити складність системи за рахунок розподілу відповідальностей між компонентами, що підвищує зрозумілість структури та полегшує подальший супровід [50].

Метод побудови серверної частини охоплює послідовність дій, спрямованих на структурну організацію системи:

1) аналіз функціональних вимог та визначення набору сервісів, які забезпечують управління користувачами, книгами, позиками, бронюваннями та пошуком;

2) декомпозиція системи на модулі, де кожен модуль відповідає за єдину функціональність, наприклад модуль авторизації, модуль пошуку, модуль взаємодії з базою даних [51];

3) проєктування сервісного шару, який реалізує бізнес-логіку, перевірку даних та формування результатів для клієнтської частини;

4) формування транспортного шару, що обробляє HTTP-запити, маршрутизує їх та координує роботу сервісів;

5) ізоляція зовнішніх залежностей шляхом запровадження адаптерів для PostgreSQL, Redis та Elasticsearch, що забезпечує гнучкість і можливість заміни технологій без зміни логіки роботи модулів;

6) імплементація механізмів безпеки, включно з аутентифікацією, авторизацією та обробкою помилок;

7) покрокове тестування, яке включає модульні, інтеграційні та навантажувальні тести для перевірки коректності реалізації методу [52].

Застосування методу забезпечує низку переваг:

1) підвищення стійкості системи за рахунок ізоляції компонентів;
2) можливість незалежної модифікації модулів;
3) спрощення налагодження та тестування;
4) ефективну підтримку високих навантажень шляхом розподілу функціональних ролей;
5) підвищення швидкодії за рахунок оптимізації взаємодії з підсистемами кешування та індексації.

Метод побудови серверної частини може бути використаний у складних інформаційних системах, що вимагають чіткої структуризації, надійності та можливості подальшого масштабування. Завдяки універсальності підходу метод може бути застосований у бібліотечних системах, освітніх платформах, системах електронної комерції та будь-яких сервісах з високою інтенсивністю обробки запитів.

3.2 Метод організації доступу до даних

Метод організації доступу до даних у веб-орієнтованій інформаційній системі бібліотеки базується на принципах багатошарової структуризації та чіткого розмежування відповідальностей між компонентами, що здійснюють читання, зміну, кешування та індексацію бібліотечних даних. Основною метою методу є забезпечення узгодженості операцій, передбачуваності результатів та стійкості системи до паралельних навантажень, що є критично важливим для середовищ, де

кілька користувачів одночасно працюють з одними і тими ж сутностями (книги, позики, бронювання) [53].

Метод ґрунтується на інтеграції трьох ключових підсистем: реляційної бази даних PostgreSQL, індексаційного механізму Elasticsearch та кешуючого шару Redis. Кожна підсистема виконує свою роль: PostgreSQL відповідає за транзакційну цілісність, Redis – за оперативний доступ до даних, а Elasticsearch – за повнотекстовий пошук та ранжування результатів. Така комбінація дозволяє мінімізувати затримки, забезпечити масштабованість та створити єдиний універсальний підхід до роботи з даними [54].

У межах методу організації доступу до даних виокремлюється така послідовність дій:

1) формалізація структури даних, під час якої визначаються сутності (книги, користувачі, позики, бронювання), їх атрибути та зв’язки;

2) побудова шару абстракції доступу, який ізолює бізнес-логіку від конкретних механізмів роботи з базою даних; на цьому етапі застосовуються об’єктно-реляційні моделі, що дозволяють зменшити складність запитів та інкапсулюють взаємодію з PostgreSQL [55];

3) створення адаптерів для оперативного читання, що використовують Redis для кешування часто вживаних запитів, зокрема даних про книги та статистику активності читача;

4) формування повнотекстового індексу, який зберігає інформацію про назви, авторів, жанри та описи книг у структурі inverted index, що значно прискорює пошукові операції [56];

5) застосування транзакційного методу, який забезпечує узгодженість операцій під час створення, редагування або видалення даних;

6) синхронізація між підсистемами, у тому числі оновлення індексу та очищення кешу після будь-якої операції, що змінює стан даних.

Застосування цього методу забезпечує такі переваги:

1) зниження навантаження на базу даних за рахунок використання кешу;

2) можливість виконання миттєвого пошуку за допомогою Elasticsearch;

3) поліпшення цілісності та передбачуваності даних через транзакції;

4) підвищення масштабованості, оскільки кожен компонент може розгортатися окремо;

5) розширюваність структури даних без потреби змінювати логіку обробки запитів.

Метод організації доступу до даних може використовуватися в будь-яких системах, що працюють з великими обсягами інформації та потребують одночасного забезпечення швидкого пошуку, транзакційної надійності та оперативної взаємодії з даними, зокрема в освітніх, архівних, аналітичних та CRM-системах.

3.3 Алгоритм роботи пошукового модуля

Алгоритм роботи пошукового модуля у веб-орієнтованій інформаційній системі бібліотеки визначає формалізовану послідовність дій, спрямованих на обробку пошукового запиту, оптимізацію витрат на доступ до даних та формування релевантної відповіді користувачеві. Метою алгоритму є забезпечення швидкого та точного пошуку книг за ключовими словами, авторськими даними або описами, використовуючи індексаційні структури Elasticsearch і механізми кешування для зменшення затримок і навантаження на основну базу даних [57]. Завдяки використанню розподіленого індексування система може ефективно опрацьовувати великі обсяги текстової інформації та забезпечувати високу точність ранжування результатів.

Алгоритм базується на комбінуванні кешованих даних та повнотекстового пошуку, що дозволяє мінімізувати час відповіді та забезпечити плавну роботу навіть за умови підвищеного навантаження. Він включає такі послідовні кроки (рис. 3.1):

1) отримання та нормалізація запиту – пошуковий запит приводиться до стандартизованого формату: видаляються зайві пробіли, текст переводиться у

нижній регістр, виконуються операції токенізації та лематизації (це необхідно для підвищення точності подальшого пошуку у структурі inverted index);

2) перевірка кешу Redis – система визначає, чи існує готова відповідь на цей запит у кеші, у разі виявлення збігу користувач одразу отримує результат, що істотно скорочує затримку виконання;

3) формування пошукового запиту до Elasticsearch – якщо кеш не містить відповідного результату, формується запит до індексаційної структури Elasticsearch, яка зберігає дані у вигляді зворотного індексу, використовуються методи fuzzy matching, phrase matching або multi-field search залежно від типу даних [58];

4) виконання пошукового запиту та ранжування результатів – Elasticsearch повертає список документів, відсортованих відповідно до функції релевантності, алгоритм додатково застосовує правила фільтрації за статусом книги (доступна, заброньована, на руках) для отримання результатів у найбільш корисному для користувача форматі;

5) запис результату в кеш – отриманий результат зберігається у Redis із заданим часом життя, що дозволяє повторно використати його у випадку однакових пошукових запитів;

6) формування відповіді та повернення результату користувачеві – система упаковує структуровані дані про книги (назва, автор, інвентарний статус, рейтинг тощо) та передає їх клієнтській частині для відображення у каталозі.

Алгоритм забезпечує низку важливих властивостей:

1) мінімізацію часу відповіді завдяки застосуванню кешу;

2) підвищення точності пошуку через використання індексаційних структур;

3) стійкість до високих навантажень;

4) можливість масштабування, оскільки Elasticsearch та Redis функціонують як незалежні сервіси;

5) узгодженість даних завдяки автоматичному оновленню індексу після змін у сховищі.

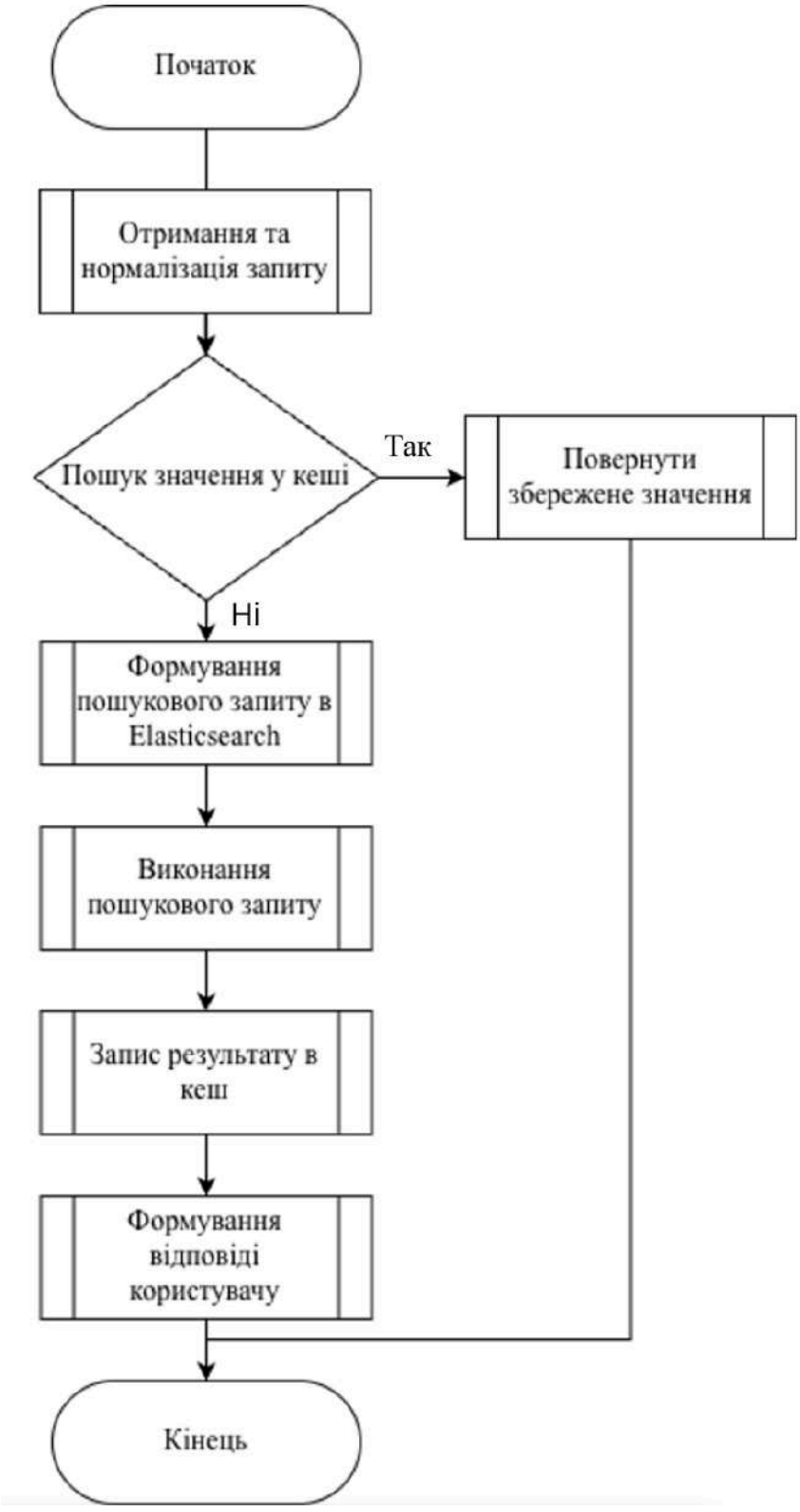


Рисунок 3.1 – Алгоритм роботи пошукового модуля

Запропонований алгоритм може бути застосований у будь-яких інформаційних системах, що потребують повнотекстового пошуку, включаючи електронні каталоги, архівні системи, освітні портали та CRM-рішення.

3.4 Алгоритм кешування даних

Алгоритм кешування даних у веборієнтованій інформаційній системі є одним з ключових механізмів підвищення продуктивності та оптимізації взаємодії між серверною частиною й підсистемами збереження інформації. Його застосування дозволяє значною мірою зменшити кількість звернень до реляційної бази даних, що, у свою чергу, підвищує швидкодію сервера та здатність системи витримувати інтенсивні навантаження. В умовах великої кількості користувачів та постійних пошукових операцій кешування є необхідним елементом архітектури, що дозволяє забезпечити стабільність роботи та мінімізувати затримки [61].

Застосування методу кешування базується на властивостях in-memory сховищ, які здатні виконувати операції читання та запису в десятки разів швидше, ніж навіть оптимізована реляційна база. У запропонованій системі Redis виступає як проміжний шар між сервером і основними підсистемами даних, забезпечуючи швидкий доступ до часто запитуваної інформації. Завдяки своїй архітектурі Redis дає можливість масштабувати систему горизонтально та зберігати великі масиви структурованих даних із мінімальною затримкою [62]. Завдяки підтримці різних структур даних – таких як хеші, списки, множини та відсортовані набори – Redis дозволяє гнучко організовувати кеш відповідно до потреб окремих модулів системи. Крім того, механізми TTL і політики витіснення забезпечують автоматичне оновлення застарілих даних, підтримуючи актуальність інформації без додаткового навантаження на сервер.

Додатковою перевагою кешування є можливість зменшення навантаження не лише на базу даних, а й на мережеву інфраструктуру системи, оскільки значна частина відповідей формується без повторного виконання складних обчислень або запитів. Це особливо важливо в сценаріях, де дані змінюються відносно рідко, але запитуються дуже часто, як-от у модулях каталогізації, статистики або персональних рекомендацій. Такий підхід дозволяє оптимізувати використання обчислювальних ресурсів, підвищити пропускну здатність API та забезпечити прогнозовану затримку при обробці однотипних запитів. У результаті кешування

стає не лише технічним інструментом, а й стратегічним методом підвищення ефективності всієї інформаційної системи. Крім того, застосування TTL-механізмів дає змогу контролювати актуальність даних у кеші та уникати повернення застарілих значень. Це забезпечує баланс між швидкодією та достовірністю інформації, що є критично важливим для динамічних вебсистем.

Алгоритм отримує на вхід ключ кешу та параметри запиту, формує на виході збережене значення або обчислений результат і містить таку розширену послідовність етапів (рис 3.2):

1) формування ключа кешу – на основі маршруту запиту та параметрів формується унікальний ключ у визначеному форматі, важливою частиною алгоритму є побудова ієрархії ключів, що дозволяє ефективно групувати результати пошуку, статистику чи інформацію про користувачів;

2) запит до кешу та визначення сценарію обробки – якщо Redis повертає значення, система ідентифікує його як валідне, якщо значення знайдено, виконується швидке повернення результату без виконання запитів до бази даних;

3) логіка обробки кеш-промаху (cache miss) – у випадку відсутності запису система перемикається на стандартний механізм обробки даних – виконується запит до PostgreSQL або Elasticsearch, важливо, що на цьому етапі проводяться додаткові перевірки, наприклад форматування результатів або підготовка до індексації [63];

4) запис нового значення до Redis – після отримання результату він записується у кеш з параметром TTL, що регулює тривалість існування запису, цей механізм запобігає накопиченню застарілих даних і забезпечує динамічне оновлення вмісту кешу в процесі роботи системи;

5) механізм синхронізації стану системи – алгоритм передбачає періодичне оновлення або видалення кешованих записів у разі змін у базі даних, усі операції, що змінюють сутності (створення книги, редагування, позика, повернення), автоматично викликають процедуру інвалідації ключів [64].

6) повернення результату клієнту – незалежно від того, отримані дані з кешу чи бази даних, клієнт отримує уніфікований структурований результат.

Рисунок 3.2 – Алгоритм кешування даних

У результаті робота алгоритму забезпечує такі властивості:

1) оптимізація часу відповіді – особливо під час повторюваних пошукових запитів;

2) розвантаження бази даних, що дає можливість обробляти більшу кількість паралельних операцій;

3) масштабованість, оскільки Redis можна легко розподіляти між вузлами;

4) стійкість до пікових навантажень, що часто зустрічається у бібліотечних системах під час подій або зміни навчальних семестрів;

5) підвищення надійності, завдяки контролю узгодженості кешу та основного сховища даних.

3.5 Алгоритм обробки клієнтських запитів

Алгоритм обробки клієнтських запитів визначає послідовність дій, яка гарантує коректну та узгоджену взаємодію між користувацьким інтерфейсом і серверною частиною інформаційної системи. Метою алгоритму є забезпечення передбачуваності виконання операцій, стабільності системи та її стійкості до помилок і високих навантажень. Алгоритм реалізує логіку приймання запиту, його перевірки, маршрутизації, виконання бізнес-операцій та формування відповіді [65].

