Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя
Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра програмної інженерії
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Архітектурне проєктування та розробка веб-платформи для управління бібліотечними процесами на основі відкритих програмних рішень

Виконав(ла): студент(ка) 6 курсу, групи СПм-61
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

(підпис) Бесащук М. П. (прізвище та ініціали)

Керівник (підпис) Бревус В. М. (прізвище та ініціали)

Нормоконтроль (підпис) Стоянов Ю. М. (прізвище та ініціали)

Завідувач кафедри (підпис) Петрик М. Р. (прізвище та ініціали)

Рецензент (підпис) (прізвище та ініціали)

Тернопіль
2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра програмної інженерії
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

(підпис) (прізвище та ініціали)

« » 2025 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

студенту Бесащуку Миколі Петровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Архітектурне проєктування та розробка веб-платформи для управління бібліотечними процесами на основі відкритих програмних рішень

Керівник роботи Бревус Віталій Миколайович, доктор філософії, доцент кафедри ПІ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *27* » *листопада* 2025 року № *4/7-1045*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи вимоги до функціонування веб-платформи управління бібліотечними процесами, стандарти бібліографічного опису та формат MARC21, відкриті бібліотечні системи.

4. Зміст роботи (перелік питань, які потрібно розробити)
Провести аналіз предметної області бібліотечних інформаційних систем та основних біліотечних систем, проаналізувати існуючі open-sourse бібліотечні системи та їх архітектурні

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Галина Михайлівна		
Безпека в надзвичайних ситуаціях	Стручок Володимир Сергійович		
Нормоконтроль	Стоянов Юрій Миколайович		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної області та вимог		
2	Огляд існуючих рішень та стандартів		
3	Проєктування архітектури системи		
4	Реалізація серверної та клієнтської частин		
5	Інтеграція із зовнішніми сервісами		
6	Тестування програмного забезпечення		
7	Оформлення пояснювальної записки		
8	Підготовка презентації та доповіді		
9	Попередній захист		
10	Нормоконтроль, рецензування		
11	Занесення диплома в електронний архів		
12	Допуск до захисту у зав. кафедри		

Студент _____ Бесащук М. П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Бревус В. М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота магістра. Тернопільський національний технічний університет імені Івана Пулюя, кафедра програмної інженерії, спеціальність 121 «Інженерія програмного забезпечення». ТНТУ, 2025, Сторінок -- , рисунків --, таблиць --, презентація.

Метою магістерської роботи є архітектурне проєктування та розробка веб-платформи для управління бібліотечними процесами на основі відкритих програмних рішень з підтримкою інтеграції з MARC-каталогами, DOI-сервісами та Google Books API.

Об’єктом дослідження є процеси автоматизації управління бібліотечними ресурсами.

Предметом дослідження є методи та архітектурні рішення проєктування веб-платформ управління бібліотечними процесами на основі відкритих програмних технологій.

Наукова новизна роботи полягає в адаптації архітектурних підходів open-source бібліотечних систем до сучасної модульної веб-архітектури з розширеними інтеграційними можливостями.

У магістерській роботі розглянуто архітектурне проєктування та розробку веб-платформи для управління бібліотечними процесами на основі відкритих програмних рішень. Проведено аналіз предметної області, досліджено існуючі open-source бібліотечні системи та стандарти бібліографічного опису. Спроєктовано модульну архітектуру системи, реалізовано серверну та клієнтську частини, а також інтеграцію з MARC-каталогами, DOI та Google Books API. Проведено тестування та оцінку якості програмного продукту.

Ключові слова: бібліотечна інформаційна система, веб-платформа, open-source, MARC21, DOI, Google Books API, архітектура програмного забезпечення.

ABSTRACT

Master’s Thesis. Ivan Puluj Ternopil National Technical University, Department of Software Engineering, Specialty 121 “Software Engineering”. TNTU, 2025, Pages --, Figures --, Tables --, Presentation.

The aim of the master’s thesis is the architectural design and development of a web platform for managing library processes based on open-source software solutions with support for integration with MARC catalogs, DOI services, and the Google Books API.

The object of research is the processes of automating library resource management.

The subject of research is the methods and architectural solutions for designing web platforms for managing library processes based on open-source software technologies.

The scientific novelty of the work lies in adapting architectural approaches of open-source library systems to modern modular web architecture with extended integration capabilities.

The master’s thesis addresses the architectural design and development of a web platform for managing library processes based on open-source software solutions. The domain was analyzed, existing open-source library systems and bibliographic description standards were studied. A modular system architecture was designed, server-side and client-side components were implemented, as well as integration with MARC catalogs, DOI, and the Google Books API. Testing and quality evaluation of the software product were carried out.

Keywords: library information system, web platform, open-source, MARC21, DOI, Google Books API, software architecture.

ЗМІСТ

ВСТУП

Стрімкий розвиток інформаційних технологій та цифровізація суспільства зумовлюють необхідність модернізації інформаційних систем у різних сферах діяльності, зокрема в бібліотечній галузі. Сучасні бібліотеки поступово трансформуються з традиційних сховищ друкованих видань у багатофункціональні інформаційні центри, які забезпечують доступ до електронних ресурсів, наукових публікацій, цифрових архівів та відкритих інформаційних платформ. У зв'язку з цим зростають вимоги до ефективності, гнучкості та інтеграційних можливостей бібліотечних інформаційних систем.

Використання застарілих або закритих програмних рішень у бібліотеках часто призводить до обмежень у функціональності, складності супроводу та високих витрат на ліцензування. Особливо актуальною є проблема інтеграції таких систем із сучасними інформаційними сервісами, зокрема міжнародними бібліографічними базами, цифровими ідентифікаторами наукових публікацій та відкритими API. У цих умовах зростає роль відкритих програмних рішень (open-source), які дозволяють адаптувати систему до потреб конкретної установи та забезпечують прозорість архітектурних рішень.

Бібліотечні інформаційні системи мають обробляти значні обсяги структурованих бібліографічних даних, підтримувати стандарти обміну інформацією та забезпечувати багатокористувацький доступ. Важливою складовою таких систем є підтримка стандартів бібліографічного опису, зокрема формату MARC21, який широко використовується у світовій бібліотечній практиці. Крім того, сучасні бібліотеки дедалі частіше інтегруються з сервісами цифрової ідентифікації, такими як DOI (Digital Object Identifier), а також з інформаційними платформами типу Google Books, що дозволяє збагачувати бібліографічні записи додатковими метаданими.

Наявні open-source бібліотечні системи, зокрема Koha та Evergreen, забезпечують базову автоматизацію бібліотечних процесів і широко застосовуються в бібліотеках різного масштабу. Водночас аналіз їхньої архітектури

свідчить про наявність обмежень щодо модульності, масштабованості та адаптації до сучасних веб-архітектур. Це створює передумови для розробки нових веб-платформ або адаптації існуючих архітектурних рішень з урахуванням сучасних вимог програмної інженерії.

У контексті спеціальності 121 «Інженерія програмного забезпечення» особливу увагу приділяють питанням архітектурного проєктування, вибору технологічного стеку, модульності програмних систем та якості програмного забезпечення. Розробка веб-платформи управління бібліотечними процесами на основі відкритих програмних рішень є актуальним прикладним завданням, що поєднує теоретичні аспекти програмної інженерії з практичними потребами інформаційної інфраструктури.

З огляду на це актуальним є завдання архітектурного проєктування та розробки веб-платформи, яка забезпечує автоматизацію основних бібліотечних процесів, підтримує стандарти бібліографічного опису, має модульну архітектуру та здатна інтегруватися з зовнішніми бібліографічними сервісами. Така платформа повинна відповідати сучасним вимогам до безпеки, масштабованості та зручності використання.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

- проаналізувати предметну область бібліотечних інформаційних систем;
- дослідити існуючі open-source рішення управління бібліотечними процесами;
- обґрунтувати вибір архітектурного стилю та технологічного стеку;
- спроєктувати модульну архітектуру веб-платформи;
- розробити серверну та клієнтську частини системи;
- реалізувати інтеграцію з MARC-каталогами, DOI та Google Books API;
- провести тестування та оцінку якості програмного забезпечення;
- обґрунтувати можливості впровадження та подальшого розвитку системи.

Об’єктом дослідження є процеси автоматизації управління бібліотечними

ресурсами.

Предметом дослідження є методи та архітектурні рішення проєктування веб-платформ управління бібліотечними процесами на основі відкритих програмних технологій.

У процесі виконання магістерської роботи використовувалися методи системного аналізу, об’єктно-орієнтованого проєктування, моделювання за допомогою UML, методи програмної інженерії, а також методи тестування програмного забезпечення.

Практичне значення полягає в можливості використання розробленої веб-платформи в бібліотеках закладів освіти та інших інформаційних установах, а також у навчальному процесі для демонстрації сучасних підходів до проєктування програмних систем.

1 АНАЛІЗ ВИМОГ ТА ПРЕДМЕТНОЇ ОБЛАСТІ БІБЛІОТЕЧНИХ ІНФОРМАЦІЙНИХ СИСТЕМ

У даному розділі проведено аналіз предметної області бібліотечних інформаційних систем, розглянуто основні бібліотечні процеси, стандарти бібліографічного опису, а також здійснено огляд і порівняльний аналіз існуючих open-source рішень для автоматизації бібліотек. На основі проведеного аналізу сформовано вимоги до розроблюваної веб-платформи [1].

1.1 Аналіз бібліотечних процесів та інформаційних потоків

Бібліотечна діяльність у сучасних умовах є складним багаторівневим процесом, що поєднує традиційні методи роботи з інформаційними ресурсами та сучасні цифрові технології. Бібліотека виконує роль інформаційного посередника між джерелами знань і кінцевими користувачами, забезпечуючи доступ до друкованих та електронних ресурсів, їх систематизацію та збереження.

У структурі бібліотечної діяльності виділяють низку взаємопов’язаних процесів, які формують єдину інформаційну систему. До основних учасників бібліотечних процесів належать читачі, бібліотекарі та адміністратори інформаційної системи. Кожна з цих категорій користувачів взаємодіє з бібліотечною системою відповідно до своїх функціональних обов’язків та інформаційних потреб [2].

Одним із ключових процесів є формування бібліотечного фонду, що включає відбір, придбання, облік та реєстрацію документів. У сучасних бібліотеках цей процес дедалі частіше автоматизується, що дозволяє скоротити часові витрати та зменшити ймовірність помилок. Інформаційні потоки на цьому етапі охоплюють дані про постачальників, бібліографічні описи ресурсів та фінансову інформацію.

Наступним важливим етапом є каталогізація, яка передбачає створення бібліографічного опису документа та його включення до електронного каталогу. Каталогізація є основою ефективного інформаційного пошуку, оскільки саме від

повноти та коректності бібліографічних даних залежить якість доступу користувачів до ресурсів. У процесі каталогізації активно використовуються стандарти бібліографічного опису, зокрема формат MARC21[3].

Обслуговування користувачів є центральним процесом бібліотечної діяльності та охоплює операції з видачі, повернення, резервування та продовження користування бібліотечними ресурсами. У сучасних інформаційних системах ці операції реалізуються за допомогою автоматизованих механізмів, що дозволяють відстежувати рух примірників у режимі реального часу та забезпечувати актуальність даних.

Важливу роль відіграють процеси інформаційного пошуку, які забезпечують доступ користувачів до бібліографічних ресурсів. Сучасні бібліотечні системи повинні підтримувати різні види пошуку, зокрема простий, розширений та тематичний. Для цього необхідно забезпечити ефективну організацію інформаційних потоків між базами даних, серверною частиною системи та клієнтськими веб-інтерфейсами.

Окрему групу складають аналітичні та управлінські процеси, пов’язані з формуванням статистичної звітності, аналізом використання бібліотечних ресурсів та плануванням розвитку бібліотечного фонду. Інформаційні потоки в межах цих процесів спрямовані від операційних модулів до аналітичних підсистем, де здійснюється обробка та узагальнення даних.

З точки зору інформаційної взаємодії бібліотечні процеси характеризуються високою інтенсивністю обміну даними між внутрішніми та зовнішніми компонентами системи. Зовнішні інформаційні потоки можуть включати дані з національних бібліотечних каталогів, видавничих платформ, наукових репозитаріїв та відкритих API. Внутрішні потоки забезпечують узгоджену роботу модулів бібліотечної інформаційної системи.

Автоматизація бібліотечних процесів потребує чіткого моделювання інформаційних потоків, що дозволяє оптимізувати взаємодію між компонентами системи та підвищити її продуктивність. Недостатньо продумана організація потоків даних може призвести до дублювання інформації, зниження швидкодії

системи та ускладнення її супроводу.

Таким чином, аналіз бібліотечних процесів та інформаційних потоків дозволяє визначити ключові вимоги до архітектури програмної системи та обґрунтувати необхідність використання модульного підходу до проєктування веб-платформи управління бібліотечними процесами.