Алгоритм отримує на вхід HTTP-запит, формує на виході структуровану JSON-відповідь і включає такі ключові етапи (рис 3.3):

1) отримання запиту сервером – транспортний шар приймає запит і виконує первинний аналіз маршруту, методу та заголовків;

2) перевірка автентичності та авторизації – система аналізує токен користувача, визначає контекст операції та перевіряє його доступ до відповідної функції, у разі недійсного або відсутнього токена запит завершується з поверненням помилки безпеки [66];

3) маршрутизація запиту – логіка обробки передається відповідному контролеру, який відповідає за конкретну частину функціоналу: керування книгами, позиками, бронюваннями чи переглядом профілю;

4) валідація параметрів – на цьому етапі перевіряється відповідність переданих даних визначеним схемам, некоректні дані призводять до завершення запиту та повернення повідомлення про помилку [67];

5) виконання бізнес-логіки – сервісний шар виконує основні операції: пошук інформації, звернення до БД, індексу або кешу, створення нових записів тощо, уся

логіка є ізольованою від механізмів збереження даних, що підвищує гнучкість системи [68];

6) формування та повернення відповіді – результат обробки структурується у формат JSON і повертається клієнту, де відображається у вигляді відповідних елементів інтерфейсу.

Рисунок 3.3 – Алгоритм обробки клієнтських запитів

Алгоритм забезпечує:

1) високу передбачуваність виконання;
2) безпеку через перевірку автентифікації;
3) гнучкість під час розширення системи;
4) стійкість до навантажень завдяки оптимізованому доступу до даних;
5) стабільність роботи за рахунок централізованої обробки помилок.

3.6 Алгоритм автентифікації та авторизації

Алгоритм автентифікації та авторизації забезпечує контроль доступу до ресурсів веборієнтованої інформаційної системи бібліотеки. Його головна мета – перевірити особу користувача та визначити перелік дій, які він має право виконувати відповідно до своєї ролі. У системі реалізовано рольову модель, де основними ролями є читач та адміністратор, кожна з яких має визначений набір дозволених операцій. Використання JWT-токенів дає можливість поєднати безпеку, масштабованість та ефективність перевірки прав доступу [69].

Алгоритм отримує на вхід ключ кешу та параметри запиту, формує на виході збережене значення або обчислений результат і включає таку розширену послідовність етапів (рис 3.4):

1) надсилання облікових даних – користувач вводить email/логін та пароль, після чого дані передаються до сервера у вигляді захищеного запиту;

2) перевірка облікових даних – сервер виконує пошук користувача в БД та проводить перевірку пароля шляхом порівняння його хешу з еталонним значенням, при невідповідності сервер повертає повідомлення про помилку;

3) формування JWT-токена – у разі успішної перевірки система генерує токен, що містить ідентифікатор користувача, його роль та час дії, токен підписується секретним ключем, що унеможливлює його підробку [70];

4) надсилання токена клієнту – токен зберігається у браузері клієнта (localStorage або sessionStorage), після чого використовується для доступу до захищених маршрутів;

5) перевірка токена під час кожного запиту – сервер валідовує токен, визначає роль користувача та зіставляє її з дозволами для запитаної операції;

6) надання або заборона доступу – у разі відповідності прав доступ дозволяється, і запит передається далі до бізнес-логіки; у протилежному випадку повертається помилка “403 Forbidden”.

Рисунок 3.4 – Алгоритм автентифікації та авторизації

Алгоритм гарантує:

1) безпеку передачі даних;
2) захист від несанкціонованого доступу;
3) масштабованість завдяки stateless-архітектурі;
4) можливість централізованого керування ролями та дозволами.

3.7 Алгоритм взаємодії підсистем

Алгоритм взаємодії підсистем визначає логіку комунікації між основними компонентами системи: клієнтською частиною, серверною частиною, сховищем даних, кешем та модулем повнотекстового пошуку. Він забезпечує узгодженість виконання операцій та стабільність функціонування системи в умовах різного навантаження. Такий алгоритм є ключовим для підтримки реактивності користувацького інтерфейсу та коректної синхронізації даних у всіх підсистемах [71].

Алгоритм забезпечує послідовну трансформацію користувацьких запитів або системних подій у вихідні дані, що надсилаються до відповідної підсистеми та включає такі послідовні етапи (рис 3.5):

1) генерація події у клієнтському інтерфейсі – користувач ініціює дію, наприклад перегляд каталогу, створення позики, пошук книги або перегляд профілю;

2) передача запиту на сервер – фронтенд формує HTTP-запит, додає токен автентифікації та передає його на backend;

3) обробка запиту сервером – сервер визначає тип операції, виконує перевірку доступу та керує маршрутизацією запиту до відповідного сервісного модуля [72];

4) взаємодія з Redis та базою даних – якщо запит належить до операцій читання, система спочатку звертається до Redis, у разі відсутності даних у кеші виконується операція з PostgreSQL або Elasticsearch, результати зберігаються у кеші для подальших запитів;

5) повернення результатів до клієнта – сформована відповідь передається назад до інтерфейсу, де дані візуалізуються у таблицях, картках або графічних елементах;

6) синхронізація станів підсистем – у разі змін у даних виконується інвалідація кешу, оновлення індексів Elasticsearch, а також реактивне оновлення інтерфейсу на клієнті.

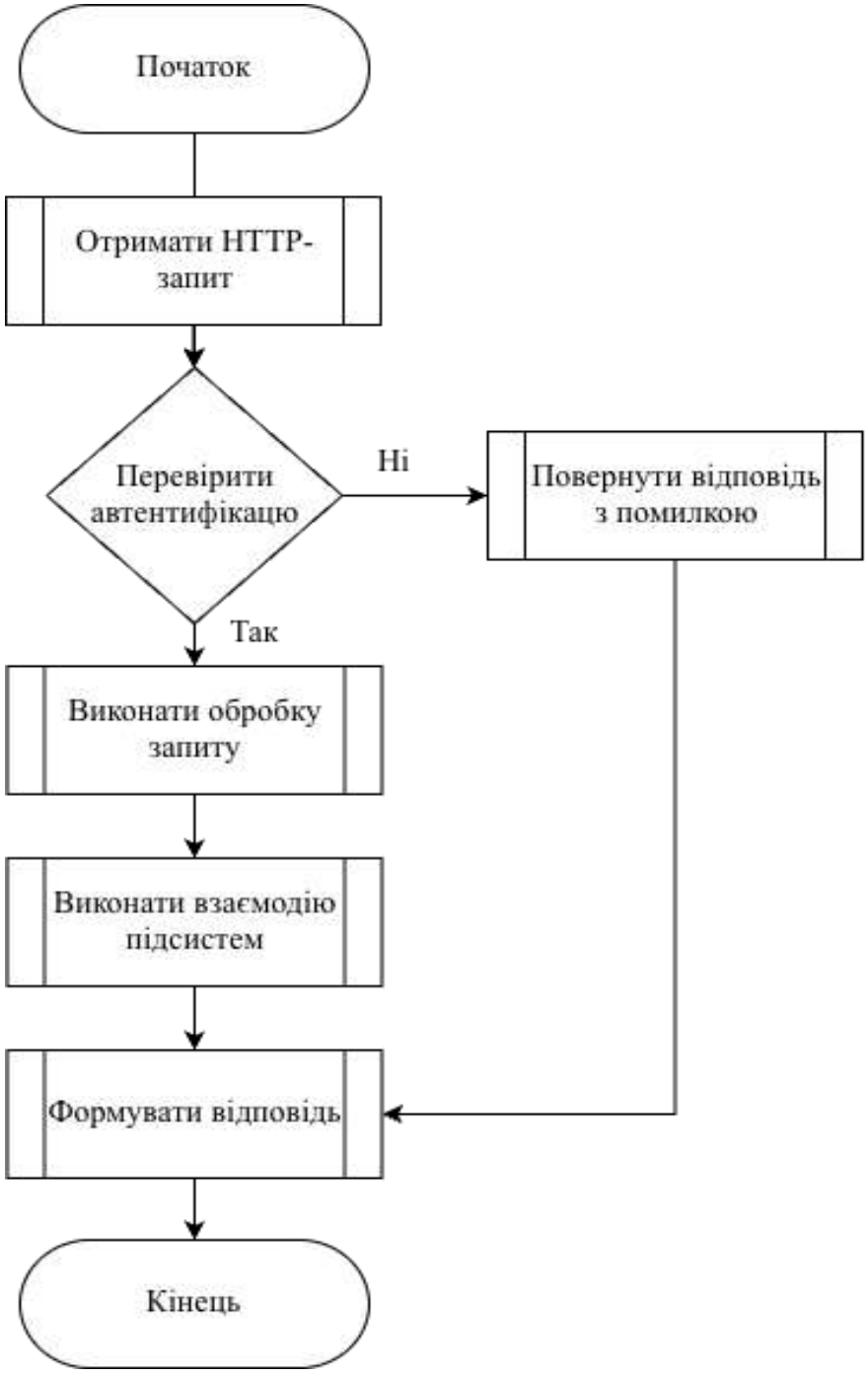


Рисунок 3.5 – Алгоритм взаємодії підсистем

Алгоритм забезпечує:

1) узгодженість між підсистемами;
2) мінімізацію затримок у роботі;
3) ефективний розподіл навантаження;
4) високу доступність завдяки кешу та незалежним сервісам.

3.8 Висновки

У третьому розділі було сформовано комплексний підхід до побудови, організації та функціонування веб-орієнтованої інформаційної системи бібліотеки, який охоплює методи та алгоритми роботи ключових елементів серверної частини. На основі розроблених структур, моделей і сценаріїв взаємодії було визначено універсальний набір механізмів, що забезпечують ефективність та стабільність системи під час обробки різних типів користувацьких операцій.

Розглянутий метод побудови серверної частини дає змогу створити гнучку архітектуру, яка підтримує незалежне масштабування модулів, ізоляцію відповідальностей та централізовану координацію обробки запитів. Окрему увагу приділено організації доступу до даних, де ключовими аспектами визначено транзакційність, суворий контроль цілісності та розмежування доступу. Реалізація цього методу дозволяє гарантувати узгодженість інформації за високих навантажень і інтенсивної паралельної взаємодії користувачів.

Алгоритм пошукового модуля забезпечує точний та швидкий пошук через механізми індексації, нормалізації тексту та ранжування результатів, що сприяє підвищенню зручності користування системою. Алгоритм кешування значно зменшує кількість звернень до бази даних і оптимізує обробку найчастіших операцій, що відіграє важливу роль у стабільності системи. Алгоритм обробки клієнтських запитів формує чітку логіку життєвого циклу кожної операції, включаючи автентифікацію, валідацію, бізнес-логіку та формування відповіді у стандартизованому форматі.

Значну роль у системі відіграє алгоритм автентифікації та авторизації, що забезпечує захист персональних даних, контроль доступу до критичних ресурсів і коректну взаємодію користувачів із функціональними можливостями системи відповідно до їх ролей. Алгоритм взаємодії підсистем гарантує цілісність станів між модулями, забезпечує узгоджену роботу каталогу, позик, бронювань, статистики та пошуку.

Додатково слід зазначити, що запропоновані методи та алгоритми формують не лише технологічну, але й методологічну основу для побудови сучасних бібліотечних ІС. Вони забезпечують формалізацію взаємодії між компонентами, підвищують передбачуваність роботи системи та створюють умови для інтеграції нових інтелектуальних сервісів, таких як автоматичне формування рекомендацій, розширений аналіз поведінкових даних користувачів чи прогнозування бібліотечної активності. Таким чином, розроблений комплекс рішень може бути використаний як базовий шаблон для побудови інших веб-орієнтованих систем подібного класу, що потребують високої продуктивності, узгодженості даних та адаптивності до змін середовища.

Сукупність розглянутих методів і алгоритмів створює комплексну логічну структуру, яка забезпечує надійність, масштабованість і відмовостійкість інформаційної системи бібліотеки. Структурованість та системність цих підходів формують основу для її подальшого розвитку і дозволяють адаптувати систему до майбутніх змін у функціональних вимогах або обсягах даних.

4 ІНФОРМАЦІЙНА СИСТЕМА БІБЛІОТЕКИ ТА ЇЇ ПРОГРАМНА РЕАЛІЗАЦІЯ

4.1 Програмна реалізація серверної частини

Програмна реалізація серверної частини веб-орієнтованої інформаційної системи бібліотеки базується на фреймворку NestJS, який реалізує модульну структуру, підтримує інверсію залежностей та надає можливість побудови масштабованих REST API. Серверна частина працює як центральний вузол, що забезпечує обробку клієнтських запитів, взаємодію з базою даних PostgreSQL, пошуковим індексом Elasticsearch та кешуючим шаром Redis [73].

Система побудована за принципом розділення відповідальностей, де кожен модуль відповідає за окрему функціональність, а бізнес-логіка зосереджена у сервісах. Це дозволяє легко розширювати функціонал і забезпечує високу гнучкість під час модернізації або масштабування системи. У структурі проєкту виділено модулі: BooksModule, LoansModule, ReservationsModule, UsersModule, AuthModule, StatisticsModule та SearchModule, які взаємодіють між собою через інжекцію залежностей.

Комунікація із СУБД реалізована через ORM Prisma, яка генерує типобезпечні запити та забезпечує контроль структури даних на етапі компіляції. Схеми моделей описують структуру таблиць бази даних, включаючи сутності користувачів, книг, позик і бронювань. Наприклад, модель книги у Prisma має таку структуру:

```
model Book {
  id            String     @id @default(uuid())
  isbn          String
  title         String
  author        String
  publisher     String?
  publishYear   Int?       @map("publish_year")
```

```
    description     String?    @db.Text
    coverImage      String?    @map("cover_image")
    genres          String[]
    totalCopies     Int        @default(0) @map("total_copies")
    availableCopies Int        @default(0) @map("available_copies")
    status          BookStatus @default(AVAILABLE)
    borrowCount     Int        @default(0) @map("borrow_count")
    averageRating   Decimal    @default(0)    @map("average_rating")
@db.Decimal(3, 2)
    createdAt       DateTime   @default(now()) @map("created_at")
    updatedAt       DateTime   @updatedAt @map("updated_at")

    loans       Loan[]
    reservations Reservation[]
    reviews     Review[]

    @@index([title, author])
    @@index([genres])
    @@map("books")
  }
```

Ця модель відображається в таблицю books, а всі операції (додавання, редагування, вилучення) виконуються через сервісний шар, який викликає методи Prisma Client. Такий підхід мінімізує можливість помилок і забезпечує чіткість бізнес-логіки.

Важливу роль у серверній частині відіграє обробка HTTP-запитів. Усі запити проходять через контролери, які визначають кінцеві точки API та маршрутизацію. Наприклад, контролер книг може містити метод отримання списку книг:

```
@Get()
async findAll(
```

```
  @Query('page') page: number = 1,
  @Query('limit') limit: number = 10,
) {
  return this.booksService.findAll(page, limit);
}
```

Цей фрагмент коду демонструє типовий обробник запиту GET, який викликає відповідний метод сервісу й повертає сформовану відповідь клієнту.

Особливістю реалізації є робота з Redis для прискорення обробки запитів. Дані, що часто повторюються (наприклад, список найпопулярніших книг, актуальна статистика або результати пошуку), кешуються у Redis із визначеним часом життя. Такий підхід зменшує навантаження на базу даних і скорочує час відгуку сервера. Кешування реалізоване через спеціальний сервіс:

```
const cacheKey = `books:page:${page}:limit:${limit}`;
// Перевіряємо кеш
const cached = await this.cacheManager.get(cacheKey);
if (cached) {
  return JSON.parse(cached as string);
}
// основна логіка…
const result = { books, total };
// Кешуємо результат на 5 хвилин
await this.cacheManager.set(
  cacheKey,
  JSON.stringify(result),
  5 * 60 * 1000,
);
return result;
```

Реалізація пошукового функціоналу здійснюється за допомогою Elasticsearch, який підтримує індексацію книг за назвою, автором та описом. Після кожної зміни у записах система викликає оновлення індексу, що забезпечує актуальність пошукових результатів.

Система також має розвинений механізм безпеки, що реалізований через модуль аутентифікації. Користувачі проходять верифікацію через JWT, що дозволяє контролювати доступ до ресурсів і керувати сеансами. Приклади використання guard:

```
@Get('check')
@UseGuards(JwtAuthGuard)
async checkAuth(@CurrentUser() user) {
  return {
    authenticated: true,
    user,
  };
}
```

Це забезпечує доступ до захищених маршрутів лише для автентифікованих користувачів.

Таким чином, серверна частина реалізована як модульна та масштабована система, що забезпечує ефективну взаємодію між підсистемами, високу продуктивність, безпеку та стабільну роботу в умовах навантаження. Ретельно продумана архітектура дозволяє забезпечити коректну роботу всіх компонентів і мінімізує витрати на подальшу підтримку.

4.2 Програмна реалізація клієнтської частини

Клієнтська частина веб-орієнтованої інформаційної системи бібліотеки реалізована з використанням стеку React [74] + Redux Toolkit + Axios + TailwindCSS

+ ShadCN/UI, що забезпечує гнучку компонентну побудову інтерфейсу, ефективний стан-менеджмент та швидку реакцію на користувацькі дії. Основною концепцією під час розробки клієнтської частини є чітке розділення презентаційної, логічної та мережевої складових, що дозволяє підтримувати чисту архітектуру та забезпечує легку масштабованість проєкту.

Структура інтерфейсу побудована за принципом модульності, де кожна сторінка та компонент є окремою функціональною одиницею. Наприклад, сторінки Books, BookPage, Loans, Reservations, Statistics, Profile реалізовані як автономні модулі з власною логікою та інтерфейсом. Це дозволило створити систему, у якій кожен модуль може розвиватися незалежно, не порушуючи роботу інших частин застосунку.

Для керування глобальним станом використовується Redux Toolkit [75], який спрощує створення сховища, редюсерів і асинхронних операцій. Загальна схема роботи стан-менеджменту включає три етапи: ініціацію дії, виконання асинхронного запиту через Axios, оновлення стану після отримання відповіді від сервера. Наприклад, типовий Redux-слайс може містити таку логіку:

```
export const fetchBooks = createAsyncThunk(
  "books/fetchBooks",
  async ({ page = 1, limit = 20 }: { page?: number; limit?: number }) => {
    const response = await booksAPI.getAll(page, limit);
    return response.data;
  }
);
```

Цей фрагмент демонструє механізм отримання списку книг та автоматичне оновлення стану після завершення запиту. Такий підхід робить клієнтську частину передбачуваною і забезпечує стабільність обробки даних у реальному часі.

Інтерфейс користувача розроблено з використанням TailwindCSS, що дозволило швидко формувати адаптивні та чисті компоненти з мінімальною

кількістю власних стилів [76]. TailwindCSS сприяє стандартизації інтерфейсу, зменшує дублювання CSS-коду та робить структуру стилів прозорою. Окремі елементи інтерфейсу реалізовані через бібліотеку ShadCN/UI, яка забезпечує якісні та строго типізовані компоненти, такі як модальні вікна, кнопки, списки, селектори та інтерактивні елементи.

Особливу увагу приділено реалізації маршрутизації. Застосунок використовує React Router, що дозволяє організувати структуру сторінок за принципом SPA (Single Page Application). Це забезпечує плавні переходи між сторінками без перезавантаження сторінки, що суттєво підвищує зручність роботи для користувачів бібліотеки. Маршрути визначаються наступним чином:

```
<BrowserRouter>
  <Routes>
    {/* Public routes */}
    <Route path="/login" element={<LoginForm />} />
    <Route path="/register" element={<RegisterForm />} />
    {/* Private routes */}
    <Route
       path="/"
       element={<PrivateRoute><DashboardLayout /></PrivateRoute>}
    >
    <Route index element={<Navigate to="/dashboard" replace />} />
    <Route path="dashboard" element={<Dashboard />} />
    <Route path="books" element={<Books />} />
    <Route path="books/:id" element={<BookDetails />} />
    <Route path="loans" element={<MyLoans />} />
  </Routes>
</BrowserRouter>
```

Такі маршрути забезпечують прямий доступ до функціональних можливостей системи та дозволяють користувачу швидко перемикатися між ключовими розділами.

Комунікація між клієнтом і сервером здійснюється через Axios, який автоматично додає JWT-токен до кожного запиту. Це дозволяє захищати приватні сторінки, зокрема сторінку позик, бронювань, статистики й адміністрування. На клієнтській частині реалізовані «route guards»: якщо користувач не автентифікований, його перенаправляють на сторінку входу.

Важливим аспектом реалізації клієнтської частини є адаптивність. Інтерфейс автоматично підлаштовується під різні розміри екранів – мобільні телефони, планшети, ноутбуки. Це гарантує доступність бібліотечної системи для всіх категорій користувачів і робить застосунок зручним у повсякденному використанні.

Таким чином, клієнтська частина системи побудована на сучасних веб-технологіях, що забезпечують високу продуктивність, інтерактивність і масштабованість. Архітектурні рішення дозволяють легко інтегрувати новий функціонал, модифікувати існуючі модулі та підтримувати стабільність роботи системи в умовах активної взаємодії користувачів.

4.3 Реалізація функціональних можливостей системи

Реалізація функціональних можливостей веб-орієнтованої інформаційної системи бібліотеки охоплює комплекс взаємопов’язаних модулів, кожен з яких підтримує окремий сценарій взаємодії користувача з системою. Під час розробки інтерфейсу й логіки роботи сторінок було враховано принципи юзабіліті, доступності, швидкодії та відповідності ролям користувачів [77] – читача та адміністратора.

Додатково при проєктуванні функцій було використано підхід компонентної архітектури, що забезпечує модульність та ізольованість інтерфейсних блоків. Це дозволяє спростити подальший розвиток системи та адаптацію функціоналу до

змін бізнес-вимог. Окрема увага приділялася уніфікації поведінкових шаблонів: усі модулі реалізують єдині правила навігації, принципи валідації даних, структуру відповідей та формат реакції на помилки. Такий підхід підвищує передбачуваність інтерфейсу та спрощує взаємодію користувачів із системою.

Головна сторінка є стартовим пунктом взаємодії користувача з системою, вона має два варіанти відображення, залежно від ролі користувача – читача і бібліотекара або адміністратора.

Для читача відображаються персональні метрики – кількість активних позик, кількість бронювань, наявні штрафи, а також дата найближчого повернення книги. Також там можна побачити список популярних книг згідно метрики позик, а також список активних позик активного користувача (рис. 4.1). Таке відображення дає змогу швидко оцінити власний статус у системі та спланувати взаємодію з бібліотекою.

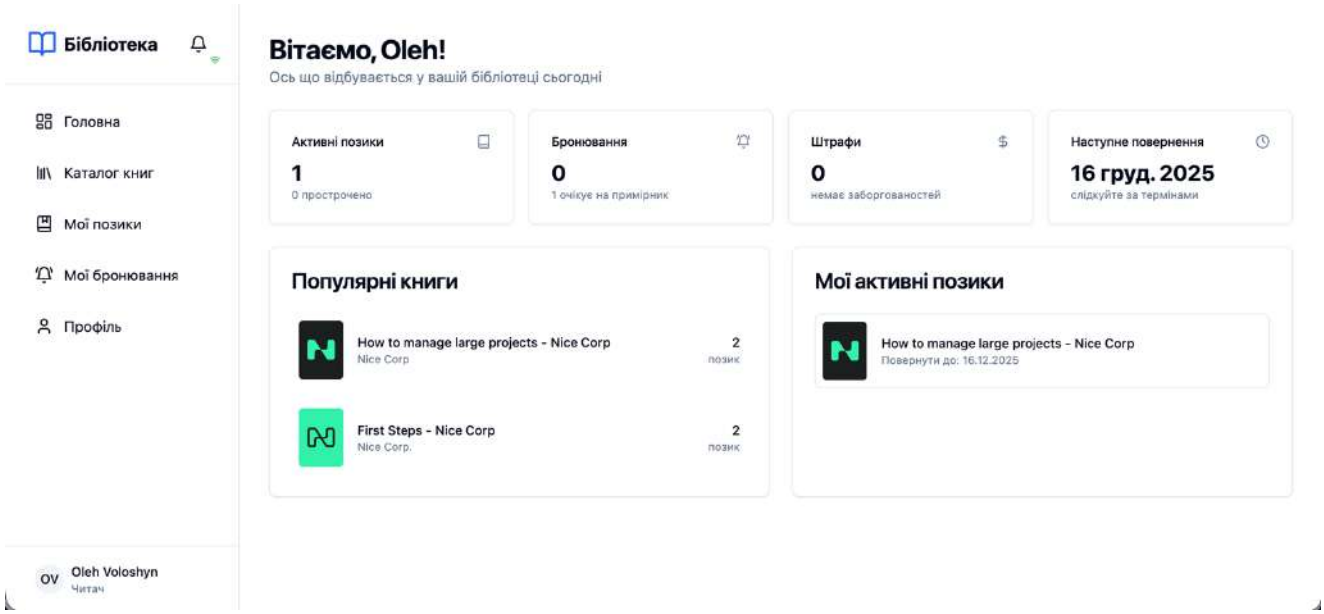


Рисунок 4.1 – Головна сторінка користувача

Для адміністратора виводяться глобальні показники – загальна кількість активних позик, кількість бронювань у системі, середня тривалість позики та інші метрики, необхідні для контролю роботи бібліотеки (рис. 4.2).

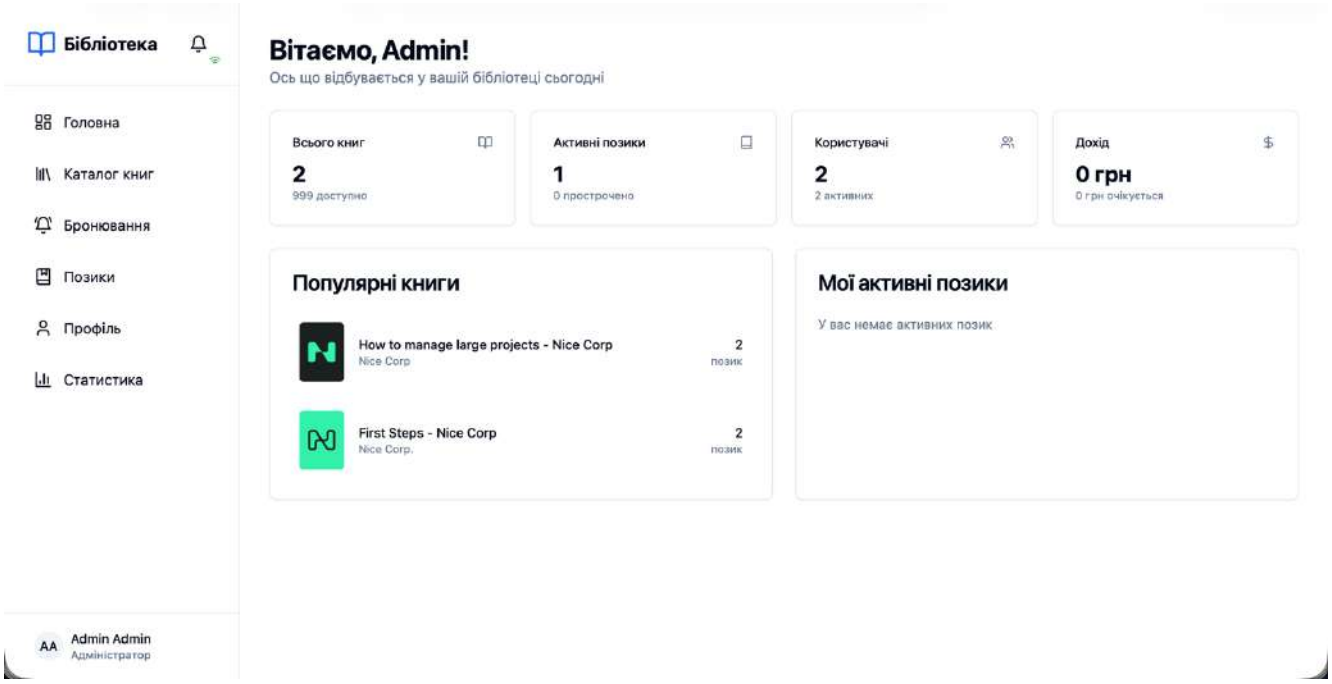


Рисунок 4.2 – Головна сторінка адміністратора

Каталог книг є одним із ключових модулів системи, що забезпечує пошук, перегляд та оцінку наявності книг.

Користувач бачить перелік карток книг із зазначенням назви, автора, рейтинг та статусу доступності (рис. 4.3). Система динамічно відображає доступ чи недоступність книги, спираючись на фактичний стан у базі даних.

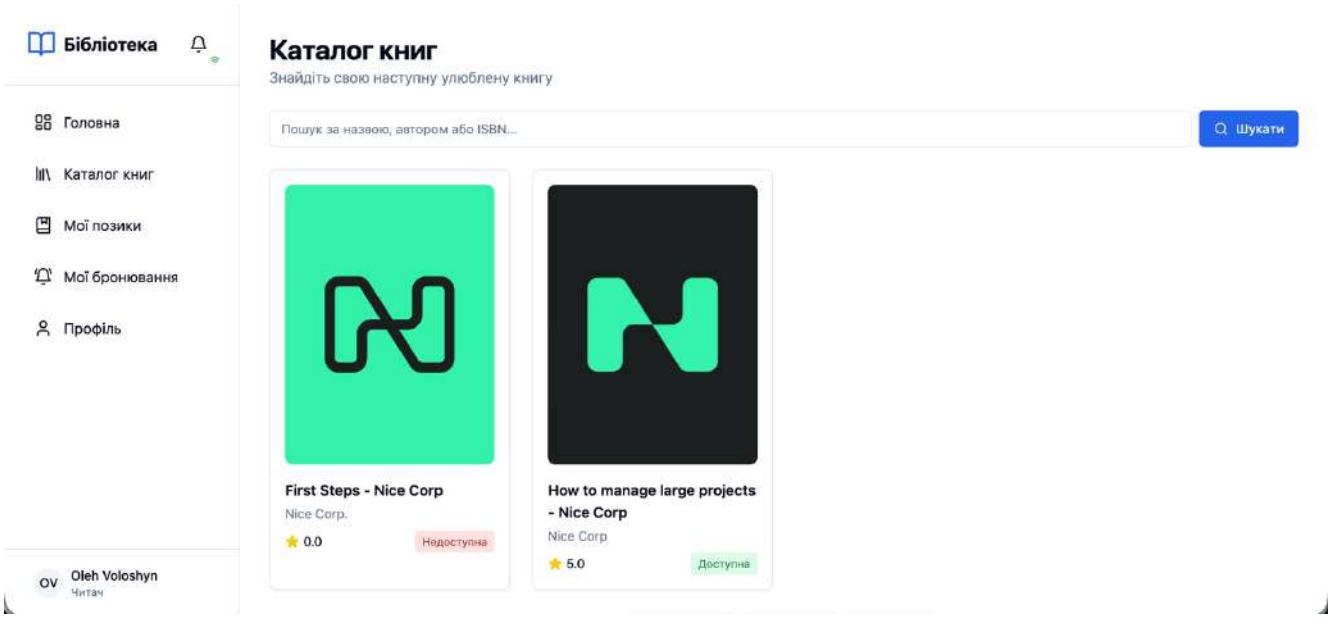


Рисунок 4.3 – Каталог книг (режим читача)

Сторінка книги містить детальну інформацію про вибрану позицію: опис, автора, рік видання, ISBN, актуальний статус та кількість доступних екземплярів.

Читач може виконувати дві основні дії – оформити позику або зробити бронювання (рис 4.4). Усі дії передаються на сервер у режимі реального часу, а зміни статусів оновлюються через глобальний стан застосунку.

У адміністратора на сторінці книги відображаються додаткові функції: редагування метаданих, видалення книги або зміна статусу її доступності (рис 4.5).