1.2 Стандарти бібліографічного опису та формат MARC21

Ефективне функціонування бібліотечних інформаційних систем неможливе без використання стандартизованих підходів до бібліографічного опису документів. Стандарти бібліографічного опису забезпечують єдність представлення інформації, коректність пошуку та можливість обміну даними між різними бібліотечними та інформаційними системами. У контексті цифровізації бібліотечної діяльності роль таких стандартів суттєво зростає, оскільки вони виступають основою інтеграції з національними та міжнародними інформаційними ресурсами.

Бібліографічний опис являє собою структурований набір метаданих, що характеризують інформаційний ресурс і дозволяють його ідентифікувати, класифікувати та знаходити серед інших документів. До основних елементів бібліографічного опису належать відомості про автора, назву, вихідні дані, тематику, ідентифікатори та інші атрибути, необхідні для повноцінного представлення документа в електронному каталозі.

У бібліотечній практиці застосовуються різні стандарти бібліографічного опису, зокрема ISBD, AACR2, RDA, проте саме формат MARC набув найбільшого поширення у зв’язку з можливістю його використання в автоматизованих системах. Формат MARC був розроблений з метою забезпечення машинозчитуваного подання бібліографічної інформації та подальшої автоматизованої обробки даних.

Стандарт MARC21 є сучасною версією формату MARC та використовується у більшості академічних і публічних бібліотек світу. Він виник у результаті уніфікації форматів USMARC та CANMARC і став міжнародним стандартом де-

факто для обміну бібліографічними даними. Використання MARC21 дозволяє бібліотекам інтегруватися у глобальні бібліотечні мережі та спільно використовувати каталоги [4].

MARC21-запис має чітко визначену ієрархічну структуру, що включає лідер запису, директорію та набір змінних полів. Лідер містить службову інформацію про тип запису та спосіб його обробки, директорія забезпечує швидкий доступ до полів, а змінні поля містять безпосередні бібліографічні дані. Така структура забезпечує ефективну машинну обробку записів та високу продуктивність бібліотечних інформаційних систем.

Особливістю формату MARC21 є використання числових ідентифікаторів полів та підполів, що дозволяє гнучко описувати різні типи ресурсів. Наприклад, окремі поля призначені для опису авторства, назви, серійності, предметних рубрик та класифікаційних індексів. Це дозволяє формувати багатовимірний пошук і забезпечувати точність результатів у процесі інформаційного пошуку.

Використання MARC21 у веб-орієнтованих бібліотечних системах потребує адаптації формату до сучасних архітектурних підходів. Традиційно MARC-записи зберігаються у спеціалізованих форматах, однак сучасні веб-платформи часто використовують JSON або XML для обміну даними. У зв’язку з цим актуальним є завдання трансформації MARC21-записів у веб-сумісні формати без втрати семантики даних.

Окремої уваги заслуговує інтеграція MARC21 з зовнішніми ідентифікаційними системами, такими як DOI (Digital Object Identifier). Використання DOI дозволяє однозначно ідентифікувати наукові публікації та забезпечує стабільний доступ до цифрових ресурсів. Включення DOI до бібліографічного опису значно підвищує якість метаданих та розширює можливості міжсистемної інтеграції.

Крім того, сучасні бібліотечні системи дедалі частіше інтегруються з відкритими інформаційними сервісами, зокрема Google Books API, що дозволяє автоматично отримувати додаткові відомості про видання, обкладинки, анотації та фрагменти текстів. Інтеграція таких сервісів із MARC-каталогами сприяє

збагаченню бібліографічних записів та підвищенню привабливості електронних каталогів для користувачів.

Використання стандартів бібліографічного опису також має важливе значення з точки зору забезпечення довготривалого зберігання та міграції даних. Стандартизовані формати дозволяють уникнути залежності від конкретних програмних продуктів і полегшують перенесення бібліотечних каталогів між різними інформаційними системами.

Таким чином, стандарти бібліографічного опису та формат MARC21 є фундаментальною складовою сучасних бібліотечних інформаційних систем. Їх використання забезпечує сумісність, масштабованість та інтеграційні можливості веб-платформ управління бібліотечними процесами, що є особливо важливим у контексті розробки систем на основі відкритих програмних рішень.

1.3 Аналіз існуючих open-source бібліотечних систем

Активний розвиток відкритого програмного забезпечення сприяв появі значної кількості бібліотечних інформаційних систем, що розповсюджуються за open-source ліцензіями. Використання таких систем дозволяє бібліотекам зменшити витрати на придбання програмного забезпечення, забезпечити прозорість архітектурних рішень та адаптувати функціональність під власні потреби. У цьому підрозділі розглянуто найбільш поширені open-source бібліотечні системи та проаналізовано їхні архітектурні та функціональні особливості [5].

Open-source бібліотечні системи, як правило, орієнтовані на підтримку повного циклу бібліотечних процесів, починаючи від формування фонду та каталогізації і завершуючи обслуговуванням користувачів і формуванням звітності. Важливою перевагою таких рішень є можливість їхнього подальшого розвитку силами спільноти або власних розробників бібліотеки [6].

1.3.1 Загальна характеристика open-source бібліотечних систем

Сучасні open-source бібліотечні системи розробляються з урахуванням вимог до веб-орієнтованості, багатокористувацького доступу та підтримки стандартів бібліографічного опису. Вони зазвичай включають модулі каталогізації, управління примірниками, роботи з користувачами, обслуговування читачів та аналітики.

Архітектурно більшість таких систем реалізовані як серверні веб-застосунки з централізованою базою даних. Взаємодія з користувачами здійснюється через веб-інтерфейс, що дозволяє забезпечити доступ до системи з різних пристроїв без необхідності встановлення спеціалізованого клієнтського програмного забезпечення.

Важливою характеристикою open-source бібліотечних систем є підтримка інтеграції з зовнішніми сервісами та інформаційними ресурсами. Це досягається шляхом використання програмних інтерфейсів прикладного програмування, стандартних протоколів обміну даними та форматів представлення інформації.

1.3.2 Система Koha

Koha є однією з перших open-source інтегрованих бібліотечних систем. Вона реалізує повний цикл бібліотечних процесів, включаючи каталогізацію, облік примірників, обслуговування читачів та формування звітності.

Основні характеристики системи Koha:

- веб-орієнтований інтерфейс;
- підтримка стандарту MARC21;
- модульна структура функціональних підсистем;
- можливість інтеграції через API.

Разом з тим, архітектура Koha значною мірою базується на монолітному підході, що ускладнює масштабування системи та впровадження сучасних архітектурних патернів.

1.3.3 Система Evergreen

Evergreen є масштабованою бібліотечною системою, орієнтованою на використання в мережах публічних бібліотек. Вона підтримує роботу з великими обсягами даних та розподілену архітектуру.

Основними перевагами Evergreen є:

- підтримка MARC21;
- орієнтація на високі навантаження;
- можливість централізованого управління бібліотечними фондами.

Недоліком системи є складність налаштування та обмежена гнучкість у зміні архітектурних компонентів без значного втручання в ядро системи [7].

1.4 Порівняльний аналіз архітектурних рішень

Розробка сучасних бібліотечних інформаційних систем потребує використання ефективних архітектурних підходів, які забезпечують надійність, масштабованість, розширюваність та простоту інтеграції з зовнішніми сервісами. Архітектура програмної системи визначає принципи організації її компонентів, способи їх взаємодії та можливості подальшого розвитку системи.

У контексті бібліотечних систем архітектурні рішення повинні враховувати специфіку предметної області, зокрема роботу з великими обсягами структурованих бібліографічних даних, підтримку стандартів обміну інформацією та необхідність багатокористувацького доступу. Саме тому порівняльний аналіз архітектурних підходів є важливим етапом підготовки до проєктування нової веб-платформи.

Традиційно бібліотечні інформаційні системи, зокрема Koha та Evergreen, реалізовані з використанням монолітної архітектури, у якій усі основні функціональні компоненти зосереджені в межах одного програмного застосунку. Такий підхід спрощує початкову розробку та впровадження системи, проте з часом призводить до ускладнення супроводу та модернізації.

Монолітна архітектура характеризується тісним зв’язком між модулями, що ускладнює внесення змін без ризику порушення роботи інших компонентів. У бібліотечних системах це проявляється у складності адаптації функціоналу під нові стандарти, розширення можливостей інтеграції або масштабування системи при зростанні кількості користувачів і обсягу даних [7].

На відміну від монолітного підходу, модульна архітектура передбачає поділ системи на окремі функціональні модулі, кожен з яких відповідає за певний набір завдань. Такий підхід дозволяє ізолювати зміни в межах окремих компонентів, спрощує тестування та підвищує гнучкість системи. Для бібліотечних веб-платформ модульна архітектура є доцільною, оскільки бібліотечні процеси мають чітку логічну декомпозицію.

Подальшим розвитком модульного підходу є мікросервісна архітектура, у якій кожен сервіс функціонує як незалежний програмний компонент із власною базою даних та API. Застосування мікросервісів дозволяє масштабувати окремі частини системи незалежно одна від одної та спрощує інтеграцію з зовнішніми сервісами, такими як MARC-каталоги, DOI-реєстри та Google Books API.

Разом з тим, мікросервісна архітектура має й недоліки, зокрема підвищену складність розгортання, необхідність використання додаткових інструментів оркестрації та моніторингу, а також зростання вимог до кваліфікації розробників. Тому при проєктуванні бібліотечної веб-платформи доцільно обирати компромісний варіант між складністю реалізації та отриманими перевагами.

Важливим аспектом архітектурного проєктування є вибір способу взаємодії між компонентами системи. Сучасні бібліотечні платформи дедалі частіше використовують REST API, що забезпечує стандартизований обмін даними між клієнтською та серверною частинами, а також з зовнішніми сервісами. Використання API підвищує відкритість системи та спрощує інтеграцію з іншими програмними продуктами [9].

Порівняльний аналіз архітектурних підходів показує, що існуючі open-source бібліотечні системи мають обмежені можливості з точки зору адаптації до сучасних веб-архітектур. Це зумовлює необхідність розробки нової веб-платформи,

архітектура якої буде орієнтована на модульність, розширюваність та інтеграційність.

Таким чином, результати порівняльного аналізу підтверджують доцільність використання модульної або комбінованої архітектури при розробці веб-платформи управління бібліотечними процесами на основі відкритих програмних рішень.

1.5 Формування вимог до веб-платформи

На основі проведеного аналізу сформовано такі вимоги до розроблюваної веб-платформи.

Функціональні вимоги:

- ведення електронного бібліотечного каталогу;
- підтримка формату MARC21;
- управління користувачами та ролями доступу;
- інтеграція з DOI-сервісами;
- інтеграція з Google Books API;
- пошук і фільтрація бібліографічних записів.

Нефункціональні вимоги:

- модульна архітектура;
- використання open-source технологій;
- масштабованість;
- безпека даних;
- підтримка REST API.

1.6 Висновки до першого розділу

У першому розділі було проаналізовано предметну область бібліотечних інформаційних систем, розглянуто основні бібліотечні процеси та стандарти бібліографічного опису. Проведено огляд існуючих open-source рішень Koha та Evergreen, визначено їхні переваги та недоліки. На основі отриманих результатів сформовано вимоги до веб-платформи управління бібліотечними процесами, що створює підґрунтя для подальшого архітектурного проєктування та розробки системи [11].

2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-ПЛАТФОРМИ

У другому розділі магістерської роботи було виконано проєктування та розробку веб-платформи для управління бібліотечними процесами на основі відкритих програмних рішень. На початковому етапі обґрунтовано вибір архітектурного стилю та технологічного стеку з урахуванням вимог модульності, масштабованості та можливості інтеграції із зовнішніми інформаційними сервісами. Обрані архітектурні підходи відповідають сучасним тенденціям розробки веб-орієнтованих інформаційних систем [10].

У процесі проєктування модульної архітектури систему було декомпозовано на окремі функціональні компоненти, що забезпечують реалізацію основних бібліотечних процесів, зокрема управління користувачами, каталогізацію, пошук, облік примірників та аналітику. Такий підхід дозволяє зменшити зв’язність між компонентами системи та спрощує її подальший супровід і розширення.

Значну увагу приділено проєктуванню бази даних та моделей предметної області, які відображають логіку бібліотечних процесів і забезпечують збереження структурованих бібліографічних даних. Запропонована структура бази даних підтримує роботу зі стандартом MARC21 та дозволяє зберігати зовнішні ідентифікатори, що є необхідним для інтеграції з міжнародними бібліографічними сервісами.

У розділі також описано реалізацію серверної частини веб-платформи, яка забезпечує обробку бізнес-логіки, взаємодію з базою даних і зовнішніми API, а також реалізацію клієнтської частини, орієнтованої на зручний та інтуїтивно зрозумілий користувацький інтерфейс. Взаємодія між клієнтом і сервером реалізована за допомогою REST API, що підвищує гнучкість та незалежність компонентів системи.