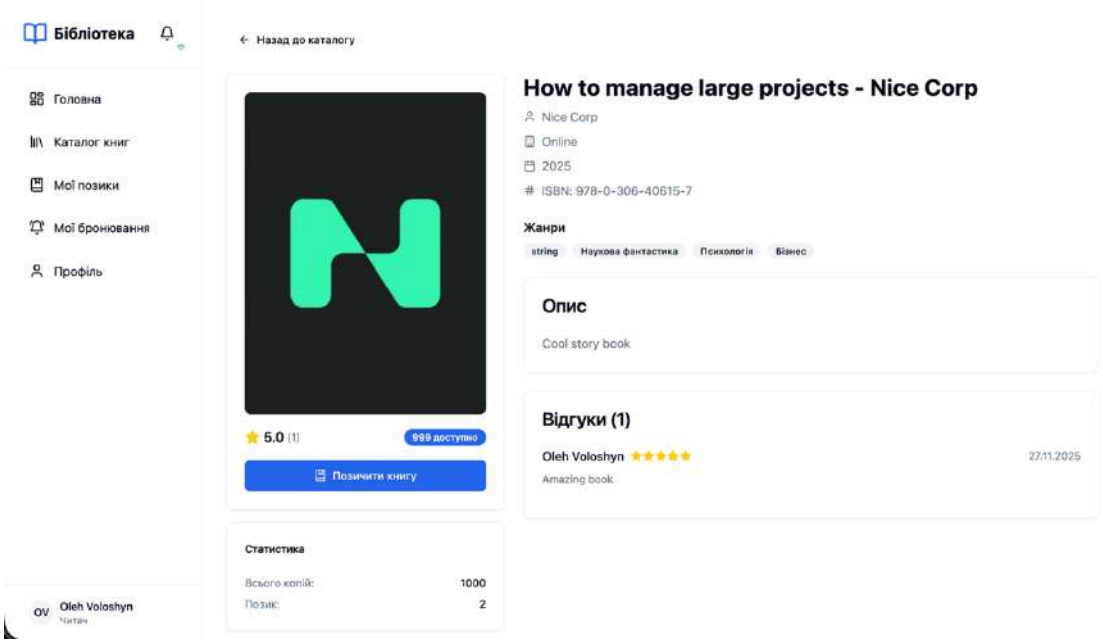


Рисунок 4.4 – Сторінка книги (режим користувача)

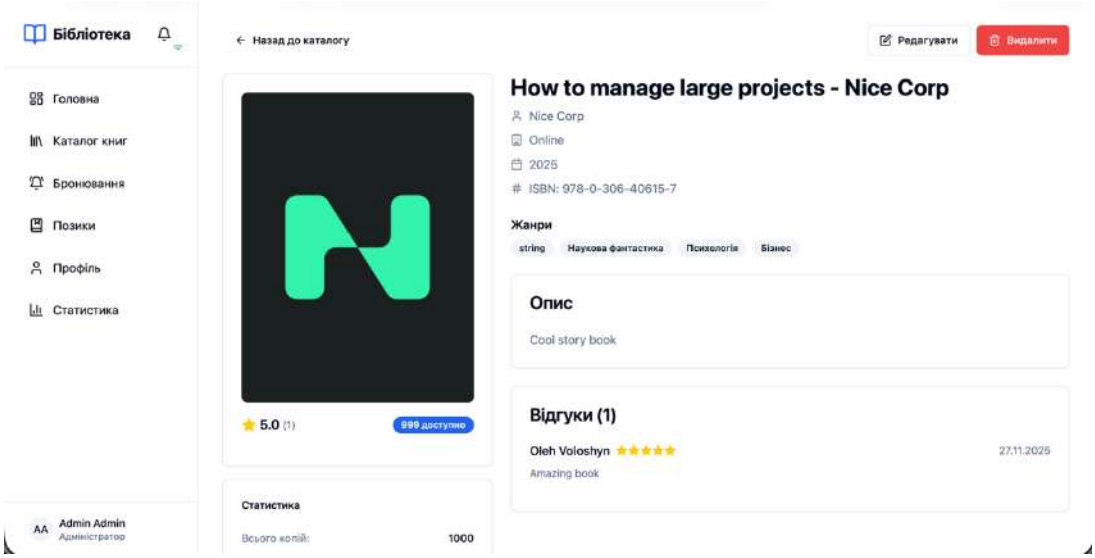


Рисунок 4.5 – Сторінка книги (режим адміністратора)

Сторінка мої позики призначена для читачів і дає змогу переглядати всі активні позики (рис 4.6).

Кожна позиція містить інформацію про дату отримання, дату планованого повернення та наявність можливості продовження. Користувач може натиснути кнопку «Повернути» або «Продовжити» – обидві дії надсилають запит на сервер і викликають оновлення статусу книги.

Оновлення позик відбувається без перезавантаження сторінки, що забезпечує комфорт і швидкодію системи [78].

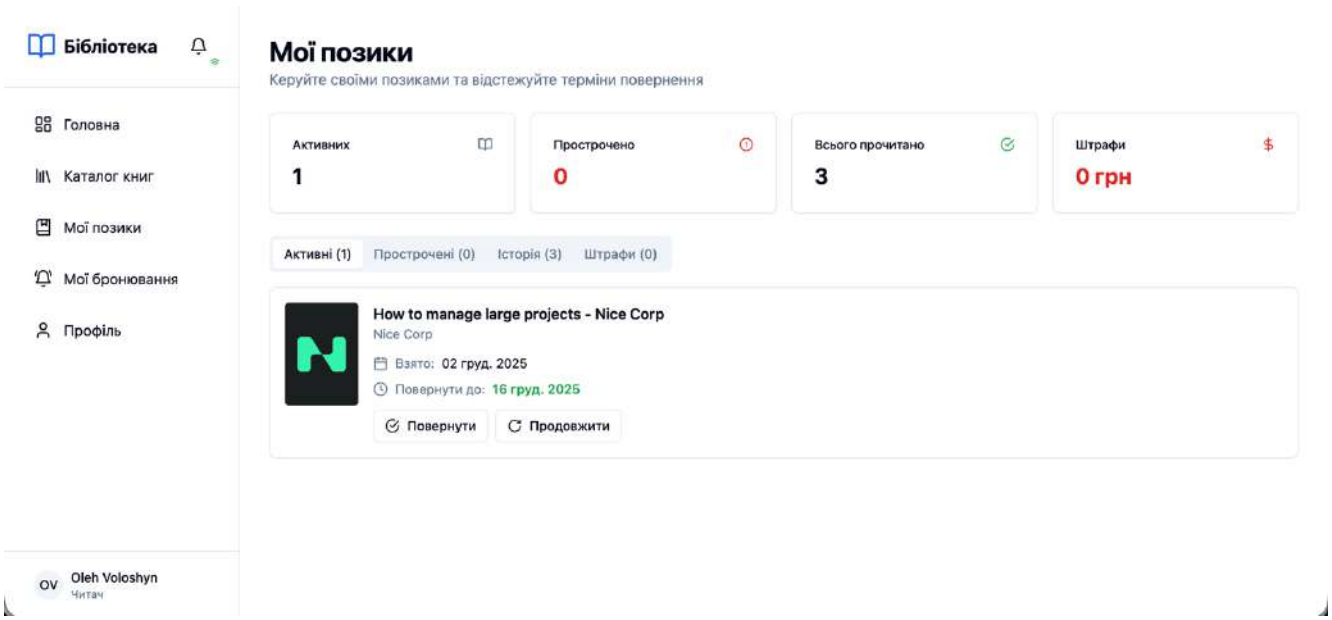


Рисунок 4.6 – Сторінка мої позики

Сторінка мої бронювання дозволяє користувачу швидко орієнтуватися в станах своїх бронювань та планувати їх використання. На цій сторінці є три категорії бронювань: Готові, Очікують і Історія. Перехід між ними здійснюється миттєво, оскільки дані зберігаються у локальному стані, а оновлення надходять тільки за потреби. Також у користувача є можливість відмінити бронювання (рис 4.7).

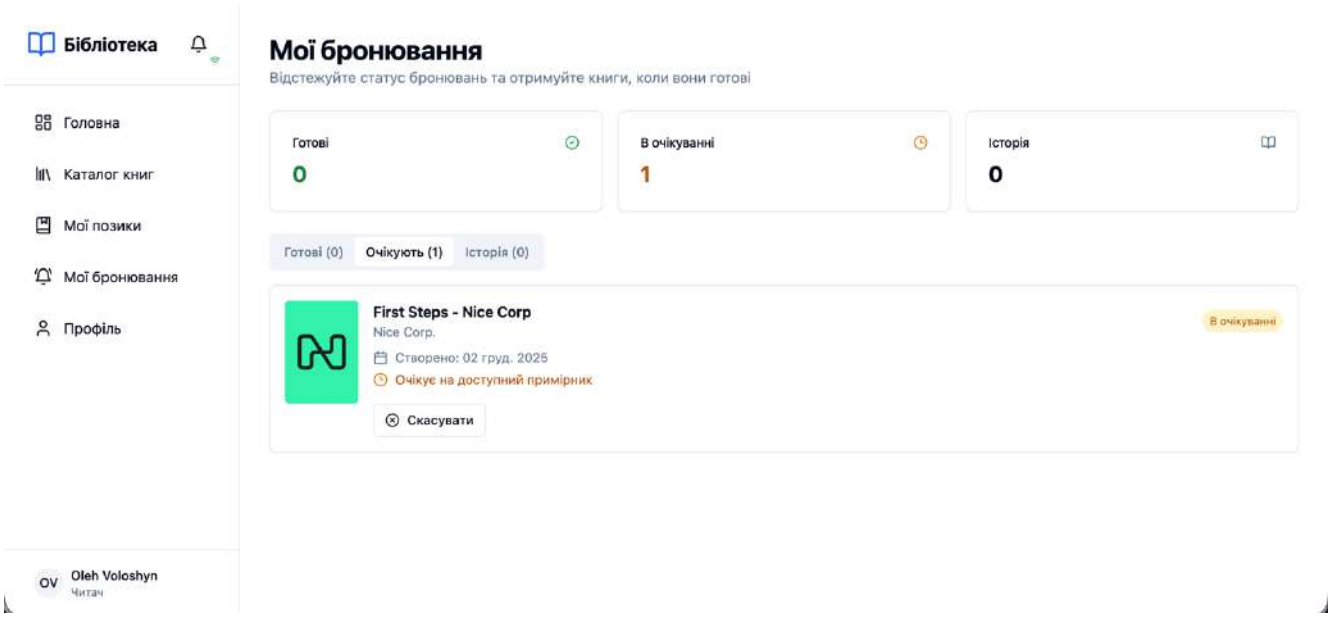


Рисунок 4.7 – Мої бронювання

Адміністратор має доступ до глобальних списків позик (рис 4.8) і бронювань (рис 4.9) усіх користувачів. Ці модулі дозволяють контролювати кількість виданих книг, актуальні строки позик, порушення термінів та стан бронювання. Інтерфейс має зручну табличну структуру та кнопки виконання адміністративних операцій.

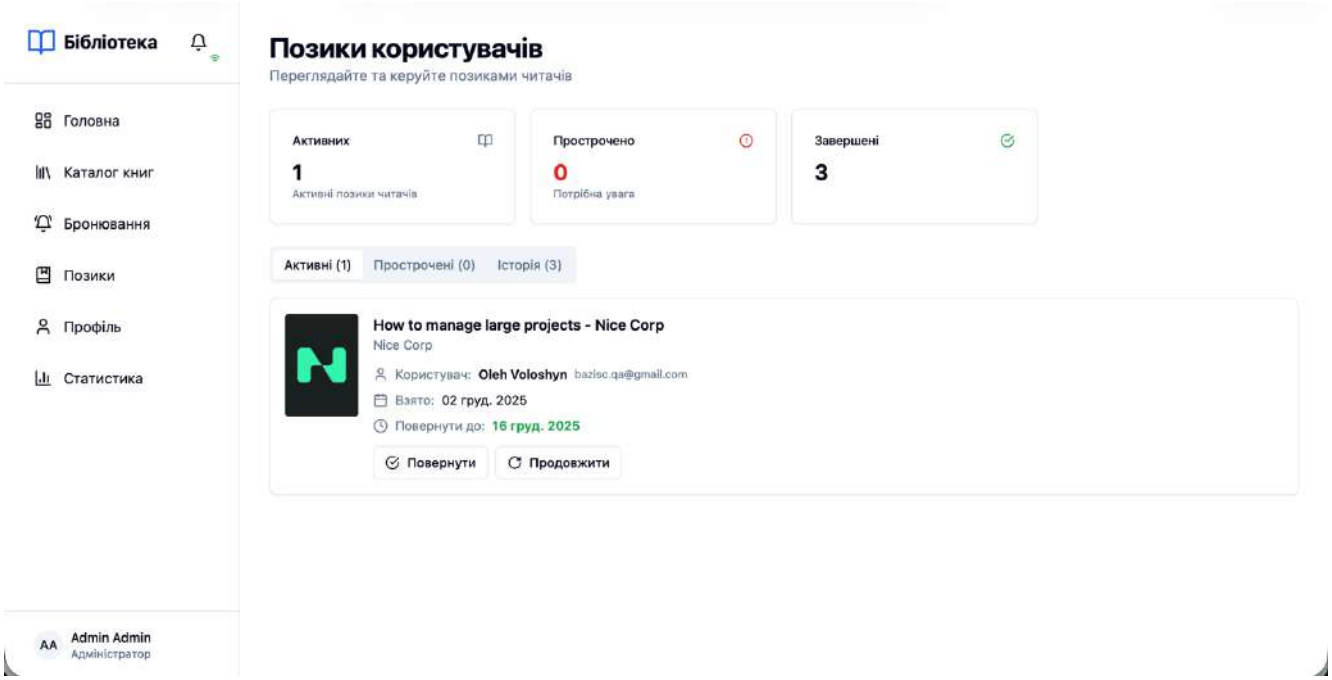


Рисунок 4.8 – Сторінка позики усіх користувачів

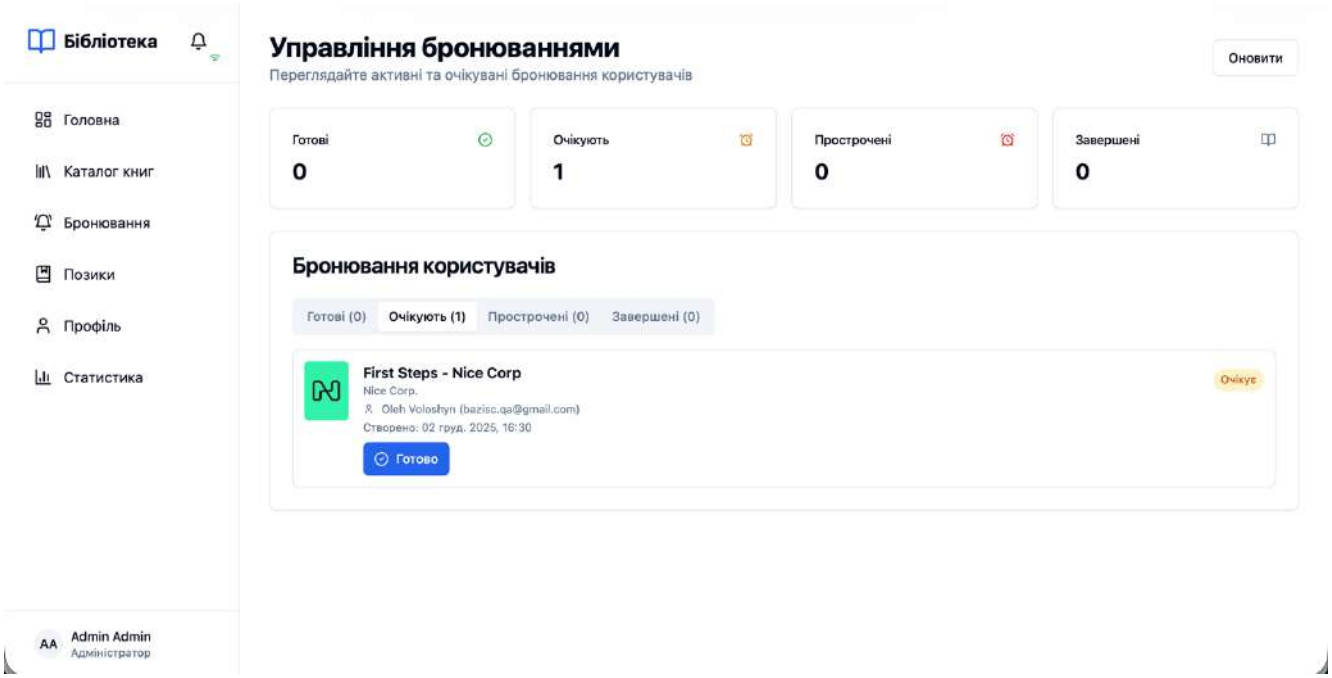


Рисунок 4.9 – Сторінка бронювань усіх користувачів

Сторінка статистики доступна лише адміністратору, вона відображає інтерактивні графіки за останні шість місяців, що демонструють кількість позик, бронювань, повернень, відгуків та інших ключових показників діяльності бібліотеки. Представлені дані дозволяють оцінити динаміку користувацької активності, виявити періоди пікового навантаження, визначити найбільш популярні книги та побачити можливі проблемні зони у функціонуванні системи. Уся інформація формується на основі агрегованих записів у базі даних, проходить попередню обробку та передається у вигляді готових структур, що значно зменшує навантаження на клієнтську частину.