Окремо розглянуто питання інтеграції з MARC-каталогами, що забезпечує сумісність розробленої платформи з існуючими бібліотечними системами, а також інтеграції з DOI-сервісами та Google Books API, які дозволяють збагачувати

бібліографічні записи додатковими метаданими та покращувати якість електронного каталогу.

Таким чином, у другому розділі було сформовано повноцінну архітектурну та технологічну основу веб-платформи управління бібліотечними процесами. Отримані результати створюють передумови для проведення тестування, оцінки якості та впровадження розробленої системи, що буде розглянуто у наступному розділі магістерської роботи [12].

2.1 Вибір архітектурного стилю та технологічного стеку

Вибір архітектурного стилю та технологічного стеку є одним із визначальних етапів проєктування веб-платформи управління бібліотечними процесами, оскільки саме на цьому етапі закладаються основні принципи побудови програмної системи, її масштабованості, розширюваності та здатності до інтеграції з зовнішніми інформаційними ресурсами. Архітектурні рішення безпосередньо впливають на якість програмного забезпечення, складність його супроводу та можливості подальшого розвитку.

У результаті аналізу предметної області та існуючих open-source бібліотечних систем, проведеного в першому розділі, було встановлено, що традиційні монолітні архітектури, які використовуються в таких системах, як Koha та Evergreen, мають низку обмежень. Зокрема, вони ускладнюють масштабування, інтеграцію з сучасними веб-сервісами та адаптацію функціональності до специфічних потреб конкретної бібліотеки. Це зумовлює необхідність використання більш гнучкого архітектурного підходу при розробці нової веб-платформи.

З урахуванням зазначених вимог для розроблюваної системи було обрано модульну сервісно-орієнтовану архітектуру, яка поєднує переваги класичної багаторівневої архітектури та сучасних REST-орієнтованих підходів. Такий архітектурний стиль передбачає логічний поділ системи на незалежні

функціональні модулі, кожен з яких відповідає за реалізацію окремого набору бізнес-функцій бібліотечної діяльності.

Модульний підхід дозволяє мінімізувати зв’язність між компонентами системи та забезпечити чітке розмежування відповідальності. Це є особливо важливим для бібліотечних веб-платформ, які повинні підтримувати широкий спектр функцій, починаючи від каталогізації та обліку примірників і завершуючи інтеграцією з зовнішніми бібліографічними сервісами. У разі необхідності окремі модулі можуть бути доопрацьовані або замінені без істотного впливу на роботу всієї системи.

Важливим елементом обраного архітектурного стилю є використання REST API як основного механізму взаємодії між клієнтською та серверною частинами системи, а також між внутрішніми модулями та зовнішніми сервісами. REST-підхід забезпечує стандартизований обмін даними на основі HTTP-протоколу та дозволяє використовувати універсальні формати представлення інформації, такі як JSON. Це значно спрощує інтеграцію з MARC-каталогами, DOI-сервісами та Google Books API.

При виборі технологічного стеку ключовим критерієм була відповідність принципам open-source, що дозволяє знизити вартість впровадження системи, забезпечити прозорість архітектурних рішень та можливість подальшого розвитку платформи без прив’язки до комерційних постачальників програмного забезпечення. Крім того, враховувалась наявність активної спільноти розробників та широкої підтримки обраних технологій [13].

Для реалізації серверної частини веб-платформи доцільним є використання сучасного фреймворку для розробки веб-застосунків, який підтримує побудову REST API, роботу з базами даних та реалізацію механізмів автентифікації й авторизації. Такий фреймворк забезпечує структурованість коду, спрощує тестування та підвищує надійність програмного забезпечення.

Як система управління базами даних обрано реляційну СУБД, оскільки бібліографічні дані мають чітку структуру та складні зв’язки між сутностями. Реляційний підхід дозволяє забезпечити цілісність даних, підтримку транзакцій та

ефективну реалізацію складних запитів, що є важливим для бібліотечних інформаційних систем. Водночас архітектура платформи передбачає можливість розширення з використанням інших типів сховищ даних у разі потреби.

Для клієнтської частини веб-платформи обрано підхід побудови односторінкового веб-застосунку, що забезпечує динамічну взаємодію з користувачем та зменшує навантаження на сервер. Такий підхід дозволяє реалізувати зручний та інтуїтивно зрозумілий інтерфейс користувача, що є важливим фактором для систем, орієнтованих на широкий спектр користувачів із різним рівнем підготовки [14].

Обраний технологічний стек забезпечує відповідність сучасним вимогам програмної інженерії, зокрема принципам розширюваності, повторного використання коду та незалежності компонентів. Крім того, він дозволяє реалізувати гнучку систему інтеграції з зовнішніми сервісами, що є критично важливим для бібліотечних веб-платформ у контексті цифровізації бібліотечної діяльності.

Таким чином, вибір модульної сервісно-орієнтованої архітектури та open-source технологічного стеку є обґрунтованим та доцільним рішенням для розробки веб-платформи управління бібліотечними процесами. Запропонований підхід забезпечує баланс між функціональністю, гнучкістю та складністю реалізації, створюючи надійну основу для подальшого проєктування та розробки системи.

2.2 Проєктування модульної архітектури системи

Проєктування модульної архітектури є одним з ключових етапів розробки веб-платформи управління бібліотечними процесами, оскільки саме архітектура визначає структуру системи, принципи взаємодії між її компонентами та можливості подальшого розвитку. У сучасній програмній інженерії модульний підхід розглядається як ефективний спосіб зменшення складності програмних систем шляхом їх логічної декомпозиції.

Модульна архітектура передбачає поділ системи на відносно незалежні функціональні блоки (модулі), кожен з яких реалізує чітко визначений набір функцій. Такий підхід дозволяє знизити зв’язність між компонентами, підвищити повторне використання коду та спростити тестування і супровід програмного забезпечення. Для бібліотечних інформаційних систем модульна архітектура є особливо доцільною, оскільки бібліотечні процеси мають чітку функціональну структуру.

При проєктуванні архітектури веб-платформи було враховано результати аналізу існуючих open-source рішень, зокрема систем Koha та Evergreen. Було встановлено, що монолітний підхід, характерний для цих систем, ускладнює впровадження нових функцій і інтеграцію з сучасними веб-сервісами. Тому у даній роботі обрано модульну архітектуру як компроміс між простотою реалізації та гнучкістю системи.

У межах платформи виокремлено такі основні модулі:

- Модуль управління користувачами та ролями, який забезпечує реєстрацію, автентифікацію та авторизацію користувачів. У цьому модулі реалізується рольова модель доступу, що дозволяє розмежувати повноваження між читачами, бібліотекарями та адміністраторами системи.
- Модуль бібліографічного каталогу, призначений для зберігання, редагування та перегляду бібліографічних записів. Даний модуль забезпечує підтримку формату MARC21 та взаємодіє з підсистемами імпорту й експорту бібліографічних даних.
- Модуль обліку примірників, який відповідає за управління фізичними та електронними примірниками ресурсів, їх статусами та місцезнаходженням.
- Модуль пошуку та навігації, що реалізує механізми простого та розширеного пошуку бібліографічних записів, фільтрацію результатів і сортування даних.
- Модуль інтеграції з зовнішніми сервісами, який забезпечує взаємодію з MARC-каталогами, DOI-реєстрами та Google Books API.

- Модуль аналітики та звітності, призначений для формування статистичних звітів та аналізу використання бібліотечних ресурсів.

Кожен з перелічених модулів має чітко визначену зону відповідальності та взаємодіє з іншими компонентами системи через стандартизовані програмні інтерфейси. Такий підхід дозволяє мінімізувати залежності між модулями та забезпечити можливість їх незалежного розвитку [15].

Взаємодія між модулями реалізується на основі REST API, що відповідає сучасним підходам до побудови веб-орієнтованих інформаційних систем. Використання REST дозволяє уніфікувати обмін даними та спростити інтеграцію клієнтської частини з серверною логікою. Крім того, REST API створює передумови для подальшого розширення платформи шляхом підключення сторонніх застосунків.

Особливу увагу при проєктуванні модульної архітектури приділено модулю інтеграції, оскільки він забезпечує взаємодію з зовнішніми джерелами бібліографічної інформації. Винесення інтеграційної логіки в окремий модуль дозволяє ізолювати зміни, пов’язані з оновленням API або форматів даних, та зменшити вплив таких змін на інші компоненти системи.

Модульна архітектура також сприяє підвищенню масштабованості системи. За потреби окремі модулі можуть бути розгорнуті на різних серверних вузлах або оптимізовані для обробки підвищених навантажень. Це є особливо важливим для бібліотечних платформ, що обслуговують значну кількість користувачів.

Таким чином, спроєктована модульна архітектура веб-платформи управління бібліотечними процесами забезпечує гнучкість, розширюваність та відповідність сучасним вимогам до програмних систем. Запропоновані архітектурні рішення створюють надійну основу для реалізації серверної та клієнтської частин платформи, що буде розглянуто в наступних підпунктах.

2.3 Моделювання архітектури та функціональності системи засобами UML

На етапі проєктування веб-платформи управління бібліотечними процесами важливу роль відіграє формалізоване моделювання архітектури та поведінки системи. Для цього використано уніфіковану мову моделювання UML (Unified Modeling Language), яка є загальноприйнятим стандартом у програмній інженерії та широко застосовується при розробці складних інформаційних систем.

UML дозволяє описати як структурні, так і поведінкові аспекти програмної системи, що забезпечує єдине розуміння архітектурних рішень між розробниками, аналітиками та замовниками. Використання UML є доцільним на різних етапах життєвого циклу програмного забезпечення — від аналізу вимог до реалізації та супроводу [16].

У межах даної магістерської роботи UML застосовується для:

- опису взаємодії користувачів із системою;
- моделювання структури основних програмних компонентів;
- відображення послідовності виконання ключових сценаріїв.

2.3.1 Діаграма прецедентів (Use Case Diagram)

Діаграма прецедентів використовується для відображення функціональних можливостей веб-платформи та взаємодії між системою і зовнішніми акторами. Вона дозволяє визначити межі системи та перелік основних сценаріїв її використання [17].

Основними акторами веб-платформи є:

- Користувач — здійснює пошук та перегляд бібліографічних ресурсів;
- Авторизований користувач — має додаткові можливості роботи з електронним каталогом;
- Адміністратор — відповідає за управління бібліотечними даними, користувачами та системними налаштуваннями.

Діаграма прецедентів відображає такі основні варіанти використання:

- пошук бібліографічних записів;
- перегляд інформації про ресурси;
- керування каталогом;
- адміністрування системи.

Діаграму прецедентів для веб-платформи представлено на рисунку 2.1.

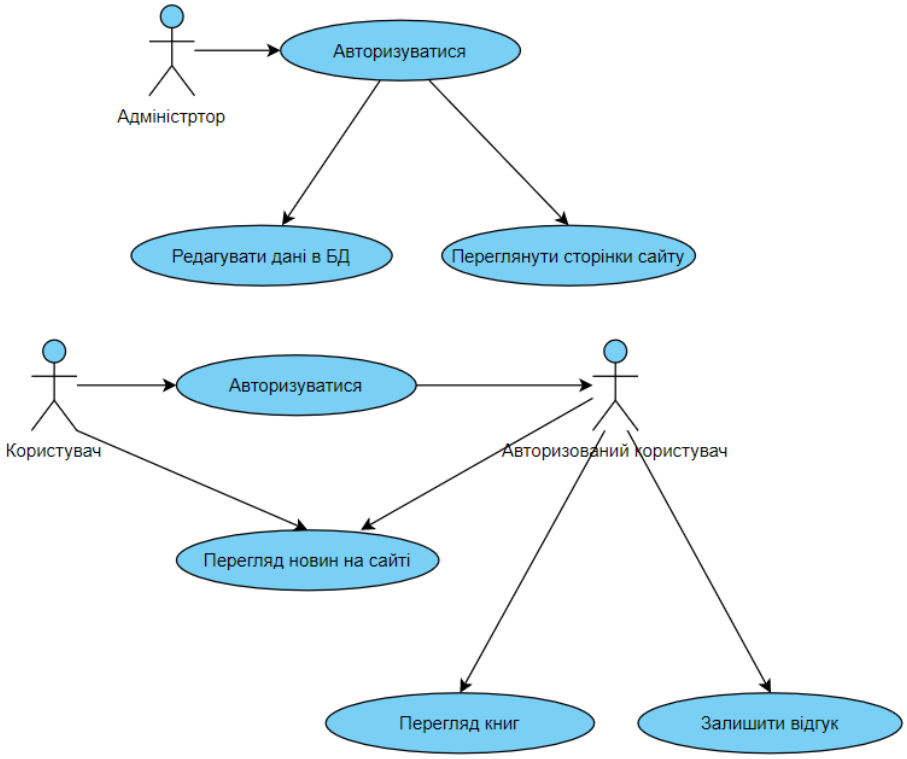

Рисунок 2.1 – Діаграма прецедентів веб-платформи управління бібліотечними процесами

2.3.2 Діаграма класів (Class Diagram)

Діаграма класів використовується для опису статичної структури веб-платформи та взаємозв’язків між основними сутностями предметної області. Вона відображає класи, їх атрибути, методи та типи зв’язків між ними.

Основними класами системи є:

- BibliographicRecord;
- Copy;
- Author;
- User;
- Role;
- SearchService;
- IntegrationService.

Діаграма класів дозволяє формалізувати модель предметної області та слугує основою для реалізації серверної логіки та проєктування бази даних.

Діаграма класів для веб-платформи управління бібліотечними процесами на рисунку 2.2.

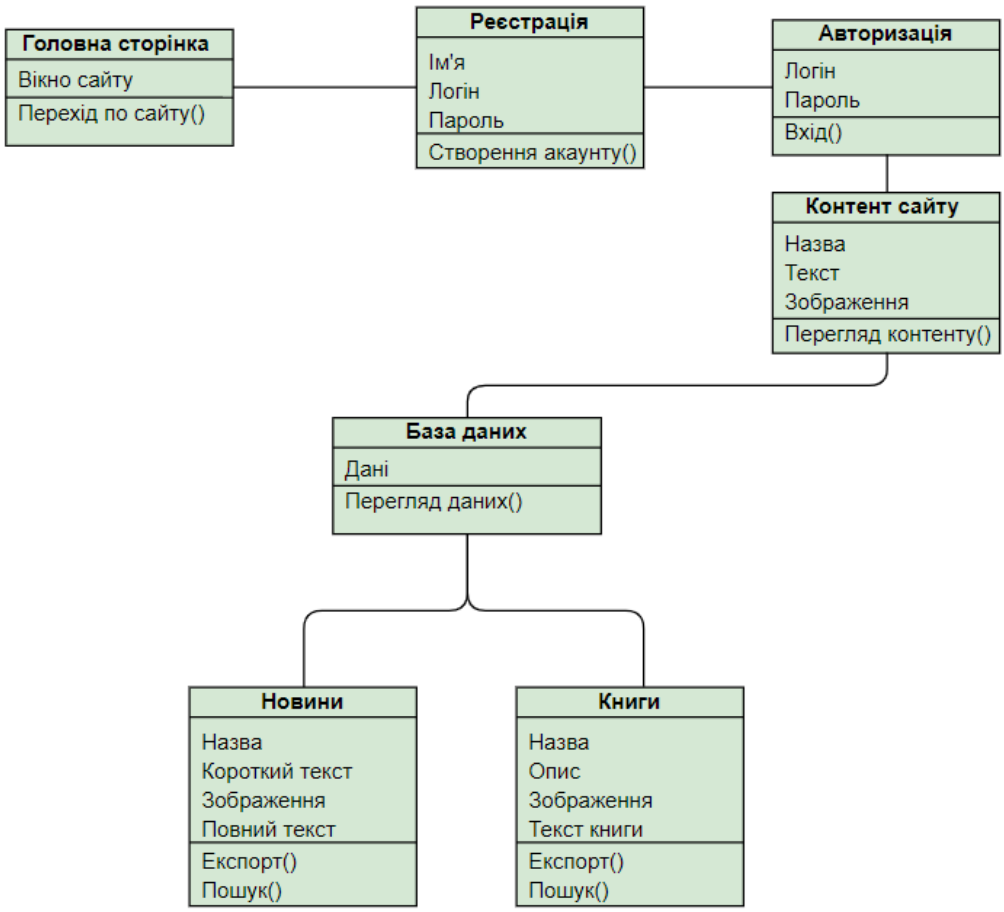

Рисунок 2.2 – Діаграма класів веб-платформи управління бібліотечними процесами

2.3.3 Діаграма послідовності (Sequence Diagram)

Діаграма послідовності використовується для моделювання динамічної взаємодії між об’єктами системи у часі. Вона дозволяє наочно відобразити порядок виклику методів та обмін повідомленнями між компонентами системи.

У даній роботі діаграма послідовності демонструє сценарій пошуку бібліографічного ресурсу, який включає взаємодію користувача, клієнтської частини, серверного API та бази даних [18].

Процес пошуку передбачає:

- введення пошукового запиту користувачем;
- передачу запиту на сервер;
- обробку даних у базі;
- повернення результатів пошуку клієнтській частині.

Діаграма послідовностей для сценарію пошуку бібліографічних ресурсів зображена на рисунку 2.3.

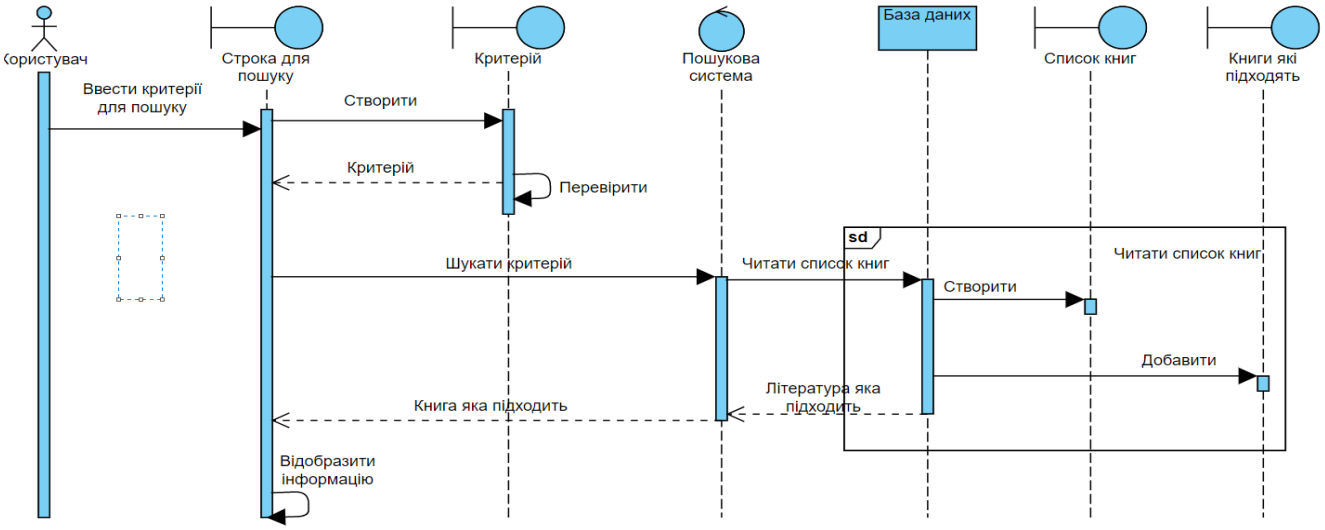

Рисунок 2.3 – Діаграма послідовності сценарію пошуку бібліографічних ресурсів

2.3.4 Архітектурні діаграми веб-платформи

Архітектурна діаграма є одним із ключових артефактів архітектурного проєктування програмного забезпечення, оскільки вона наочно відображає структуру системи, основні компоненти та взаємозв’язки між ними. Архітектурні діаграми використовуються для документування архітектурних рішень і слугують засобом комунікації між розробниками, аналітиками та іншими зацікавленими сторонами.

У межах даної магістерської роботи архітектура веб-платформи управління бібліотечними процесами представлена у вигляді компонентної UML-діаграми, яка демонструє логічну організацію системи, її модульну структуру та інтеграційні зв’язки з зовнішніми сервісами.

Загальну архітектуру розроблюваної веб-платформи, що відображає взаємодію користувача, клієнтської частини, серверного API та бази даних, подано на рисунку 2.4.

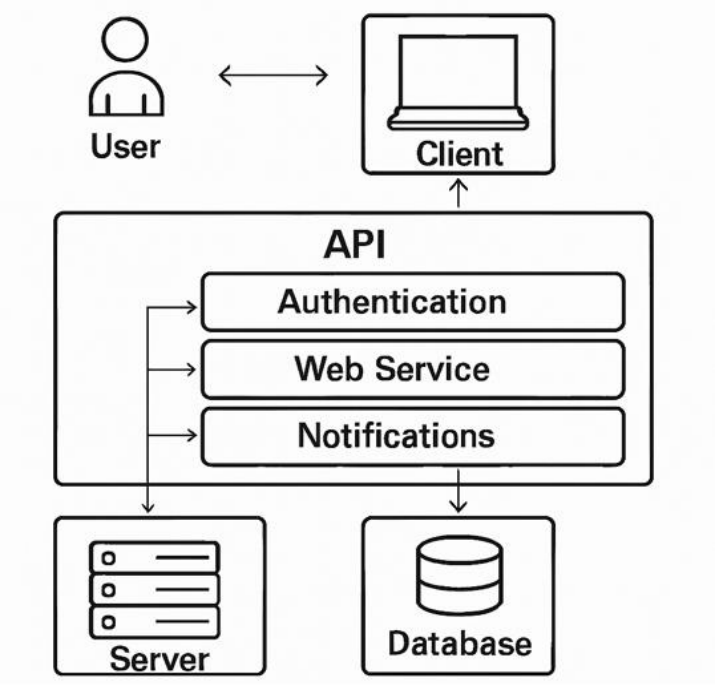

Рисунок 2.4 — Архітектура веб-платформи управління бібліотечними процесами

2.4 Проєктування бази даних та моделей предметної області

Проєктування бази даних є одним із ключових етапів розробки веб-платформи управління бібліотечними процесами, оскільки саме база даних забезпечує збереження, цілісність та доступність інформації, що використовується всіма компонентами системи. Якість проєктування структури даних безпосередньо впливає на продуктивність системи, коректність обробки бібліотечних процесів та можливості подальшого розширення функціональності.

При проєктуванні бази даних було враховано результати аналізу предметної області, проведеного у першому розділі, а також вимоги до підтримки стандартів бібліографічного опису та інтеграції з зовнішніми інформаційними сервісами. Предметна область бібліотечної системи характеризується наявністю складних зв’язків між бібліографічними записами, примірниками, користувачами та операціями обслуговування, що зумовлює доцільність використання реляційної моделі даних.

Основою моделі даних є бібліографічний запис, який містить структуровану інформацію про документ, включаючи відомості про авторів, назву, рік видання, видавця, тематичну класифікацію та ідентифікатори. Для забезпечення сумісності з форматами MARC21 передбачено збереження як нормалізованих атрибутів, так і додаткових полів, що дозволяють відображати специфічні бібліографічні елементи.

Окрему роль у моделі предметної області відіграють примірники, які відображають фізичну або електронну наявність документа в бібліотечному фонді. Один бібліографічний запис може відповідати кільком примірникам, кожен з яких має власний інвентарний номер, статус доступності та історію використання. Такий підхід дозволяє коректно реалізувати процеси обліку та обслуговування читачів.

Модель користувачів системи включає інформацію про читачів, бібліотекарів та адміністраторів, а також їхні ролі та права доступу. Це забезпечує реалізацію механізмів авторизації та контролю доступу до функціональних можливостей веб-платформи. Взаємодія користувачів з бібліотечними ресурсами

фіксується у вигляді операцій, що дозволяє відстежувати рух примірників та формувати аналітичну звітність [19].

Особливу увагу при проєктуванні бази даних було приділено підтримці інтеграції з зовнішніми сервісами. Для цього в структурі даних передбачено збереження зовнішніх ідентифікаторів, зокрема DOI, а також метаданих, отриманих з Google Books API. Це дозволяє збагачувати бібліографічні записи додатковою інформацією без порушення цілісності основної моделі даних.

З метою підвищення продуктивності та забезпечення цілісності інформації було застосовано принципи нормалізації даних, що дозволяє мінімізувати дублювання інформації та спростити підтримку актуальності записів. Водночас у проєкті передбачено можливість оптимізації запитів шляхом використання індексів та агрегованих представлень, що є важливим для реалізації пошуку та фільтрації бібліографічних ресурсів.

Таким чином, спроєктована база даних та модель предметної області забезпечують надійну основу для реалізації функціональних можливостей веб-платформи управління бібліотечними процесами, підтримують інтеграцію з відкритими бібліографічними сервісами та створюють передумови для подальшого масштабування і розвитку системи.

2.5 Реалізація серверної частини веб-платформи

Серверна частина веб-платформи управління бібліотечними процесами виконує ключову роль у забезпеченні бізнес-логіки системи, обробці запитів користувачів та взаємодії з базою даних і зовнішніми сервісами. Вона є центральним компонентом архітектури, який забезпечує узгоджену роботу всіх функціональних модулів та відповідає за цілісність і безпеку даних.