Для підвищення швидкодії статистичний модуль використовує кешування результатів на стороні серверної частини. Це дозволяє зменшити кількість повторних обчислень та забезпечує стабільну швидкість завантаження навіть при великому обсязі даних. У випадку зміни інформації (наприклад, появи нової позики або повернення книги) кеш автоматично оновлюється, що гарантує актуальність відображених показників.

Інтерфейс графіків побудовано на динамічних компонентах, які адаптуються до типу даних, підтримують зміну масштабу, підсвічування вибраних значень та

інтерактивні підказки. Такий підхід забезпечує зручність аналізу, дозволяючи адміністратору швидко знайти необхідну інформацію та приймати обґрунтовані управлінські рішення.

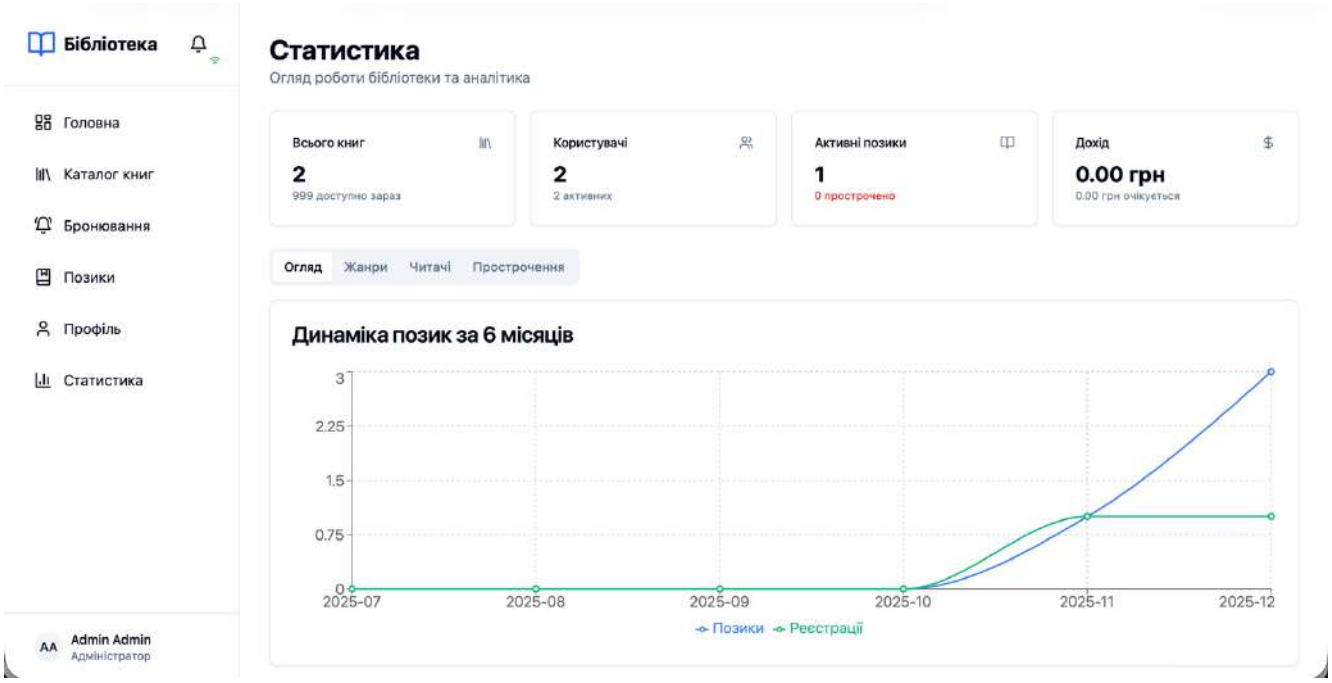


Рисунок 4.10 – Статистика бібліотеки

Профіль містить інформацію про користувача, можливість зміни пароля, оновлення персональних даних та керування налаштуваннями облікового запису.

Цей модуль є важливою частиною системи, оскільки забезпечує зручність використання та персоналізацію. Крім того, профіль реалізує механізми валідації введених даних і контроль ролей, що підвищує безпеку доступу. Завдяки інтерактивному інтерфейсу користувач може оперативно вносити зміни, що сприяє підтриманню актуальності інформації та покращує загальний досвід взаємодії з системою.

Додатково передбачено можливість відстеження історії активності, що допомагає користувачу контролювати власні дії в системі. Інтерфейс профілю оптимізовано для роботи на різних пристроях, що забезпечує доступ до функцій облікового запису навіть у мобільному середовищі.

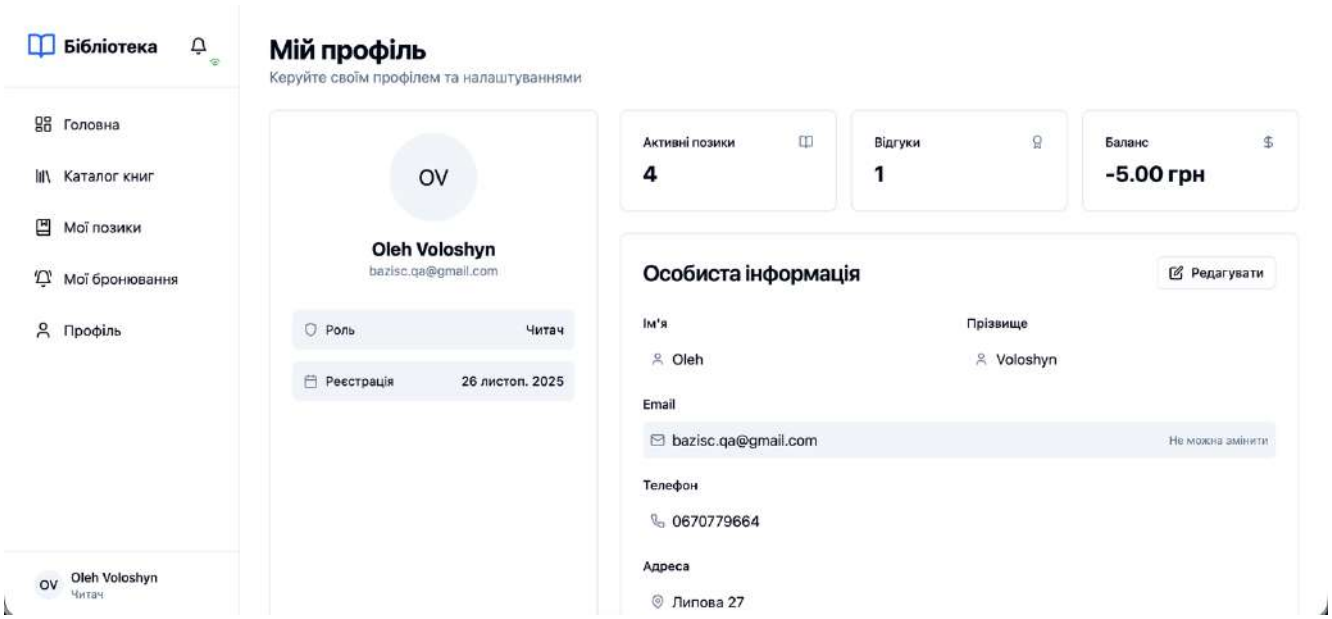


Рисунок 4.11 – Профіль користувача

4.4 Тестування інформаційної системи

Тестування веб-орієнтованої інформаційної системи бібліотеки є ключовим етапом, що забезпечує перевірку працездатності, надійності та відповідності реалізованого функціоналу вимогам, сформованим під час проєктування. Оскільки система включає серверну та клієнтську частини, а також складну структуру взаємодії підсистем – пошуку, позик, бронювань, каталогу книг, статистики та керування користувачами, – тестування проводиться комплексно та охоплює різні рівні перевірок.

Першим етапом виконувалося модульне тестування, спрямоване на перевірку окремих компонентів серверної логіки: сервісів, контролерів і допоміжних утиліт (рис 4.12). За основу бралися сценарії перевірки CRUD-операцій над ключовими сутностями: книгами, користувачами, позиками та бронюваннями. Особлива увага приділялася перевірці коректності обробки помилок, реакції системи на некоректні дані та відповідності структури відповідей REST API стандартизованому формату. Для окремих модулів використовувалися граничні значення – мінімальні й максимальні допустимі значення полів, відсутність обов’язкових параметрів, дубльовані записи та перевірки на унікальність.

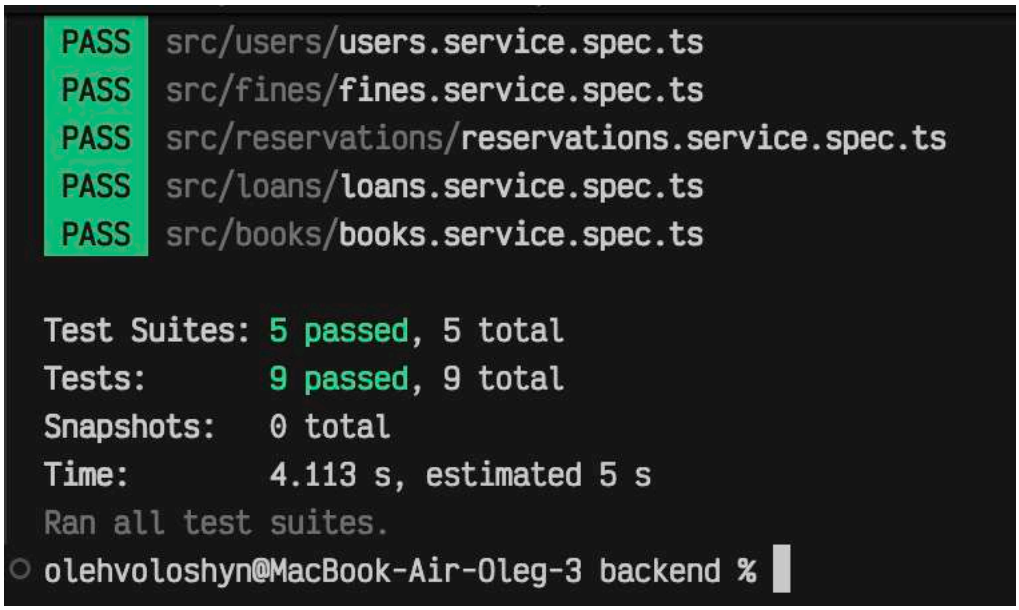


Рисунок 4.12 – Результат unit-тестування

Другим етапом проводилося інтеграційне тестування, під час якого перевірялася взаємодія між модулями серверної частини, а також між сервером і клієнтом. Тестування цього рівня було критично важливим, оскільки система синхронізує роботу багатьох підсистем: оновлення статусів книг після оформлення позики, внесення змін у статистику, оновлення індексу пошуку та формування актуальних даних у профілі користувача.

Перевірки включали такі сценарії:

1) оформлення позики та негайне відображення змін у каталозі;
2) створення бронювання та оновлення списку бронювань;
3) видалення книги адміністратором і коректна реакція пошукового модуля.

Усі ці дії виконувалися з урахуванням ролей користувачів та правил доступу.

Крім того, важливим складником є тестування інтерфейсу клієнтської частини, яке включало перевірку відображення компонентів, коректності маршрутизації, поведінки форм, валідації даних та адаптивності інтерфейсу. Окремо перевірялися життєво важливі сценарії: робота кнопок «Позичити» та «Забронювати», редагування профілю, оновлення списків позик у режимі реального часу. Під час тестування використовувалася реальна серверна частина, що дозволило оцінити поведінку застосунку в умовах фактичного обміну даними.

Наступним етапом стало навантажувальне тестування, яке дало змогу перевірити стабільність роботи системи за високої інтенсивності запитів. Використовувалися різні сценарії: одночасне звернення великої кількості користувачів до каталогу книг, масові запити до пошукової підсистеми, формування численних позик і бронювань. Метою було визначити межі продуктивності системи, час відповіді серверної частини та поведінку кешу при збільшенні навантаження. Особливу увагу приділяли використанню Redis-кешу, який суттєво знижує навантаження на базу даних під час повторних запитів.

Не менш важливим було тестування безпеки, під час якого перевірялися захист авторизаційних токенів, стійкість до підбору паролів, коректність обмеження доступу до ресурсів користувачів різних ролей, а також запобігання ін’єкційним атакам. Перевірялася поведінка системи при спробах модифікації токенів, несанкціонованого доступу до захищених маршрутів, підробки запитів або передачі некоректних структур даних.

Завершальним етапом було приймальне тестування, яке підтвердило відповідність реалізованої системи вимогам та сценаріям взаємодії, що були описані в технічному завданні. Це включало повний набір функцій: каталог, сторінку книги, позики, бронювання, статистику, профіль і адміністративний модуль. Кожен сценарій проходив перевірку з точки зору користувача, аналізувалася сталість поведінки, відсутність критичних помилок і відповідність результатів очікуванням.

Таке комплексне тестування підтвердило правильність архітектурних рішень і стабільність роботи системи в умовах різних навантажень. Отримані результати дали змогу оптимізувати окремі процеси, покращити продуктивність та забезпечити надійну роботу інформаційної системи бібліотеки.

4.5 Розгортання інформаційної системи

Розгортання веб-орієнтованої інформаційної системи бібліотеки включає підготовку серверного середовища, встановлення необхідних компонентів,

конфігурацію виконуваних сервісів та налаштування інфраструктури для забезпечення безперервної роботи системи. Процес розгортання поділяється на кілька послідовних етапів, що охоплюють інсталяцію серверної частини, підготовку клієнтської частини, розгортання бази даних, налаштування пошукового індексу Elasticsearch і запуск допоміжних сервісів, таких як Redis.

Першим етапом є підготовка серверного середовища. Зазвичай використовується Linux-сервер, що забезпечує стабільність, безпеку та можливість подальшої автоматизації. На початковому етапі встановлюється Node.js, необхідний для виконання серверної частини, а також PostgreSQL, яке є основною системою керування базами даних для проєкту. Після встановлення бази даних створюється окремий користувач, призначається ізольована база даних, а також застосовуються міграції Prisma, що формують структуру таблиць відповідно до моделі системи.

Наступним кроком розгортається серверна частина. Після копіювання коду на сервер виконуються встановлення npm-залежностей та компіляція TypeScript у JavaScript. Для забезпечення стабільної роботи застосовується процес-менеджер, наприклад PM2, який дозволяє автоматично перезапускати сервер у разі падіння, вести журнали подій та запускати застосунок у режимі кластера. У конфігураційних файлах задаються важливі параметри: змінні середовища, строки підключення до бази даних, адреси Elasticsearch та Redis, секрети JWT і визначення портів.

Окремим етапом є розгортання пошукового індексу Elasticsearch та кешуючого сервісу Redis. Elasticsearch встановлюється як окремий сервіс, після чого ініціалізується індекс для зберігання інформації про книги: поля назви, автора, опису, ключових слів і статусу доступності. Redis конфігурується для роботи в режимі пам’яті з можливістю встановлення TTL для кешованих ключів, що дозволяє оптимізувати взаємодію серверної частини та значно пришвидшити обробку повторних запитів.

Після розгортання серверної частини переходять до встановлення клієнтської частини. Збірка React-застосунку виконується за допомогою Vite, після чого

статичні файли переміщуються на веб-сервер, такий як Nginx, який обслуговує користувацькі сторінки й перенаправляє API-запити на серверну частину. Конфігурація Nginx включає маршрутизацію /api/ на порт Node.js, а також обробку сторінок SPA через fallback на index.html, що забезпечує коректну роботу React Router.