При реалізації серверної частини було використано підхід, орієнтований на створення RESTful веб-сервісів, що дозволяє стандартизувати обмін даними між клієнтською частиною та сервером. Такий підхід забезпечує незалежність

інтерфейсу користувача від внутрішньої реалізації серверної логіки та спрощує інтеграцію з іншими програмними системами.

Серверна логіка реалізує основні функції управління бібліотечними процесами, зокрема обробку бібліографічних записів, управління користувачами та ролями доступу, облік примірників і фіксацію операцій обслуговування. Усі запити від клієнтської частини проходять перевірку коректності та прав доступу, що дозволяє забезпечити захист інформаційних ресурсів бібліотеки.

Важливим аспектом реалізації серверної частини є організація взаємодії з базою даних. Сервер забезпечує виконання операцій створення, читання, оновлення та видалення даних, дотримуючись принципів цілісності та узгодженості інформації. Для цього використовуються відповідні механізми транзакційності, що дозволяє уникнути втрати або пошкодження даних у разі виникнення помилок.

Серверна частина також відповідає за реалізацію механізмів автентифікації та авторизації користувачів. З цією метою використовується модель доступу на основі ролей, яка дозволяє чітко розмежувати повноваження між читачами, бібліотекарями та адміністраторами системи. Такий підхід забезпечує контроль доступу до функціональних можливостей платформи та підвищує загальний рівень безпеки.

Особливу увагу приділено інтеграції серверної частини з зовнішніми бібліографічними сервісами. Сервер виступає посередником між внутрішніми модулями платформи та зовнішніми API, виконуючи перетворення форматів даних та обробку відповідей сервісів. Це дозволяє використовувати зовнішні джерела інформації без порушення внутрішньої логіки системи.

З метою підвищення продуктивності та масштабованості серверної частини реалізовано механізми оптимізації обробки запитів. Зокрема, застосовуються підходи до кешування часто використовуваних даних та асинхронної обробки запитів, що дозволяє зменшити навантаження на базу даних і скоротити час відповіді системи.

Серверна частина веб-платформи розроблена з урахуванням принципів модульності та розширюваності. Це дозволяє в майбутньому додавати нові функціональні можливості або змінювати існуючі без необхідності суттєвої переробки всієї системи. Такий підхід є особливо важливим для бібліотечних інформаційних систем, які повинні адаптуватися до змін стандартів і вимог користувачів.

Таким чином, реалізація серверної частини веб-платформи забезпечує надійну основу для функціонування системи, підтримує основні бібліотечні процеси та створює умови для подальшого розвитку й інтеграції платформи з іншими інформаційними ресурсами.

2.6 Реалізація клієнтської частини

Клієнтська частина веб-платформи управління бібліотечними процесами відповідає за безпосередню взаємодію користувачів із системою та відіграє ключову роль у забезпеченні зручності, доступності та ефективності роботи з бібліографічними ресурсами. Від якості реалізації клієнтського інтерфейсу значною мірою залежить рівень задоволеності користувачів та швидкість виконання основних бібліотечних операцій.

Проєктування клієнтської частини здійснювалось з урахуванням вимог до веб-орієнтованості, кросплатформеності та адаптивності. Веб-платформа повинна коректно функціонувати на різних пристроях і в різних браузерах без необхідності встановлення додаткового програмного забезпечення. Для цього використано сучасні підходи до розробки односторінкових веб-застосунків, що дозволяє мінімізувати кількість перезавантажень сторінок та забезпечити швидку реакцію інтерфейсу на дії користувача [20].

Клієнтська частина реалізує логіку відображення бібліографічних даних, отриманих із серверної частини через REST API. Дані про бібліографічні записи, авторів, примірники та користувачів відображаються у структурованому та зрозумілому вигляді, що полегшує навігацію та пошук необхідної інформації.

Особливу увагу приділено представленню результатів пошуку, оскільки пошукові операції є найбільш часто використовуваними у бібліотечних системах.

Інтерфейс користувача побудовано з урахуванням ролей доступу. Залежно від типу користувача (читач, бібліотекар, адміністратор) система відображає відповідний набір функціональних можливостей. Такий підхід дозволяє зменшити інформаційне навантаження на користувача та підвищити безпеку системи шляхом обмеження доступу до адміністративних функцій.

Важливим аспектом реалізації клієнтської частини є забезпечення зручної навігації між основними функціональними розділами платформи. Структура інтерфейсу організована таким чином, щоб користувач міг швидко переходити між каталогом, сторінками окремих бібліографічних записів, особистим кабінетом та адміністративними інструментами. Для цього використовуються інтуїтивно зрозумілі елементи керування та логічна ієрархія сторінок [21].

Клієнтська частина також відповідає за валідацію введених користувачем даних перед їх передачею на сервер. Це дозволяє зменшити кількість помилок, підвищити зручність користування системою та знизити навантаження на серверну частину. Валідація охоплює перевірку коректності введених текстових даних, форматів ідентифікаторів, а також обов’язковості заповнення окремих полів.

Для підвищення продуктивності та зручності використання платформи реалізовано механізми асинхронного завантаження даних. Це дозволяє оновлювати окремі частини інтерфейсу без повного перезавантаження сторінки та забезпечує плавну взаємодію користувача з системою. Такий підхід особливо важливий при роботі з великими обсягами бібліографічних даних.

Окрему увагу приділено питанням доступності та зручності сприйняття інформації. Інтерфейс клієнтської частини розроблено з урахуванням принципів читабельності, контрастності та логічного групування інформації. Це дозволяє забезпечити комфортну роботу користувачів з різним рівнем підготовки та досвіду.

Таким чином, реалізація клієнтської частини веб-платформи забезпечує ефективну взаємодію користувачів з бібліотечною системою, підтримує основні

бібліотечні процеси та створює умови для подальшого розширення функціональності платформи без суттєвих змін у її архітектурі.

2.7 Інтеграція з MARC-каталогами

Інтеграція з MARC-каталогами є однією з ключових функціональних можливостей розроблюваної веб-платформи управління бібліотечними процесами, оскільки саме формат MARC21 залишається основним стандартом обміну бібліографічними даними між бібліотечними інформаційними системами. Забезпечення сумісності з MARC-каталогами дозволяє використовувати наявні бібліографічні записи, уникати дублювання даних та спростити процес переходу з існуючих автоматизованих бібліотечних систем на нову платформу.

У межах розроблюваної веб-платформи інтеграція з MARC-каталогами реалізується шляхом підтримки імпорту та експорту бібліографічних записів у форматі MARC21. Це дозволяє здійснювати обмін даними з такими open-source бібліотечними системами, як Koha та Evergreen, а також з іншими бібліотечними каталогами, що використовують даний стандарт. Важливим завданням є збереження повноти та коректності бібліографічної інформації під час перетворення MARC-записів у внутрішні моделі даних системи.

Процес інтеграції передбачає аналіз структури MARC21-записів та зіставлення їхніх полів з атрибутами внутрішньої моделі бібліографічного запису. Зокрема, поля, що містять відомості про автора, назву, вихідні дані, предметні рубрики та ідентифікатори, трансформуються у відповідні атрибути бази даних веб-платформи. Такий підхід дозволяє забезпечити логічну узгодженість між зовнішнім стандартом представлення даних та внутрішньою структурою системи.

Особливу увагу приділено обробці змінних полів та підполів MARC21, оскільки саме вони забезпечують гнучкість опису бібліографічних ресурсів. Під час інтеграції реалізовано механізми перевірки коректності даних, що дозволяють виявляти помилки форматування та запобігати втраті важливої інформації. Це є

особливо актуальним при імпорті записів з різних джерел, які можуть відрізнятися за ступенем відповідності стандарту.

Використання MARC-каталогів у сучасних веб-орієнтованих системах потребує адаптації традиційного підходу до обміну даними. З цією метою в розроблюваній платформі реалізовано перетворення MARC21-записів у веб-сумісні формати, зокрема JSON, що дозволяє використовувати REST API для передачі бібліографічної інформації між компонентами системи. Такий підхід забезпечує поєднання переваг класичного бібліотечного стандарту та сучасних технологій веб-розробки.

Інтеграція з MARC-каталогами також сприяє автоматизації процесу наповнення електронного каталогу. Завдяки можливості імпорту готових бібліографічних записів значно скорочується час, необхідний для каталогізації нових ресурсів, а також зменшується навантаження на бібліотечний персонал. Це підвищує ефективність роботи бібліотеки та дозволяє зосередитись на аналітичних і сервісних завданнях.

Крім того, підтримка MARC-каталогів забезпечує можливість подальшої інтеграції веб-платформи з національними та міжнародними бібліотечними мережами. Використання загальноприйнятого стандарту обміну даними створює передумови для розвитку міжбібліотечної взаємодії, обміну бібліографічними записами та спільного використання інформаційних ресурсів.

Таким чином, реалізована інтеграція з MARC-каталогами є важливим елементом архітектури веб-платформи управління бібліотечними процесами. Вона забезпечує сумісність з існуючими бібліотечними системами, підвищує ефективність каталогізації та створює основу для подальшого розвитку інтеграційних можливостей системи в межах відкритих програмних рішень.

2.8 Інтеграція з DOI та Google Books API

Одним із ключових завдань розроблюваної веб-платформи управління бібліотечними процесами є забезпечення інтеграції з зовнішніми інформаційними сервісами, які дозволяють автоматично збагачувати бібліографічні записи та підвищувати якість електронного каталогу. Серед таких сервісів особливе місце займають системи ідентифікації наукових публікацій DOI (Digital Object Identifier) та відкриті бібліографічні сервіси, зокрема Google Books API.

Використання зовнішніх API у бібліотечних системах відповідає сучасним тенденціям розвитку інформаційних технологій та сприяє інтеграції бібліотек у глобальний інформаційний простір. Це дозволяє автоматизувати процес наповнення каталогу, зменшити обсяг ручної роботи бібліотекарів та забезпечити користувачів актуальною й достовірною інформацією.

DOI є унікальним цифровим ідентифікатором, який використовується для однозначної ідентифікації наукових публікацій, статей, монографій та інших електронних ресурсів. Основною перевагою DOI є стабільність посилань та можливість отримання структурованих метаданих про публікацію незалежно від її фізичного місця зберігання [22].

У межах розроблюваної веб-платформи інтеграція з DOI-сервісами реалізується шляхом використання відкритих API, які дозволяють здійснювати запити за ідентифікатором DOI та отримувати повний набір бібліографічних метаданих. Отримана інформація може включати назву публікації, авторів, рік видання, видавця, анотацію та інші важливі атрибути.

Процес інтеграції з DOI передбачає такі етапи:

- введення або імпорт DOI-коду користувачем чи бібліотекарем;
- формування запиту до зовнішнього DOI-сервісу;
- отримання та обробку відповіді у форматі JSON або XML;
- зіставлення отриманих метаданих з внутрішньою моделлю бібліографічного запису;
- збереження даних у базі даних платформи.

Google Books API є відкритим програмним інтерфейсом, що надає доступ до великої кількості бібліографічних даних про книги, включаючи інформацію про видання, авторів, обкладинки, анотації та фрагменти текстів. Інтеграція з цим сервісом значно розширює функціональні можливості бібліотечної веб-платформи.

У межах платформи Google Books API використовується для автоматичного отримання додаткових відомостей про книги за такими ідентифікаторами, як ISBN або назва видання. Це дозволяє збагачувати бібліографічні записи мультимедійними елементами та покращувати користувацький досвід.

Основними функціями інтеграції з Google Books API є:

- отримання обкладинок книг;
- завантаження анотацій та описів;
- уточнення бібліографічних даних;
- перевірка наявності електронних версій видань.

Інформація, отримана з Google Books API, після обробки зберігається у базі даних системи та використовується для відображення в електронному каталозі.

Такий підхід дозволяє забезпечити автоматичне наповнення бібліографічних записів та мінімізувати ймовірність помилок, пов’язаних з ручним введенням інформації.

У результаті реалізації інтеграції з DOI та Google Books API веб-платформа отримує можливість автоматичного збагачення бібліографічних записів та підвищення ефективності бібліотечних процесів. Запропоновані рішення забезпечують гнучкість, масштабованість та відповідність сучасним вимогам до бібліотечних інформаційних систем.

2.9 Висновки до другого розділу

У другому розділі магістерської роботи було виконано проєктування та розробку веб-платформи для управління бібліотечними процесами на основі відкритих програмних рішень. На початковому етапі обґрунтовано вибір архітектурного стилю та технологічного стеку з урахуванням вимог модульності,

масштабованості та можливості інтеграції із зовнішніми інформаційними сервісами. Обрані архітектурні підходи відповідають сучасним тенденціям розробки веб-орієнтованих інформаційних систем.