Для забезпечення безпеки та продуктивності застосовуються SSL-сертифікати, зокрема Let’s Encrypt, що гарантують шифрування трафіку, а також механізми кешування статичних файлів на стороні Nginx. Налаштовуються обмеження на максимальну кількість одночасних підключень, розмір запитів та політика CORS, що дозволяє уникнути небажаних взаємодій зі сторонніми доменами.

Фінальним етапом розгортання є налаштування системи моніторингу та журналювання. Для цього можуть використовуватися інструменти типу PM2 Monitoring, Grafana або Prometheus, які дозволяють контролювати навантаження на сервер, кількість активних запитів, час відповіді, а також відстежувати помилки та винятки. Це дозволяє оперативно реагувати на проблеми та забезпечує стабільну роботу системи у довгостроковій перспективі.

Завдяки такій структурі процесу розгортання система отримує стабільне серверне середовище, оптимізовану фронтенд-частину та налаштовані допоміжні сервіси, що разом гарантують безперебійне функціонування бібліотечного застосунку.

4.6 Оцінка якості роботи системи

Оцінка якості роботи веб-орієнтованої інформаційної системи бібліотеки є важливим підсумковим етапом, який дозволяє визначити рівень відповідності реалізованого програмного продукту поставленим вимогам, а також оцінити його працездатність, зручність використання та надійність у реальних умовах експлуатації. Якість системи розглядається комплексно – через призму

продуктивності, доступності, безпеки, функціональної повноти й задоволеності користувачів.

Першим аспектом є продуктивність, що включає швидкість обробки запитів, стабільність реакції сервера та ефективність використання ресурсів. Навантажувальне тестування за допомогою Grafana K6 [79] показало, що система здатна обробляти велику кількість паралельних запитів завдяки поєднанню асинхронної архітектури Node.js, оптимізованих REST-маршрутів та використанню Redis у якості кешу для прискорення запитів до популярних ресурсів (рис. 4.13). Середній час відповіді системи залишається стабільним навіть при збільшенні навантаження, що свідчить про коректно реалізовану логіку оптимізації запитів і структурування бізнес-операцій.

```
olehvoloshyn@MacBook-Air-Oleg-3 scripts % k6 run k6-load.js

  THRESHOLDS

    http_req_duration
    ✓ 'p(95)<800' p(95)=3.87ms

    http_req_failed
    ✓ 'rate<0.01' rate=0.00%


  TOTAL RESULTS

    checks_total.......: 5442    45.061299/s
    checks_succeeded...: 100.00% 5442 out of 5442
    checks_failed......: 0.00%   0 out of 5442

    ✓ books success
    ✓ popular success
    ✓ single book success

    HTTP
    http_req_duration..............: avg=1.79ms min=262µs med=1.48ms max=24.85ms p(90)=3.21ms p(95)=3.87ms
      { expected_response:true }...: avg=1.79ms min=262µs med=1.48ms max=24.85ms p(90)=3.21ms p(95)=3.87ms
    http_req_failed................: 0.00%  0 out of 5442
    http_reqs......................: 5442   45.061299/s

    EXECUTION
    iteration_duration.............: avg=1s     min=1s    med=1s     max=1.03s   p(90)=1s     p(95)=1.01s
    iterations.....................: 1814   15.020433/s
    vus............................: 1      min=1         max=20
    vus_max........................: 20     min=20        max=20

    NETWORK
    data_received..................: 10 MB  86 kB/s
    data_sent......................: 555 kB 4.6 kB/s
```

Рисунок 4.13 – Результат навантажувального тестування за допомогою Grafana K6

Другим важливим параметром є функціональна повнота, яка передбачає відповідність реалізованих можливостей функціональним вимогам, сформованим під час проєктування. Система надає повний набір інструментів для роботи бібліотеки: каталог книг, механізми позик і бронювань, модуль статистики, панель адміністрування, профіль користувача та засоби пошуку. Кожна з підсистем працює узгоджено, що підтверджено інтеграційним тестуванням. Функціонал не містить критичних суперечностей або логічних помилок, що дозволяє використовувати систему як основний інструмент автоматизації роботи бібліотеки.

Особливу увагу приділено юзабіліті та доступності інтерфейсу, що є ключовим показником для користувачів різних категорій – як читачів, так і адміністраторів. Інтерфейс клієнтської частини реалізований відповідно до сучасних принципів дизайну, забезпечує інтуїтивну навігацію, адаптивність до різних розмірів екранів та логічну структуризацію контенту. Проведене тестування користувацьких сценаріїв показало, що користувач може виконати основні дії – пошук, оформлення позики, перегляд бронювань – без додаткових інструкцій, що підвищує загальний рівень задоволення від роботи із системою.

Ще одним критичним параметром оцінювання є безпека системи. Реалізація авторизації через JWT, хешування паролів за допомогою стійких алгоритмів, обмеження доступу за ролями та перевірка маршрутів дають змогу захистити систему від більшості поширених загроз. Проведене тестування безпеки підтвердило, що система коректно реагує на спроби доступу до заборонених ресурсів, а механізми перевірки токенів унеможливлюють несанкціоноване втручання. Структура серверної частини мінімізує ризики ін’єкційних атак та небезпечних маніпуляцій із даними.

Не менш важливою характеристикою є масштабованість, що визначає здатність системи розширюватися відповідно до зростання потреб бібліотеки. Завдяки модульності NestJS, типобезпечності Prisma та гнучкій структурі бази даних система може бути розширена без суттєвих змін архітектури. Підтримка кешування, асинхронна обробка запитів і можливість горизонтального

масштабування сервера дозволяють обслуговувати більше користувачів без зниження продуктивності.

Загальні результати оцінки свідчать про високий рівень якості програмного забезпечення. Система є функціонально завершеною, стабільною, безпечною та зручною у користуванні. Вона відповідає сучасним стандартам розробки веб-застосунків та може бути рекомендована для впровадження як інформаційна система автоматизації бібліотечних процесів.

4.7 Висновки

У четвертому розділі було здійснено всебічний аналіз програмної реалізації веб-орієнтованої інформаційної системи бібліотеки та продемонстровано практичне втілення розроблених методів, моделей і алгоритмів у вигляді функціонального програмного продукту. Реалізація серверної частини забезпечила формування стійкої, модульної архітектури, що підтримує масштабованість, розширюваність та ефективну взаємодію із зовнішніми сервісами. Принципи інверсії залежностей, структурованість модульного підходу NestJS та використання ORM Prisma дозволили створити систему, здатну обробляти складні бізнес-процеси з високою точністю та стабільністю.

Клієнтська частина системи була реалізована з використанням сучасних веб-технологій, що забезпечили зручний, адаптивний та інтуїтивно зрозумілий інтерфейс. React, Redux Toolkit, TailwindCSS та ShadCN/UI дали змогу сформувати динамічний застосунок, здатний оперативно реагувати на зміни стану та відображати оновлену інформацію без перезавантаження сторінки. Така архітектура інтерфейсу сприяє підвищенню залученості користувачів і покращує загальну якість взаємодії з системою.

У підрозділі, присвяченому реалізації функціональних можливостей системи, було описано ключові модулі, які складають основу роботи бібліотеки: каталог, сторінку книги, механізми позик і бронювань, профіль користувача, панель адміністратора та модуль статистики. Для кожного з них визначено логіку

роботи, елементи інтерфейсу та механізми взаємодії з серверною частиною. Завдяки цьому система стала комплексним інструментом автоматизації бібліотечних процесів, що охоплює всі необхідні сценарії роботи користувача.

Тестування системи підтвердило її стабільність, продуктивність і відповідність функціональним вимогам. Модульні, інтеграційні, навантажувальні та безпекові тести продемонстрували коректність алгоритмів, здатність системи працювати під високим навантаженням і ефективність архітектурних рішень, прийнятих під час розробки. Особливу увагу було приділено захисту даних, структурі ролей і механізмам автентифікації, що забезпечує захищене середовище для роботи користувачів.

Окремим етапом стало розгортання системи, яке продемонструвало практичну можливість її використання в реальному середовищі. Налаштування серверної інфраструктури, деплой клієнтської частини, конфігурація Nginx [81], організація зв’язку з Redis і Elasticsearch – усе це створило повноцінну робочу платформу, готову до експлуатації. Застосування кращих практик DevOps [82] підвищило стабільність роботи системи та забезпечило можливість її подальшого масштабування.

Узагальнюючи результати роботи, можна стверджувати, що реалізована інформаційна система повністю відповідає вимогам, сформульованим під час проєктування. Вона є багатофункціональною, гнучкою, технологічно сучасною та готовою до впровадження в діяльність бібліотечних установ. Розділ 4 продемонстрував, що обрана архітектура, інструменти розробки та підходи до організації роботи системи були обґрунтованими та ефективними, а створений програмний продукт є якісним завершенням усієї розробленої методики побудови веб-орієнтованої інформаційної системи бібліотеки.

ВИСНОВКИ

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено метод та веб-орієнтовану інформаційну систему бібліотеки, що забезпечує автоматизацію ключових процесів обліку, пошуку, бронювання та видачі бібліотечних ресурсів. Проведені дослідження дозволили створити комплексну архітектуру, яка поєднує серверні та клієнтські технології, спрямовані на підвищення ефективності роботи бібліотечних установ.

У першому розділі здійснено огляд відомих рішень та сучасних інформаційних систем, що застосовуються у бібліотечній сфері. Проаналізовано функціональні можливості, переваги та недоліки популярних платформ, показано тенденції розвитку бібліотечних технологій та визначено напрями, у яких існуючі рішення потребують удосконалення. Це дозволило обґрунтувати необхідність розробки власної, адаптивної та гнучкої системи.

У другому розділі розроблено проєктну частину інформаційної системи, що включає моделювання вимог, створення UML-діаграм, розробку структури даних, архітектури системи та опис бізнес-процесів. Визначено ключові елементи взаємодії підсистем, обґрунтовано вибір технологій – NestJS, PostgreSQL, Prisma ORM, Redis, Elasticsearch, React, Redux Toolkit. Результатом стало формування узгодженої моделі системи, придатної до програмної реалізації.

У третьому розділі розроблено методи та алгоритми роботи веб-орієнтованої інформаційної системи. Деталізовано методи побудови серверної частини, організації доступу до даних, роботи пошукового модуля, кешування, обробки клієнтських запитів та механізмів автентифікації. Представлено алгоритм взаємодії підсистем, що гарантує цілісність даних, узгодженість станів та високу продуктивність системи під час реальної експлуатації.

У четвертому розділі здійснено програмну реалізацію системи, описано механізми роботи серверної та клієнтської частин, реалізацію основних функціональних можливостей та логіку користувацького інтерфейсу. Проведено тестування системи, яке підтвердило її стабільність, працездатність, відповідність

проєктним вимогам та готовність до розгортання. Розглянуто детальний процес деплою, налаштування інфраструктури та оцінку якості роботи системи.

Набула подальшого розвитку інформаційна технологія автоматизації бібліотечних процесів, що поєднує сучасні методи проєктування, алгоритмічну базу, модульну архітектуру та принципи побудови високонавантажених веб-систем. Запропонований підхід дозволяє масштабувати систему, адаптувати її до потреб різних бібліотек та розширювати функціональні можливості відповідно до вимог користувачів.

Впровадження результатів роботи дозволило створити ефективний інструмент управління бібліотечними ресурсами, що забезпечує підвищення продуктивності роботи бібліотеки, спрощення взаємодії користувачів із системою, зменшення операційних затрат, а також підвищення доступності інформаційних ресурсів. Система може бути інтегрована в бібліотеки різного масштабу та використовуватися як основа для подальших наукових і практичних розробок.

За темою кваліфікаційної роботи магістра опублікована одна стаття у фаховому науковому виданні, що підтверджує актуальність проведених досліджень та наукову цінність отриманих результатів.

За темою кваліфікаційної роботи опубліковані тези доповіді (додаток А), та взято участь у IV Міжнародна конференція молодих учених та здобувачів вищої освіти «Філософські виміри техніки» (26 листопада 2025 р, м. Тернопіль):

1) Волошин О.В. Веб-орієнтована інформаційна система бібліотеки. *Матеріали IV Міжнародна конференція молодих учених та здобувачів вищої освіти «Філософські виміри техніки»*, м. Тернопіль, 26 листопада 2025 р, С. 94-96.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Koha Library Software Community. Official Documentation and Wiki. URL: https://koha-community.org (дата звернення: 08.10.2025).

2. Greenstone Wiki. URL: https://wiki.greenstone.org/doku.php?id=index (дата звернення: 09.10.2025).

3. VuFind Documentation. URL: https://vufind.org/wiki/ (дата звернення: 10.10.2025).

4. Singh M., Sanaman G. Open source integrated library management systems: Comparative analysis of Koha and NewGenLib. *The Electronic Library*. 2012. Vol. 30, No. 6. P. 809-832.

5. Komara R., Pahlevi M., Hernawan R. F. Web-Based Library Information System Design. *International Journal of Research and Applied Technology (INJURATECH)*. 2023. Vol. 3, No. 1. P. 161-170.

6. Express.js. Fast, Unopinionated, Minimalist Web Framework for Node.js. URL: https://expressjs.com (дата звернення: 12.10.2025).

7. NestJS. A Progressive Node.js Framework for Building Efficient Server-Side Applications. URL: https://nestjs.com (дата звернення: 12.10.2025).

8. Dublin Core Metadata Initiative. Metadata Element Set, Version 1.1. URL: https://www.dublincore.org/specifications/dublin-core/dces/ (дата звернення: 13.10.2025).

9. Open Archives Initiative. Protocol for Metadata Harvesting (OAI-PMH) Standard Specification. URL: https://www.openarchives.org/OAI/openarchivesprotocol.html (дата звернення: 14.10.2025).

10. DSpace Open Source Repository System. URL: https://dspace.lyrasis.org (дата звернення: 14.10.2025).

11. Node.js Documentation. URL: https://nodejs.org/en/docs (дата звернення: 15.10.2025).

12. CORAL ERMS. URL: https://coral-erm.org (дата звернення: 15.10.2025).

13. Pressman R. S. Software Engineering: A Practitioner's Approach. Palgrave macmillan, 2005.

14. Sommerville I. Software Engineering. America: Pearson Education Inc, 2011.

15. Fowler M. UML Distilled: A Brief Guide to the Standard Object Modeling Language. Boston : Addison-Wesley Professional, 2018.

16. About the Unified Modeling Language Specification Version 2.5.1. URL: https://www.omg.org/spec/UML (дата звернення: 20.10.2025).

17. Connolly T., Begg C. Database systems: a practical approach to design, implementation, and management. Pearson Education, 2005.

18. Silberschatz A., Korth H., Sudarshan S. Database system concepts. Vol. 5. New York: Mcgraw-hill, 2002.

19. Arlow J., Neustadt I. UML 2 and the unified process: practical object-oriented analysis and design. Pearson Education, 2005.

20. Unhelkar B. Software engineering with UML. Auerbach Publications, 2017.

21. Larman C. Applying UML and patterns. Vol. 2. Upper Saddle River: Prentice Hall, 1998.

22. Bennett S., McRobb S., Farmer R. Object-Oriented Systems Analysis and Design Using UML. McGraw-Hill, 2010.

23. UML 2.0 Infrastructure. URL: https://www.omg.org/spec/UML/2.4.1/Infrastructure/PDF (дата звернення: 25.10.2025).

24. Ambler S. Agile modeling: effective practices for extreme programming and the unified process. John Wiley & Sons, 2002.

25. Elmasri R., Navathe S. Fundamentals of database systems. 7th ed. 2016.

26. Coronel C., Morris S. Database systems: design, implementation, and management. Boston: Cengage learning, 2016. 784 p.

27. Revesz P. Introduction to databases. Texts in Computer Science, 2010.

28. PostgreSQL: Documentation. URL: https://www.postgresql.org/docs (дата звернення: 29.10.2025).

29. Gormley C., Tong Z. Elasticsearch: the definitive guide: a distributed real-time search and analytics engine. O'Reilly Media, Inc., 2015.

30. Han J., Pei J., Tong H. Data mining: concepts and techniques. Morgan kaufmann, 2022.

31. Bass L., Clements P., Kazman R. Software architecture in practice. Addison-Wesley Professional, 2021.

32. Richards M., Ford N. Fundamentals of software architecture: an engineering approach. O'Reilly Media, 2020.

33. Martin R.C. Clean architecture: a craftsman's guide to software structure and design. Prentice Hall Press, 2017.

34. Prisma Documentation. URL: https://www.prisma.io/docs (дата звернення: 04.11.2025).

35. Kathare N., Reddy O.V., Prabhu V. A comprehensive study of elasticsearch. *International journal of science and research (IJSR)*. 2020.

36. Docker Documentation. URL: https://docs.docker.com (дата звернення: 05.11.2025).

37. White S., Miers D. BPMN modeling and reference guide: understanding and using BPMN. Future Strategies Inc., 2008.

38. Dumas M., La Rosa M., Mendling J., Reijers H. Fundamentals of Business Process Management. Berlin ; Heidelberg : Springer, 2018.

39. Benantar M. Access Control Systems: Security, Identity Management and Trust Models. New York: Springer, 2006. P. 55-84.

40. Xie I., Matusiak K. Discover digital libraries: Theory and practice. Elsevier, 2016.

41. Provost F., Fawcett T. Data Science for Business: What you need to know about data mining and data-analytic thinking. O'Reilly Media, Inc., 2013.

42. About the Business Process Model and Notation Specification Version 2.0.2. URL: https://www.omg.org/spec/BPMN (дата звернення: 10.11.2025).

43. Aggarwal, K. K. Software engineering. New Age International, 2005.

44. NestJS – Official Documentation. URL: https://docs.nestjs.com/ (дата звернення: 12.11.2025).

45. Coronel C., Morris S., Rob P. Database Systems: Design, Implementation, and Management. Boston: Cengage Learning, 2016. P. 221-256.

46. Enhance your database connections with Prisma Accelerate's auto-scaling. URL: https://www.prisma.io/blog/introducing-auto-scaling-for-prisma-accelerate (дата звернення: 14.11.2025).

47. Elastic fundamentals. URL: https://www.elastic.co/docs/get-started (дата звернення: 16.11.2025).

48. Redis Docs. URL: https://redis.io/docs (дата звернення: 16.11.2025).

49. Kleppmann M. Designing data-intensive applications. 2019.

50. Software Architecture Patterns. Martin Fowler Wiki. URL: https://martinfowler.com/architecture/ (дата звернення: 18.11.2025).

51. Robert C.M. Agile software development. Martin: Prentice Hall, 2003.

52. Clean Architecture Overview. Robert C. Martin Official Blog. URL: https://blog.cleancoder.com/uncle-bob/2012/08/13/the-clean-architecture.html (дата звернення: 19.11.2025).

53. Beynon-Davies P. Database systems. Basingstoke, UK: Palgrave Macmillan, 2004. p. 61.

54. Turnbull D., Berryman J. Relevant search: with applications for Solr and Elasticsearch. Simon and Schuster, 2016.

55. Connolly T., Begg C. Database Systems: A Practical Approach to Design, Implementation, and Management. 2015.

56. Zobel J., Moffat A. Inverted files for text search engines. *ACM computing surveys (CSUR)*. 2006.

57. Manning C.D. Introduction to Information Retrieval. Syngress Publishing, 2008.

58. Baeza-Yates R., Ribeiro-Neto B. Modern Information Retrieval. Vol. 463. New York: ACM press, 1999.

59. Elastic Docs. URL: https://www.elastic.co/guide (дата звернення: 28.11.2025).

60. Hearst M. User interfaces for search. *Modern Information Retrieval*. 2011. P. 21-55.

61. Flynn M. J. Computer organization and architecture. In: Operating Systems: An Advanced Course. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005. p. 17-98.

62. Kleppmann M. Designing data-intensive applications: The big ideas behind reliable, scalable, and maintainable systems. O'Reilly Media, Inc., 2017.

63. Distributed Systems: Principles and Paradigms. URL: https://vowi.fsinf.at/images/b/bc/TU_Wien-Verteilte_Systeme_VO_%28G%C3%B6schka%29_-_Tannenbaum-distributed_systems_principles_and_paradigms_2nd_edition.pdf (дата звернення 06.12.2025)

64. Richards M. Microservices vs. service-oriented architecture. Sebastopol: O'Reilly Media, 2015. P. 22-24.

65. Evans E., Szpoton R. Domain-driven design. Helion, 2015.

66. JSON Web Token (JWT). RFC 7519 URL: https://www.rfc-editor.org/rfc/rfc7519 (дата звернення: 06.12.2025).

67. Fowler M. Patterns of Enterprise Application Architecture. Addison-Wesley, 2012. P. 107-137.

68. Service Layer. Martin Fowler – Enterprise Application Architecture. URL: https://martinfowler.com/eaaCatalog/serviceLayer.html (дата звернення: 07.12.2025).

69. OWASP Foundation. *Authorization* Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authorization_Cheat_Sheet.html (дата звернення: 07.12.2025).

70. Skanda C., Srivatsa B., Premananda B.S. Secure hashing using bcrypt for cryptographic applications. *2022 IEEE North Karnataka Subsection Flagship International Conference (NKCon)*. IEEE, 2022. p. 1-5.

71. Dumas M., La Rosa M., Mendling J., Reijers H. Fundamentals of business process management. Springer-Verlag, 2018.

72. Wang P. P., Ming X. G., Li D., Kong F. B., Wang L., Wu Z. Y. Modular development of product service systems. *Concurrent engineering.* 2011. P. 85-96.

73. Newman S. Building microservices: designing fine-grained systems. O'Reilly Media, Inc., 2021.

74. React – Official Documentation. URL: https://react.dev (дата звернення: 10.12.2025).

75. Redux Toolkit – Official Documentation. URL: https://redux-toolkit.js.org (дата звернення: 10.12.2025).

76. TailwindCSS – Official Documentation. URL: https://tailwindcss.com (дата звернення: 10.12.2025).

77. Nielsen J. Usability engineering. Morgan Kaufmann, 1994.

78. Norman D. The design of everyday things. MIT Press, 2013.

79. Grafana Labs. k6 Load Testing Documentation. URL: https://k6.io/docs (дата звернення: 10.12.2025).

80. ISO/IEC 25010:2023. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuaRE) – Product quality mode. URL: https://www.iso.org/standard/78176.html (дата звернення: 30.11.2025).

81. Nginx Documentation. URL: https://nginx.org/en/docs (дата звернення: 10.12.2025).

82. Exploring Docker for DevOps: What It Is and How It Works. URL: https://www.docker.com/blog/docker-for-devops/ (дата звернення: 10.12.2025)

83. IEEE Computer Society. Guide to the Software Engineering Body of Knowledge (SWEBOK), Version 3.0. URL: https://www.computer.org/education/bodies-of-knowledge/software-engineering (дата звернення: 10.12.2025).

84. What is Distributed Computing? – Distributed Computing Explained – AWS. URL: https://aws.amazon.com/what-is/distributed-computing/ (дата звернення: 10.12.2025).

ДОДАТОК А

(обов’язковий)

КОПІЯ ОПУБЛІКОВАНОЇ НАУКОВОЇ СТАТТІ

1) Волошин О.В. Веб-орієнтована інформаційна система бібліотеки. *Матеріали IV Міжнародна конференція молодих учених та здобувачів вищої освіти «Філософські виміри техніки»*, м. Тернопіль, 26 листопада 2025 р, С. 94-96.

УДК 004.9
Волошин О. В.
Науковий керівник – Медзатий Д.М., канд. техн. наук, доцент
Хмельницький національний університет, Україна

ВЕБ-ОРІЄНТОВАНА ІНФОРМАЦІЙНА СИСТЕМА БІБЛІОТЕКИ

Веб-орієнтована інформаційна система є критично важливою для сучасної бібліотеки, оскільки вона забезпечує цілодобовий дистанційний доступ до ресурсів, усуваючи географічні та часові обмеження. Її архітектура складається з ключових модулів: каталогізації, обслуговування, веб-порталу та адміністрування, що автоматизують внутрішні процеси та підвищують ефективність роботи. Основними перевагами є розширення аудиторії та централізація даних, хоча система має обмеження, пов'язані із залежністю від інтернет-інфраструктури, високими початковими витратами та потребою у кваліфікованих IT-фахівцях для підтримки кібербезпеки.
Ключові слова: інформаційна система, автоматизація процесів, дистанційний доступ 24/7, IT-інфраструктура.

Voloshyn O. V.
Scientific supervisor – Medzatyi D. M.., Ph.D., Assoc. Prof.
Khmelnytskyi National University, Ukraine

WEB-BASED LIBRARY INFORMATION SYSTEM

Актуальність розроблення веб-орієнтованої інформаційної системи для бібліотеки зумовлена низкою сучасних тенденцій та потреб, які виникають в умовах цифровізації суспільства та зростання вимог користувачів до швидкого й дистанційного доступу до інформації [1]. У традиційному форматі бібліотеки стикаються з обмеженнями фізичного простору, робочого часу та необхідністю ручної обробки великих обсягів даних, що уповільнює обслуговування [2]. Веб-орієнтована інформаційна система дозволяє усунути ці географічні та часові бар'єри, надаючи користувачам можливість цілодобового доступу до електронного каталогу, інформації про наявність видань та електронних ресурсів бібліотеки з будь-якої точки світу, де є інтернет-з'єднання. Це значно розширює аудиторію бібліотеки, включаючи віддалених студентів, дослідників та всіх, хто не може фізично відвідати заклад. Крім цього, така система оптимізує внутрішні робочі процеси бібліотеки. Автоматизація процесів, таких як комплектування фонду, інвентаризація, облік видачі та повернення літератури, а також формування звітів, знижує ймовірність людських помилок і звільняє персонал для виконання більш кваліфікованих завдань, наприклад, надання консультацій чи організації освітніх заходів. Це підвищує ефективність роботи бібліотеки та якість обслуговування. Веб-орієнтована платформа також є центральним комунікаційним хабом, дозволяючи швидко інформувати користувачів про нові надходження, події та зміни в режимі роботи, а також впроваджувати персоналізовані сервіси, наприклад, рекомендації літератури на основі історії читання. З огляду на необхідність збереження та ефективного управління зростаючими електронними колекціями та мультимедійними ресурсами, розроблення сучасної веб-орієнтованої інформаційної системи є критично важливим кроком для перетворення бібліотеки на динамічний та конкурентоспроможний інформаційно-освітній центр, що відповідає вимогам інформаційного суспільства.

Архітектура веб-орієнтованої інформаційної системи бібліотеки складається з кількох взаємопов'язаних функціональних модулів, кожен з яких виконує свій набір критично важливих функцій, що забезпечують її цілісну роботу. Центральним елементом є модуль управління фондом та каталогізації, головне призначення якого полягає у створенні, зберіганні та систематизації всіх бібліографічних та облікових даних фонду; його функції охоплюють детальний опис нових надходжень у стандартизованих форматах, таких як MARC, ведення точного обліку кожного примірника з фіксацією інвентарних номерів та місць зберігання, а також забезпечення імпорту й експорту даних для обміну з іншими інформаційними системами.

Цей модуль нерозривно пов'язаний з модулем обслуговування (абонемент), який повністю автоматизує взаємодію з читачами, включаючи функції реєстрації нових користувачів та управління їхніми обліковими картками. Основна його робота зосереджена на операціях видачі та повернення літератури, де система фіксує терміни користування, автоматично контролює заборгованість і розраховує штрафні санкції за прострочення, а також забезпечує функціонал бронювання видань, які наразі знаходяться на руках у інших читачів.

Зовнішньою, користувацькою частиною системи є веб-орієнтований портал, що виконує роль віртуального обличчя бібліотеки та надає функції доступу до її ресурсів у режимі 24/7. Користувачі можуть здійснювати розширений та фасетний пошук у каталозі, оперативно переглядати інформацію про наявність та фізичне місцезнаходження потрібних видань, а також через особистий кабінет отримувати персоналізовані послуги. До ключових функцій порталу також належить можливість онлайн-замовлення примірників, продовження терміну користування та безпосередній доступ до повнотекстових електронних колекцій і цифрових архівних матеріалів.

За коректність роботи всієї системи відповідає модуль адміністрування та звітності, який забезпечує внутрішнє управління та контроль. Його функціонал включає адміністрування прав та ролей користувачів, налаштування всіх параметрів системи, від графіків роботи до бібліотечних правил, а також гарантує безпеку даних через механізми резервного копіювання. Крім того, цей модуль виконує критичну функцію генерації статистичних та аналітичних звітів щодо динаміки фондокористування, популярності окремих видань та ефективності обслуговування, надаючи необхідні дані для стратегічного планування та оптимізації бібліотечної діяльності.

Впровадження веб-орієнтованої інформаційної системи для бібліотеки надає суттєві переваги. Найважливішою перевагою є доступність ресурсів, оскільки система усуває географічні та часові бар'єри, надаючи користувачам цілодобовий віддалений доступ до каталогу та електронних фондів з будь-якого пристрою, що має вихід в інтернет. Це значно розширює читацьку аудиторію та підвищує загальний рівень задоволеності користувачів. Далі, система забезпечує підвищення ефективності внутрішніх процесів, оскільки автоматизація рутинних операцій знижує навантаження на персонал, мінімізує помилки та прискорює обслуговування. Також до переваг належить централізація даних, що забезпечує єдине, актуальне джерело інформації про весь фонд, що критично важливо для управління великими колекціями. Нарешті, така система дозволяє впроваджувати інноваційні сервіси, включаючи персоналізовані рекомендації, інтерактивні онлайн-замовлення та швидкий доступ до повнотекстових цифрових ресурсів, що перетворює бібліотеку на сучасний інформаційний центр.

Водночас, впровадження та експлуатація такої системи має й певні обмеження. Ключовим викликом є залежність від технологій та інфраструктури, адже для безперебійної роботи як користувачі, так і бібліотека повинні мати стабільний доступ до Інтернету та належне апаратне забезпечення. Крім цього, виникає потреба у кваліфікованому персоналі, здатному обслуговувати та підтримувати складну IT-інфраструктуру, а також проводити навчання для існуючого бібліотечного штату. Нарешті, існує постійний ризик, пов'язаний із захистом даних, оскільки зберігання великих обсягів бібліографічної та персональної інформації вимагає надійних заходів кібербезпеки для запобігання несанкціонованому доступу чи втраті даних, що також є важливим обмеженням і викликом.

Джерела та література:

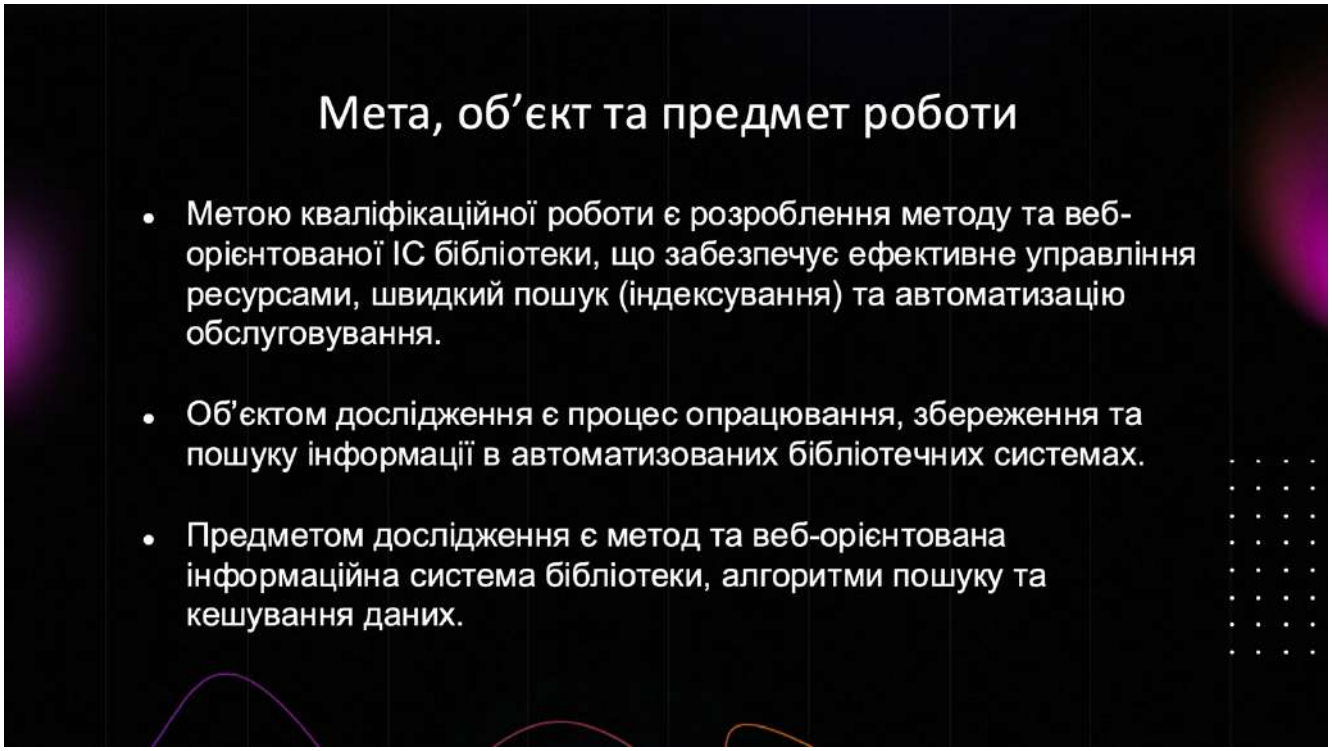
1. Ivashkevych O. DIGITAL TRANSFORMATION (DIGITALIZATION) OF LIBRARIES IN UKRAINE: CURRENT STATE AND PERSPECTIVES. Scientific journal "Library Science. Record Studies. Informology". 2021. No. 2. P. 50–56.