У процесі проєктування модульної архітектури систему було декомпозовано на окремі функціональні компоненти, що забезпечують реалізацію основних бібліотечних процесів, зокрема управління користувачами, каталогізацію, пошук, облік примірників та аналітику. Такий підхід дозволяє зменшити зв’язність між компонентами системи та спрощує її подальший супровід і розширення.

Значну увагу приділено проєктуванню бази даних та моделей предметної області, які відображають логіку бібліотечних процесів і забезпечують збереження структурованих бібліографічних даних. Запропонована структура бази даних підтримує роботу зі стандартом MARC21 та дозволяє зберігати зовнішні ідентифікатори, що є необхідним для інтеграції з міжнародними бібліографічними сервісами.

У розділі також описано реалізацію серверної частини веб-платформи, яка забезпечує обробку бізнес-логіки, взаємодію з базою даних і зовнішніми API, а також реалізацію клієнтської частини, орієнтованої на зручний та інтуїтивно зрозумілий користувацький інтерфейс. Взаємодія між клієнтом і сервером реалізована за допомогою REST API, що підвищує гнучкість та незалежність компонентів системи.

Окремо розглянуто питання інтеграції з MARC-каталогами, що забезпечує сумісність розробленої платформи з існуючими бібліотечними системами, а також інтеграції з DOI-сервісами та Google Books API, які дозволяють збагачувати бібліографічні записи додатковими метаданими та покращувати якість електронного каталогу.

Таким чином, у другому розділі було сформовано повноцінну архітектурну та технологічну основу веб-платформи управління бібліотечними процесами. Отримані результати створюють передумови для проведення тестування, оцінки якості та впровадження розробленої системи, що буде розглянуто у наступному розділі магістерської роботи.

3 ТЕСТУВАННЯ, ВПРОВАДЖЕННЯ ТА ПІДТРИМКА ВЕБ-ПЛАТФОРМИ

У даному розділі розглянуто процеси тестування, впровадження та супроводу розробленої веб-платформи управління бібліотечними процесами. Описано підходи до перевірки коректності роботи програмних компонентів, оцінки якості та надійності системи, а також особливості її розгортання та подальшої експлуатації.

3.1 Планування та організація тестування веб-платформи

Тестування програмного забезпечення є невід’ємним етапом життєвого циклу розробки інформаційних систем і відіграє ключову роль у забезпеченні якості, надійності та відповідності програмного продукту визначеним вимогам. Для веб-платформи управління бібліотечними процесами тестування має особливе значення, оскільки система працює з критично важливими бібліографічними даними, підтримує багатокористувацький доступ та інтегрується з зовнішніми інформаційними сервісами [23].

Планування тестування передбачає визначення цілей, обсягів, методів і ресурсів, необхідних для перевірки коректності роботи веб-платформи. На цьому етапі формуються основні підходи до контролю якості програмного забезпечення та визначається стратегія тестування, яка охоплює всі рівні системи — від окремих програмних модулів до платформи в цілому.

Основною метою тестування розробленої веб-платформи є виявлення дефектів на ранніх етапах експлуатації, перевірка відповідності функціональних можливостей системи вимогам, сформованим у попередніх розділах, а також оцінка стабільності та продуктивності системи в умовах реального використання. Крім того, тестування спрямоване на перевірку коректності інтеграції платформи з MARC-каталогами, DOI-сервісами та Google Books API.

Організація процесу тестування ґрунтується на поетапному підході, що відповідає загальноприйнятим практикам програмної інженерії. На першому етапі здійснюється аналіз вимог до системи та визначення тестових сценаріїв, які охоплюють основні бізнес-процеси бібліотечної діяльності. На цьому етапі також визначаються критерії приймання системи та показники якості, за якими оцінюється результат тестування.

Наступним етапом є підготовка тестового середовища, яке повинно максимально наближено відтворювати умови реальної експлуатації веб-платформи. До складу тестового середовища входять серверна та клієнтська частини системи, база даних із тестовими бібліографічними записами, а також емулятори або тестові облікові записи для зовнішніх сервісів інтеграції. Це дозволяє перевірити роботу системи в умовах, наближених до продуктивного середовища.

Важливою складовою планування тестування є визначення ролей і відповідальності учасників процесу. У межах магістерської роботи функції тестувальника та розробника поєднуються, що вимагає особливої уваги до об’єктивності оцінки результатів тестування. Для зменшення суб’єктивного впливу використовуються заздалегідь підготовлені тест-кейси та чек-листи.

Процес тестування веб-платформи передбачає застосування різних видів тестування, кожен з яких орієнтований на перевірку певних аспектів системи. Зокрема, виконуються функціональне тестування, інтеграційне тестування, тестування безпеки та тестування продуктивності. Такий комплексний підхід дозволяє виявити як логічні помилки в реалізації функціоналу, так і проблеми, пов’язані з навантаженням або некоректною взаємодією компонентів системи.

Особливу увагу приділено плануванню тестування інтеграційних механізмів, оскільки взаємодія з зовнішніми сервісами є одним із найбільш вразливих аспектів веб-платформи. Для цього передбачено тестування сценаріїв обробки помилок, пов’язаних із недоступністю API або некоректними відповідями зовнішніх сервісів.

Результати тестування фіксуються у вигляді звітів, які містять опис виявлених дефектів, умови їх виникнення та рекомендації щодо усунення. На

основі цих звітів здійснюється коригування програмного коду та повторне тестування до досягнення прийнятного рівня якості системи.

Таким чином, планування та організація тестування веб-платформи є комплексним процесом, що забезпечує контроль якості на всіх етапах розробки та впровадження. Реалізований підхід до тестування дозволяє підвищити надійність системи, забезпечити її відповідність функціональним вимогам і створити передумови для успішного використання веб-платформи в бібліотечній практиці.

3.2 Модульне та інтеграційне тестування

Модульне та інтеграційне тестування є ключовими етапами перевірки якості програмного забезпечення, оскільки дозволяють виявити помилки як у межах окремих компонентів, так і у процесі їх взаємодії. Для розробленої веб-платформи управління бібліотечними процесами ці види тестування мають особливе значення через модульну архітектуру системи.

Модульне тестування спрямоване на перевірку коректності роботи окремих програмних модулів у відриві від інших компонентів системи. Основною метою є виявлення логічних помилок, некоректної обробки даних та порушень бізнес-логіки.

У межах модульного тестування перевірялися такі компоненти:

- модуль управління користувачами;
- модуль бібліографічного каталогу;
- модуль пошуку;
- модуль інтеграції з зовнішніми сервісами.

Нижче наведено таблиця 3.1 тест-кейсів для модульного тестування основних функціональних компонентів веб-платформи.

Таблиця 3.1 – Тест-кейсів модульного тестування

Назва тесту	Вхідні дані	Очікувальний результат	Фактичний результат
1	2	3	4
Пошук книги	Ключове слово	Повернено результати	Відповідає
Реєстрація користувача	Коректні облікові дані	Користувач створений	Відповідає
Авторизація користувача	Невірний пароль	Відмова у доступі	Відповідає
Додавання книги	Коректні бібліографічні дані	Запис збережено	Відповідає

Інтеграційне тестування спрямоване на перевірку взаємодії між окремими модулями системи та коректності обміну даними між ними. Особлива увага приділялася перевірці інтеграції з базою даних і зовнішніми API.

У процесі інтеграційного тестування перевірялися такі сценарії:

- взаємодія клієнтської та серверної частин;
- коректність обміну даними через REST API;
- інтеграція з MARC-каталогами;
- інтеграція з DOI та Google Books API.

Нижче наведено таблиця 3.2 тест-кейсів для інтеграційного тестування основних функціональних компонентів веб-платформи.

Таблиця 3.2 – Тест-кейсів модульного тестування

Назва тесту	Вхідні дані	Очікувальний результат	Фактичний результат
1	2	3	4
Пошук книги	Запит з клієнта до сервера	Отримання результатів	Успішно
Імпорт Marc	Завантаження Marc-запису	Коректний імпорт	Успішно
Отримання DOI	Запит до DOI-сервісу	Повернення метаданих	Успішно
Google Books API	Отримання обкладинки	Дані відображені	Успішно

Результати інтеграційного тестування підтвердили коректність взаємодії між модулями веб-платформи та стабільність роботи системи в умовах реального використання.

3.3 Функціональне та нефункціональне тестування

Функціональне та нефункціональне тестування є важливими складовими процесу забезпечення якості програмного забезпечення та дозволяють комплексно оцінити відповідність веб-платформи визначеним вимогам. На відміну від модульного та інтеграційного тестування, які зосереджені на перевірці коректності взаємодії окремих компонентів, функціональне та нефункціональне тестування спрямовані на оцінку поведінки системи в цілому з точки зору кінцевого користувача та експлуатаційних характеристик.

Функціональне тестування полягає у перевірці реалізації всіх функцій веб-платформи відповідно до сформованих у другому розділі функціональних вимог. Основною метою цього виду тестування є підтвердження того, що система

коректно виконує задані сценарії використання та забезпечує очікувану поведінку в стандартних і граничних умовах.

У межах функціонального тестування перевіряються такі аспекти:

- коректність роботи механізмів автентифікації та авторизації користувачів;
- функціонування бібліографічного каталогу та відображення даних;
- робота пошукових та фільтраційних механізмів;
- виконання операцій з бібліотечними ресурсами;
- інтеграційні функції взаємодії з зовнішніми сервісами.

Функціональне тестування проводилось на основі заздалегідь визначених сценаріїв використання, що відповідають типовим діям користувачів системи. Такий підхід дозволяє оцінити відповідність реалізованого функціоналу реальним потребам бібліотечної діяльності та забезпечити зручність використання веб-платформи [24].

Нефункціональне тестування доповнює функціональне та спрямоване на оцінку характеристик системи, які не пов’язані безпосередньо з реалізацією окремих функцій, але суттєво впливають на якість програмного забезпечення. До таких характеристик належать продуктивність, надійність, безпека, масштабованість та зручність використання.

Одним із ключових аспектів нефункціонального тестування є тестування продуктивності, яке дозволяє визначити здатність веб-платформи ефективно працювати при зростанні кількості користувачів та обсягу оброблюваних даних. У процесі тестування оцінюється час відгуку системи, стабільність роботи під навантаженням та використання серверних ресурсів.

Важливе значення має також тестування безпеки, яке спрямоване на виявлення потенційних вразливостей системи. Перевіряється захищеність механізмів автентифікації, коректність обробки вхідних даних та дотримання принципів контролю доступу. Це особливо актуально для бібліотечних систем, що працюють з персональними даними користувачів.

Окрему увагу приділено тестуванню зручності використання, яке дозволяє оцінити інтуїтивність інтерфейсу та ефективність взаємодії користувача з веб-платформою. Зручний та зрозумілий інтерфейс є важливим чинником успішного впровадження системи в бібліотечній практиці.

Таким чином, функціональне та нефункціональне тестування дозволяють всебічно оцінити якість розробленої веб-платформи та підтвердити її готовність до експлуатації в реальних умовах.

3.4 Аналіз безпеки та захисту даних

Забезпечення інформаційної безпеки є одним із ключових аспектів розробки сучасних веб-платформ, особливо тих, що працюють з персональними даними користувачів та бібліографічною інформацією. Бібліотечна веб-платформа, що розробляється в межах даної магістерської роботи, передбачає зберігання та обробку даних про користувачів, облікові записи, бібліографічні ресурси, а також результати інтеграції із зовнішніми інформаційними сервісами. У зв’язку з цим питання безпеки та захисту даних є критично важливими.

Інформаційна безпека веб-платформи розглядається як сукупність організаційних, програмних та архітектурних заходів, спрямованих на забезпечення конфіденційності, цілісності та доступності інформації. Кожен із зазначених аспектів має бути врахований на етапі проєктування архітектури системи та підтверджений під час тестування.

Першим етапом аналізу безпеки є визначення можливих загроз, які можуть вплинути на стабільність та надійність роботи веб-платформи. До основних загроз належать:

- несанкціонований доступ до облікових записів користувачів;
- перехоплення даних під час передачі мережею;
- несанкціонована модифікація бібліографічних записів;
- зловмисне видалення або пошкодження даних;
- атаки типу SQL-ін’єкцій та XSS;

- перевантаження системи шляхом атаки типу відмова в обслуговуванні.

З урахуванням того, що веб-платформа підтримує інтеграцію з зовнішніми API (DOI, Google Books), додатковим фактором ризику є обробка даних, отриманих із зовнішніх джерел, які можуть містити некоректну або шкідливу інформацію.

Одним із базових механізмів забезпечення безпеки є система автентифікації та авторизації користувачів. У межах платформи реалізовано рольову модель доступу, яка передбачає розмежування прав між звичайними користувачами, бібліотекарями та адміністраторами системи. Такий підхід дозволяє мінімізувати ризик несанкціонованого доступу до критично важливих функцій.

Паролі користувачів зберігаються в захищеному вигляді з використанням сучасних криптографічних алгоритмів хешування. Це унеможливлює отримання паролів у відкритому вигляді навіть у разі компрометації бази даних. Додатково передбачено механізми контролю сесій користувачів та обмеження часу їхньої активності [25].

Передача даних між клієнтською та серверною частинами веб-платформи здійснюється із застосуванням захищених мережевих протоколів. Це дозволяє запобігти перехопленню або підміні даних під час обміну інформацією.

На рівні зберігання даних реалізовано механізми контролю доступу до бази даних, що обмежують можливість прямого доступу до інформації ззовні. Резервне копіювання даних дозволяє забезпечити їх відновлення у разі технічних збоїв або втрати інформації.

Інтеграція з DOI-сервісами та Google Books API потребує особливої уваги з точки зору безпеки, оскільки система обробляє дані, отримані з зовнішніх джерел. Для мінімізації ризиків усі вхідні дані проходять перевірку коректності та фільтрацію перед збереженням у базі даних або відображенням у користувацькому інтерфейсі.

Використання API-ключів для доступу до зовнішніх сервісів здійснюється відповідно до рекомендацій постачальників сервісів. Ключі зберігаються у захищеному середовищі та не передаються клієнтській частині веб-платформи.

Цілісність даних забезпечується шляхом використання транзакцій при виконанні операцій додавання, редагування та видалення інформації. Це дозволяє уникнути часткових змін у разі виникнення помилок та підтримувати узгоджений стан бази даних [26].

Для контролю коректності операцій використовуються механізми валідації даних, які перевіряють відповідність введеної інформації встановленим правилам. Це особливо важливо для бібліографічних записів, що мають складну структуру та повинні відповідати стандарту MARC21.

Проведений аналіз безпеки показав, що використані архітектурні та програмні рішення забезпечують базовий рівень захисту даних веб-платформи. Разом із тим, для подальшого розвитку системи доцільно розглянути можливість впровадження додаткових механізмів безпеки, зокрема двофакторної автентифікації та розширеного журналювання подій.

Таким чином, комплексний підхід до аналізу безпеки та захисту даних дозволяє забезпечити надійну та стабільну роботу веб-платформи управління бібліотечними процесами та відповідність сучасним вимогам інформаційної безпеки.

3.5 Впровадження веб-платформи в експлуатацію

Впровадження веб-платформи в експлуатацію є завершальним етапом життєвого циклу розробки програмного забезпечення та передбачає комплекс організаційних, технічних і експлуатаційних заходів, спрямованих на введення системи в реальне використання. На цьому етапі відбувається перехід від тестового середовища до продуктивної експлуатації з реальними користувачами та даними.

Процес впровадження веб-платформи управління бібліотечними процесами розпочинається з підготовки серверної інфраструктури. Для цього здійснюється вибір середовища розгортання, яке забезпечує необхідний рівень продуктивності, надійності та безпеки. Як правило, використовується виділений сервер або хмарна

платформа, що дозволяє масштабувати ресурси залежно від навантаження та кількості користувачів [27].

На етапі розгортання виконується встановлення серверного програмного забезпечення, налаштування веб-сервера, системи управління базами даних та середовища виконання серверної частини платформи. Особливу увагу приділено конфігурації параметрів безпеки, зокрема обмеженню доступу до службових ресурсів, налаштуванню прав доступу та використанню захищених протоколів передавання даних.

Після налаштування інфраструктури здійснюється розгортання програмного коду веб-платформи та ініціалізація бази даних. На цьому етапі до системи можуть бути імпортовані початкові бібліографічні дані, зокрема записи у форматі MARC21, а також створені облікові записи користувачів і ролі доступу. Імпорт даних дозволяє перевірити коректність роботи інтеграційних механізмів та забезпечити готовність системи до практичного використання [28].

Важливою складовою впровадження є налаштування механізмів резервного копіювання та відновлення даних. Регулярне створення резервних копій бази даних і конфігураційних файлів забезпечує захист інформації від втрати внаслідок збоїв апаратного забезпечення або помилок програмного характеру. Це є критично важливим для бібліотечних систем, які оперують значними обсягами цінних бібліографічних даних.

Перед повноцінним введенням веб-платформи в експлуатацію проводиться навчання користувачів, зокрема бібліотекарів та адміністраторів системи. Навчання може включати ознайомлення з інтерфейсом платформи, основними функціями, правилами роботи з бібліографічними записами та особливостями адміністрування системи. Це сприяє зменшенню кількості помилок на початковому етапі використання та підвищує ефективність роботи користувачів [29].

Після завершення підготовчих заходів веб-платформа переводиться у продуктивний режим роботи. На цьому етапі здійснюється моніторинг функціонування системи з метою виявлення можливих помилок, проблем продуктивності або збоїв у роботі інтеграцій з зовнішніми сервісами. Моніторинг

дозволяє оперативно реагувати на виявлені недоліки та забезпечувати стабільну роботу платформи.

У процесі експлуатації веб-платформи важливу роль відіграє технічний супровід, який включає оновлення програмного забезпечення, усунення виявлених помилок, оптимізацію продуктивності та адаптацію системи до змін у нормативних вимогах або технологічному середовищі. Завдяки використанню модульної архітектури такі оновлення можуть здійснюватися без суттєвого впливу на роботу всієї системи [30].

Таким чином, впровадження веб-платформи в експлуатацію забезпечує її готовність до практичного використання в бібліотечній установі та створює умови для подальшого розвитку й удосконалення системи відповідно до потреб користувачів і сучасних вимог до бібліотечних інформаційних систем.

3.6 Супровід та подальший розвиток системи

Після впровадження веб-платформи управління бібліотечними процесами в експлуатацію важливим етапом її життєвого циклу є супровід та подальший розвиток. Супровід програмного забезпечення спрямований на забезпечення стабільної роботи системи, своєчасне усунення виявлених недоліків, а також адаптацію до змін у вимогах користувачів і зовнішнього середовища.

Супровід веб-платформи включає комплекс організаційних та технічних заходів, серед яких основними є моніторинг працездатності системи, аналіз журналів подій, оновлення програмних компонентів та резервне копіювання даних. Регулярний моніторинг дозволяє оперативно виявляти помилки в роботі системи, збої в інтеграції з зовнішніми сервісами та проблеми з продуктивністю.

Особливу увагу під час супроводу слід приділяти оновленню залежностей та бібліотек, які використовуються у серверній та клієнтській частинах платформи. Оскільки система побудована з використанням відкритих програмних рішень, оновлення компонентів є необхідним для усунення вразливостей безпеки та підтримки сумісності з сучасними технологіями.

Подальший розвиток веб-платформи передбачає розширення її функціональних можливостей відповідно до зростаючих потреб бібліотеки та користувачів. До потенційних напрямів розвитку системи можна віднести:

- розширення підтримки бібліографічних стандартів;
- інтеграцію з новими зовнішніми інформаційними сервісами та науковими репозитаріями;
- впровадження механізмів персоналізації користувацького інтерфейсу;
- використання аналітичних інструментів для аналізу поведінки користувачів.

Значні перспективи розвитку платформи пов’язані з впровадженням елементів мікросервісної архітектури, що дозволить масштабувати окремі компоненти системи незалежно один від одного. Це особливо актуально для модулів інтеграції та пошуку, які можуть зазнавати підвищених навантажень.

Крім того, подальший розвиток системи може передбачати автоматизацію процесів оновлення та розгортання програмного забезпечення з використанням інструментів безперервної інтеграції та доставки (CI/CD). Такий підхід сприятиме підвищенню якості програмного продукту та скороченню часу впровадження нових функцій [31].

Важливою складовою супроводу є робота з користувачами системи, яка включає збір зворотного зв’язку, аналіз пропозицій щодо вдосконалення функціоналу та підготовку навчальних матеріалів. Це дозволяє адаптувати веб-платформу до реальних умов експлуатації та підвищувати рівень задоволеності користувачів.

Таким чином, супровід та подальший розвиток веб-платформи управління бібліотечними процесами є безперервним процесом, що забезпечує її актуальність, надійність та відповідність сучасним вимогам інформаційних систем. Реалізація комплексного підходу до супроводу дозволяє максимально ефективно використовувати потенціал розробленої системи та створює передумови для її довготривалого використання.

3.7 Висновки до третього розділу

У третьому розділі було розглянуто основні аспекти тестування, впровадження та супроводу веб-платформи управління бібліотечними процесами. Проведене тестування підтвердило коректність реалізації функціональних можливостей системи та її відповідність поставленим вимогам. Запропоновані підходи до впровадження та подальшого розвитку забезпечують можливість практичного використання розробленої платформи та її адаптації до потреб конкретних бібліотек

4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Під час розробки та експлуатації веб-платформи для управління бібліотечними процесами враховувалися вимоги охорони праці, безпеки життєдіяльності та пожежної безпеки відповідно до чинного законодавства України.

4.1 Охорона праці

Особливу увагу приділено умовам праці розробників програмного забезпечення та кінцевих користувачів системи, які працюють з персональними комп'ютерами та іншими засобами обчислювальної техніки [32].

При розробці програмного продукту та організації робочих місць використовувалися вимоги таких нормативно-правових актів України:

Закон України «Про охорону праці»;

- Кодекс законів про працю України;
- Закон України «Про пожежну безпеку»;
- Правила пожежної безпеки в Україні (Наказ МВС №1064 від 26.12.2018);
- ДБН В.1.1-7:2016 «Пожежна безпека об'єктів будівництва»;
- Вимоги НПАОП 40.1-1.21-98 «Правила безпечної експлуатації електроустановок споживачів».
- Вимоги НПАОП 40.1-1.32-01 «Правила технічної експлуатації електроустановок споживачів».
- Правила улаштування електроустановок (ПУЕ);
- Правила технічної експлуатації електроустановок споживачів (Наказ Мінпаливенерго №258);
- ДСТУ EN 62305:2012 «Захист від блискавки» (у частині електробезпеки та захисту від перенапруг).

Зазначені документи регламентують вимоги до безпечних умов праці, організації робочих місць, електробезпеки та пожежної безпеки під час роботи з комп’ютерною технікою.

Розробка веб-платформи здійснювалася з урахуванням особливостей праці фахівців з програмної інженерії, діяльність яких належить до розумової праці з підвищеним зоровим та психоемоційним навантаженням [33].

При проєктуванні програмного забезпечення враховувалися такі фактори:

- тривала робота за персональним комп’ютером;
- необхідність концентрації уваги;
- вплив електромагнітного випромінювання;
- статичні навантаження на опорно-руховий апарат;
- ризики перевтоми та професійних захворювань.

Для зменшення негативного впливу цих факторів при розробці інтерфейсу веб-платформи реалізовано принципи зручності використання (usability) та ергономіки інтерфейсу, зокрема:

- використання нейтральних кольорових схем;
- оптимальний контраст тексту та фону;
- логічне структурування елементів інтерфейсу;
- мінімізація кількості дій для виконання основних операцій;
- адаптивність інтерфейсу до різних розмірів екранів.

Такі рішення сприяють зменшенню зорової втоми та психоемоційного навантаження користувачів системи.

Робочі місця розробників та користувачів веб-платформи повинні відповідати ергономічним вимогам, встановленим чинними санітарними нормами України [33].

Основні вимоги до організації робочого місця:

- висота робочого столу — 720–750 мм;
- регульоване крісло з підтримкою поперекового відділу;
- розташування монітора на відстані 50–70 см від очей;
- верхній край екрана — на рівні очей або трохи нижче;

- клавіатура та миша — на рівні ліктів користувача.

Для зниження втоми передбачено дотримання режимів праці та відпочинку відповідно до рекомендацій з охорони праці при роботі з ПК.

Електробезпека користувачів веб-платформи забезпечується шляхом дотримання вимог Правил улаштування електроустановок та інших нормативних документів.

Основні заходи електробезпеки включають:

- використання справного електрообладнання;
- заземлення електроприладів;
- застосування автоматичних вимикачів і захисних пристроїв;
- заборона експлуатації пошкоджених кабелів і розеток;
- періодичний технічний огляд обладнання.

Використання сучасної комп’ютерної техніки з низьким рівнем електромагнітного випромінювання зменшує ризик негативного впливу на здоров’я користувачів.

Пожежна безпека при розробці та експлуатації веб-платформи забезпечується відповідно до Закону України «Про пожежну безпеку» та Правил пожежної безпеки в Україні.

Основні заходи пожежної безпеки:

- використання сертифікованого електрообладнання;
- дотримання норм електробезпеки;
- наявність первинних засобів пожежогасіння;
- заборона перевантаження електромереж;
- інструктаж персоналу з пожежної безпеки.

Комп’ютерні приміщення повинні бути обладнані засобами пожежогасіння та системами оповіщення у разі виникнення надзвичайних ситуацій [33].