2. Nakaziba S., Ngulube P. A model for enhancing digital transformation through technology-related continuing professional development activities in academic libraries in context. Discover Education. 2024. Vol. 3, no. 1.

ДОДАТОК Б

(обов’язковий)

ПРЕЗЕНТАЦІЯ ДО ЗАХИСТУ КВАЛІФІКАЦІЙНОЇ РОБОТИ



Наукова новизна

Наукова новизна отриманих результатів полягає в удосконаленому методі побудови веб-орієнтованої бібліотечної системи, який відрізняється поєднанням алгоритмів повнотекстового пошуку, кешування даних і динамічного ранжування результатів для підвищення точності й швидкодії. Набула подальшого розвитку інформаційна технологія автоматизованого управління бібліотечним фондом, що забезпечує інтеграцію пошукового модуля з індексованим сховищем даних та механізмами оперативного оновлення інформації в реальному часі.

Практична значимість

Практична значимість отриманих результатів полягає у розробленні веб-орієнтованої інформаційної системи бібліотеки, що може бути використана навчальними закладами, публічними бібліотеками, корпоративними центрами знань та електронними архівами. Система забезпечує пошук за індексованими даними, автоматизацію процесів позики та бронювання, формування статистичних звітів, збереження бібліографічної інформації, рольову модель доступу та сучасний інтерфейс взаємодії з користувачами. Ключовими перевагами є зручність використання, можливість масштабування, підтримка різних сценаріїв доступу, висока швидкодія та адаптивність до збільшення обсягів даних.

Актуальність дослідження

Актуальність теми роботи полягає в необхідності створення сучасної веб-орієнтованої інформаційної системи бібліотеки, яка здатна забезпечити автоматизацію внутрішніх процесів, підтримку великих обсягів даних, зручність для користувачів та високий рівень продуктивності. Сьогодні багато бібліотечних установ потребують оновлення своїх інформаційних систем або впровадження нових, оскільки традиційні програмні рішення часто не забезпечують достатнього рівня інтерактивності, масштабованості та швидкодії. Тому дослідження, спрямоване на розробку методу та програмної реалізації веб-орієнтованої інформаційної системи бібліотеки, є важливим і своєчасним.

Постановка задачі

Поставлена мета досягається розв'язанням таких завдань:

- розробити методи та моделі побудови серверної частини, структури даних і взаємодії підсистем;
- створити ключові алгоритми системи – пошуку, кешування, обробки запитів, автентифікації та авторизації;
- розробити клієнтську частину та програмну реалізацію інформаційної системи бібліотеки;
- оцінити якість, масштабованість та можливості інтеграції розробленої системи в інфраструктуру бібліотеки.

Огляд існуючих відкритих рішень

Приклади існуючих інформаційних систем бібліотеки.

VuFind

Greenstone

Koha

Метод організації доступу до даних

У межах методу організації доступу до даних виокремлюється така послідовність дій:

- формалізація структури даних, під час якої визначаються сутності (книги, користувачі, позики, бронювання), їх атрибути та зв'язки;
- побудова шару абстракції доступу, який ізолює бізнес-логіку від конкретних механізмів роботи з базою даних; на цьому етапі застосовуються об'єктно-реляційні моделі, що дозволяють зменшити складність запитів та інкапсулюють взаємодію з PostgreSQL;

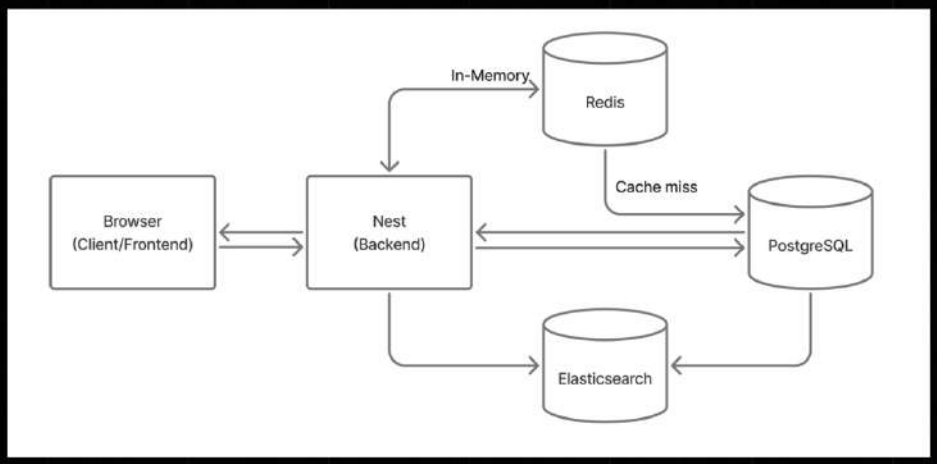
Метод організації доступу до даних (2)

- створення адаптерів для оперативного читання, що використовують Redis для кешування часто вживаних запитів, зокрема даних про книги та статистику активності читача;
- формування повнотекстового індексу, який зберігає інформацію про назви, авторів, жанри та описи книг у структурі inverted index, що значно прискорює пошукові операції;

Метод організації доступу до даних (3)

- застосування транзакційного методу, який забезпечує узгодженість операцій під час створення, редагування або видалення даних;
- синхронізація між підсистемами, у тому числі оновлення індексу та очищення кешу після будь-якої операції, що змінює стан даних.

Діаграма спроєктованої архітектури системи



Обрані програмні технології для бекенд частини

NodeJS

NestJS

Prisma

PostgreSQL

Redis

Elasticsearch

Обрані програмні технології для фронтенд частини

React

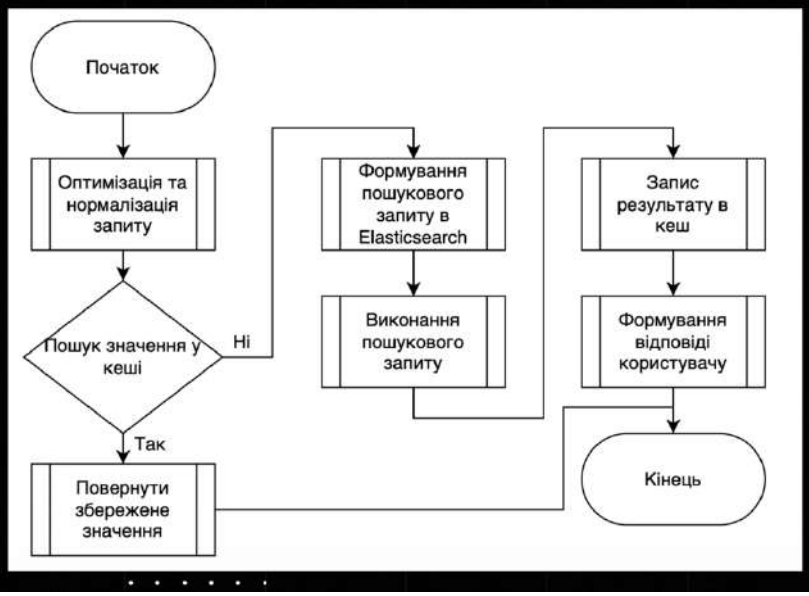
Redux

Axios

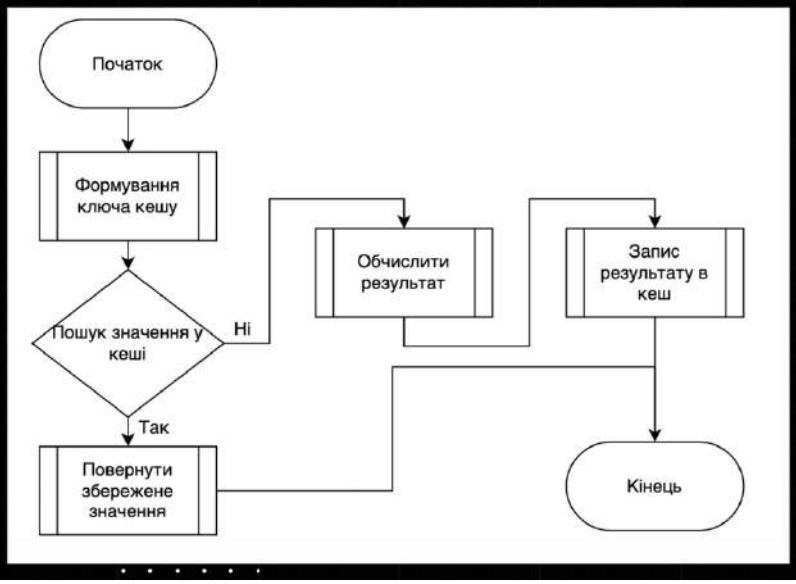
TailwindCSS

ShadCN/UI

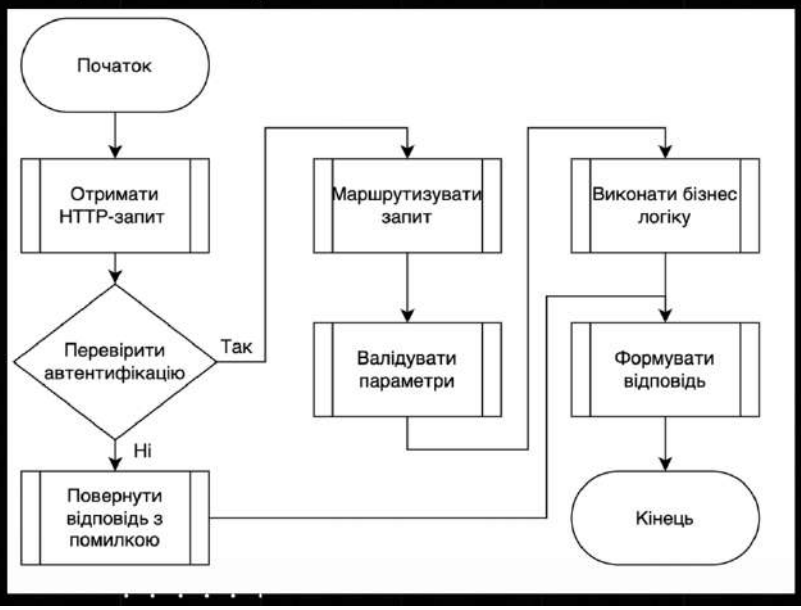
Алгоритм роботи пошукового модулю



Алгоритм роботи кешування даних



Алгоритм обробки клієнтських запитів



Подальші дослідження та розвиток

- Подальші дослідження можуть бути спрямовані на розширення функціональних можливостей веб-орієнтованої інформаційної системи бібліотеки шляхом впровадження інтелектуальних. сервісів рекомендацій на основі аналізу поведінки користувачів
- Перспективним напрямом є інтеграція з зовнішніми освітніми платформами, науковими репозитаріями та університетськими інформаційними системами.
- Приділити увагу підвищенню рівня безпеки, масштабованості та адаптації системи до хмарних середовищ і мобільних клієнтів.

Висновок

У роботі за результатами виконаних теоретичних та практичних досліджень розроблено метод та веб-орієнтовану інформаційну систему бібліотеки, що забезпечує автоматизацію ключових процесів обліку, пошуку, бронювання та видачі бібліотечних ресурсів. Проведені дослідження дозволили створити комплексну архітектуру, яка поєднує серверні та клієнтські технології, спрямовані на підвищення ефективності роботи бібліотечних установ.

Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Олег ВОЛОШИН

Співавтор:

Назва: Метод та веб-орієнтована інформаційна система бібліотеки

Експерт: Дмитро МЕДЗАТИЙ

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:3.4%

Коефіцієнт подібності 2:0.2%

Мікропробіли: 0

Заміна букв: 4

Інтервали: 0

Білі знаки: 1

Дата створення звіту: 2025-12-12 09:15:41.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

Дата експерт

Fri Dec 12 09:46:35 EET 2025, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 0.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 11%**

	Document		Sum coincidence on the DB	
ID: 252627 Title: МКР Метод та веб-орієнтована інформаційна система бібліотеки Added in a DB: 2025-12-12 Authors: Олег ВОЛОШИН Heads: Дмитро МЕДЗАТИЙ Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	105458	771	2965 (3%)	47 (6%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Олег ВОЛОШИН

Тема: Метод та веб-орієнтована інформаційна система бібліотеки

Спеціальність: 126 «Інформаційні системи та технології»

Обсяг дипломної роботи:

Кількість сторінок записки 70

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розроблення методу та веб-орієнтованої інформаційної системи бібліотеки, яка забезпечує ефективне управління бібліотечними ресурсами, швидкий доступ до електронного каталогу, повнотекстовий пошук із використанням індексованих структур даних, а також автоматизацію ключових операцій бібліотечного обслуговування.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: У першому розділі проведений огляд відомих методів та рішень. У другому розділі виконано проєктування інформаційної системи бібліотеки. У третьому розділі розроблено метод та алгоритм функціонування інформаційної системи бібліотеки. У четвертому розділі описано реалізацію інформаційної системи бібліотеки.

4. Позитивні сторони роботи:

5. Негативні сторони роботи: мало уваги приділено формалізації методу

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно з діючими стандартами оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на середньому науково-технічному рівні.

8. Інші зауваження: —

9. Оцінка дипломної роботи: добре/С (80).

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Олександр БОРЛЯК, зав. каф. КІ ХНУ

"__" _____ 2025 р. _____ (підпис)

Зав. кафедри КІС
д-р. філософії Ользі ПАВЛОВІЙ

Олег ВОЛОШИН
ПІБ здобувача вищої освіти

ФІТ, 2 курсу, групи ІСТм-24-1

ЗАЯВА

З правилами чинного Положення про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, згідно з яким виявлення академічного плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів академічної відповідальності, ознайомлений (а). Про використання спеціалізованих програмних засобів (СПЗ) StrikePlagiarism та Anti-Plagiarism для перевірки кваліфікаційних робіт здобувачів вищої освіти на наявність академічного плагіату оповіщений (а). Надаю університету право на передачу моєї роботи для обробки та збереження в базах даних СПЗ і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються СПЗ.

Також надаю свою згоду на обробку й збереження університетом моєї роботи в Інституційному репозитарії Хмельницького національного університету.

Робота надається для перевірки в електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

10 грудня 2025 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Метод та веб-орієнтована інформаційна система бібліотеки.
Автор Олег ВОЛОШИН
Освітня програма Інформаційні системи та технології
Рівень вищої освіти другий (магістерський)
Спеціальність 126 Інформаційні системи та технології
Науковий керівник: к.т.н., доцент Дмитро МЕДЗАТИЙ

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	відповідає
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) усі запозичення фрагментарні, або мають належним чином оформленні посилання;

2) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з джерелами на один фрагмент речення;

3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.

4) значна частина знайденого плагіату відноситься до списку використаних джерел

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 3,4%; та системою Anti-Plagiarism складає 0%, що, з урахуванням наведених обгрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

15.12.2025

Завідувач кафедри _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

Гарант освітньої програми _____ Ольга ПАВЛОВА
Підпис Ім'я, ПРІЗВИЩЕ

Керівник кваліфікаційної роботи _____ Дмитро МЕДЗАТИЙ
Підпис Ім'я, ПРІЗВИЩЕ