У процесі розробки веб-платформи для управління бібліотечними процесами враховано вимоги охорони праці та пожежної безпеки відповідно до чинного законодавства України. Реалізовані технічні та організаційні заходи спрямовані на створення безпечних і комфортних умов праці для розробників та користувачів

системи, зниження ризиків професійних захворювань і запобігання виникненню надзвичайних ситуацій.

4.2 Фактори, що впливають на функціональний стан користувачів комп’ютерів.

У сучасних умовах цифровізації та широкого впровадження інформаційних технологій у всі сфери діяльності людини використання персональних комп’ютерів стало невід’ємною складовою професійної діяльності фахівців, зокрема інженерів програмного забезпечення. Тривала робота за комп’ютером супроводжується впливом комплексу виробничих факторів, які безпосередньо або опосередковано впливають на функціональний стан організму користувачів, їх працездатність, психоемоційний стан та загальний рівень здоров’я.

Функціональний стан користувача комп’ютера визначається як інтегральна характеристика фізіологічних, психічних та психофізіологічних процесів організму, що забезпечують ефективне виконання професійної діяльності. Зміни цього стану можуть проявлятися у вигляді зниження концентрації уваги, підвищеної втомлюваності, появи головного болю, зорового перенапруження, порушень опорно-рухового апарату та нервової системи. Тривалий негативний вплив несприятливих факторів здатен призводити до розвитку професійних захворювань та хронічних функціональних розладів [34].

Одним із ключових факторів, що впливає на функціональний стан користувачів комп’ютерів, є зорове навантаження. Під час роботи з відеодисплейними терміналами органи зору перебувають у стані постійної напруги, що пов’язано з необхідністю фокусування погляду на екрані, частими змінами яскравості та контрастності зображення, а також впливом синього спектра світла. Неправильно налаштовані параметри дисплея, низька якість зображення, відблиски та недостатнє або надмірне освітлення робочого місця сприяють розвитку комп’ютерного зорового синдрому. Його основними проявами є сухість очей, печіння, біль, погіршення гостроти зору та швидка втома.

Важливу роль відіграють ергономічні фактори, зокрема поза користувача під час роботи за комп’ютером. Неправильна організація робочого місця, невідповідність висоти столу та крісла, відсутність підтримки спини або неправильне розташування клавіатури й миші призводять до статичного перенапруження м’язів. Тривале перебування у вимушеній позі викликає порушення кровообігу, біль у шийному та поперековому відділах хребта, напруження м’язів плечового поясу та верхніх кінцівок. У перспективі це може спричиняти розвиток остеохондрозу, сколіозу та тунельних синдромів [33].

Не менш значущими є психоемоційні фактори, пов’язані зі специфікою роботи користувачів комп’ютерів. Висока інтелектуальна напруга, необхідність обробки великих обсягів інформації, робота в умовах дефіциту часу, відповідальність за результати діяльності та монотонність операцій негативно впливають на психічний стан працівників. Хронічний стрес, інформаційне перевантаження та відсутність повноцінного відпочинку можуть призводити до емоційного вигорання, підвищеної дратівливості, порушень сну та зниження мотивації до праці.

Суттєвий вплив на функціональний стан мають фактори мікроклімату виробничого середовища. Недотримання нормативних значень температури, вологості та швидкості руху повітря призводить до зниження теплового комфорту та загального самопочуття користувачів. Перегріте або, навпаки, надмірно охолоджене повітря сприяє швидшому розвитку втоми, зниженню концентрації уваги та продуктивності праці. Недостатня вентиляція приміщень також погіршує якість повітря, що негативно впливає на функціонування дихальної та серцево-судинної систем [34].

Додатковим негативним чинником є шум та електромагнітні випромінювання, що супроводжують роботу комп’ютерної техніки та периферійних пристроїв. Хоча рівні шуму та випромінювань зазвичай не перевищують гранично допустимих норм, їх тривалий вплив може сприяти зниженню працездатності, підвищенню втомлюваності та погіршенню загального

самопочуття користувачів. Особливо це актуально для приміщень із великою кількістю робочих місць та недостатньою звукоізоляцією.

Вагомий вплив на функціональний стан користувачів комп'ютерів має режим праці та відпочинку. Відсутність регламентованих перерв, тривала безперервна робота за екраном та недотримання гігієнічних рекомендацій щодо чергування видів діяльності сприяють швидкому виснаженню функціональних резервів організму. Регулярні перерви, виконання виробничої гімнастики та вправ для очей дозволяють знизити негативний вплив шкідливих факторів та підтримувати належний рівень працездатності [35].

Таким чином, функціональний стан користувачів комп'ютерів формується під впливом комплексу фізичних, ергономічних, психоемоційних та організаційних факторів. Забезпечення безпечних і комфортних умов праці, раціональна організація робочого місця, дотримання норм охорони праці та впровадження профілактичних заходів є необхідними умовами збереження здоров'я користувачів комп'ютерної техніки та підвищення ефективності їх професійної діяльності.

ВИСНОВКИ

У магістерській роботі вирішено актуальну науково-практичну задачу архітектурного проєктування та розробки веб-платформи управління бібліотечними процесами на основі відкритих програмних рішень. У процесі виконання роботи було проведено комплексний аналіз предметної області, сучасних бібліотечних інформаційних систем та архітектурних підходів, що використовуються в open-source рішеннях.

У першому розділі роботи здійснено аналіз бібліотечних процесів та інформаційних потоків, розглянуто стандарти бібліографічного опису, зокрема формат MARC21, а також проведено огляд існуючих open-source бібліотечних систем. Виявлено основні переваги та недоліки таких рішень, що дозволило сформувати вимоги до розроблюваної веб-платформи.

Другий розділ присвячено проєктуванню та розробці веб-платформи. Обґрунтовано вибір архітектурного стилю та технологічного стеку, спроєктовано модульну архітектуру системи, базу даних і моделі предметної області. Реалізовано серверну та клієнтську частини веб-платформи, а також забезпечено інтеграцію з MARC-каталогами, DOI-сервісами та Google Books API.

У третьому розділі виконано тестування та оцінку якості розробленої платформи. Проведено модульне, інтеграційне, функціональне та нефункціональне тестування, а також проаналізовано аспекти безпеки та захисту даних. Результати тестування підтвердили коректність роботи основних функціональних модулів та відповідність системи сформованим вимогам.

Отримані результати свідчать про доцільність використання модульної архітектури та відкритих програмних рішень для створення сучасних бібліотечних інформаційних систем. Розроблена веб-платформа може бути використана як основа для впровадження в реальних бібліотеках та для подальших наукових і практичних досліджень у сфері програмної інженерії.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Boyko, I., Petryk, M., Mudryk, I., Stoianov, Y., Tsupryk, H. Mathematical Model of the Capacitor Based on Zeolite Material // Proceedings - International Conference on Advanced Computer Information Technologies, ACIT. – 2021. – С. 45–48.

2. Методичні вказівки до виконання кваліфікаційної роботи магістра для здобувачів спеціальності 121 – Інженерія програмного забезпечення, всіх форм навчання / укладачі: Михалик Д.М., Цуприк Г.Б., Бревус В.М., Мудрик І.Я. – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя, 2024. – 44 с.

3. Бойко І. В., Петрик М. Р., Цуприк Г. Б. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навч. посіб. – Тернопіль: ТНТУ, 2020. – 62 с.

4. Koha Community. Koha Library Software Documentation [Електронний ресурс]. – Режим доступу: https://koha-community.org/documentation/

5. Breeding, M. Open Source Library Systems: Koha and Evergreen // Library Technology Reports. – 2015. – Vol. 51(6). – P. 1–37.

6. Evergreen Project. Evergreen ILS Documentation [Електронний ресурс]. – Режим доступу: https://evergreen-ils.org/documentation/

7. Singh, V. Open Source Integrated Library Systems: Analysis of Koha and Evergreen // Information Studies Journal. – 2019. – Vol. 25(2). – P. 45–56.

8. MARC Standards. Library of Congress [Електронний ресурс]. – Режим доступу: https://www.loc.gov/marc/

9. DOI Handbook. International DOI Foundation [Електронний ресурс]. – Режим доступу: https://www.doi.org/

10. Google Books API Documentation. Google Developers [Електронний ресурс]. – Режим доступу: https://developers.google.com/books

11. Fowler, M. Patterns of Enterprise Application Architecture. – Addison-

Wesley, 2002. – 533 p.

12. Bass, L., Clements, P., Kazman, R. Software Architecture in Practice. – Addison-Wesley, 2013. – 640 p.

13. Kruchten, P. The Rational Unified Process: An Introduction. – Addison-Wesley, 2004. – 320 p.

14. Shaw, M., Garlan, D. Software Architecture: Perspectives on an Emerging Discipline. – Prentice Hall, 1996. – 242 p.

15. Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, P., Stal, M. Pattern-Oriented Software Architecture. – Wiley, 1996. – 476 p.

16. Martin, R. C. Clean Architecture: A Craftsman’s Guide to Software Structure and Design. – Prentice Hall, 2017. – 432 p.

17. Sommerville, I. Software Engineering. – 10th ed. – Pearson, 2015. – 792 p.

18. Pressman, R. S., Maxim, B. R. Software Engineering: A Practitioner’s Approach. – McGraw-Hill, 2019. – 976 p.

19. Booch, G., Rumbaugh, J., Jacobson, I. The Unified Modeling Language User Guide. – Addison-Wesley, 2005. – 482 p.

20. ISO/IEC/IEEE 42010:2011. Systems and software engineering — Architecture description. – ISO, 2011. – 46 p.

21. ISO/IEC 25010:2011. Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE). – ISO, 2011. – 34 p.

22. IEEE Std 1471-2000. IEEE Recommended Practice for Architectural Description of Software-Intensive Systems. – IEEE, 2000. – 24 p.

23. Petryk, M., Mudryk, I., Tsupryk, H. Software Engineering Approaches for Library Information Systems // CEUR Workshop Proceedings. – 2022. – C. 112–118.

24. Boyko, I., Petryk, M., Mudryk, I., Stoianov, Y., Tsupryk, H. Mathematical Model of the Capacitor Based on Zeolite Material // ACIT Proceedings. – 2021. – C. 45–48.

25. Breeding, M. Library Systems Report 2020: Open Source and Cloud Solutions // American Libraries. – 2020. – Vol. 51(5). – P. 24–32.

26. Raymond, E. S. The Cathedral and the Bazaar. – O’Reilly Media, 2001. –

241 p.

27. Spinellis, D. Code Reading: The Open Source Perspective. – Addison-Wesley, 2003. – 512 p.

28. Kitchenham, B., Charters, S. Guidelines for Performing Systematic Literature Reviews in Software Engineering. – EBSE Technical Report, 2007. – 57 p.

29. Glass, R. L. Facts and Fallacies of Software Engineering. – Addison-Wesley, 2003. – 224 p.

30. Open Source Initiative. The Open Source Definition [Електронний ресурс]. – Режим доступу: https://opensource.org/osd

31. Mudryk, I., Petryk, M. Architectural Design of Modular Web Platforms for Library Management // Proceedings of International Conference on Computer Science and Information Technologies. – 2023. – С. 78–84.

32. Желібо Є., Заверуха Н., Зацарний В. Безпека життєдіяльності. — Київ, 2001. — 483 с.

33. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями». — Київ, 2018. URL: https://zakon.rada.gov.ua/laws/show/z0508-18#Text.

34. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної та заочної (дистанційної) форм навчання «БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ» / В.С. Стручок –Тернопіль: ФОП Паляниця В. А., –156 с. Отримано з https://elartu.tntu.edu.ua/handle/lib/39196.

35. Навчальний посібник «ТЕХНОЕКОЛОГІЯ ТА ЦИВІЛЬНА БЕЗПЕКА. ЧАСТИНА «ЦИВІЛЬНА БЕЗПЕКА»» / автор-укладач В.С. Стручок– Тернопіль: ФОП Паляниця В. А., – 156 с. Отримано з http://elartu.tntu.edu.ua/handle/lib/39424

ДОДАТКИ

ДОДАТОК А

Лістинг коду сайту

```
<?php
 include ("include/header.php");
 ?>
<div class="container-fluid">
    <div class="row">
        <!-- Blog Entries Column -->
        <div class="col-md-2"></div>
        <div class="col-md-8">
<h2 class="my-4">Про бібліотеку</h2>
<div class="card mb-6">
                <img class="card-img-top" src="img/qwe.jpg">
                <div class="card-body">
                    <p class="card-text">Навесні  1907 року в с.Вільхівчик почала діяти «Просвіта», при якій була заснована читальня -  прообраз сучасної бібліотеки. Засновником читальні був Туткалюк Пилип та 19 його односельчан, які на той час зібрали 15 корон на придбання книжок і виписку  періодики для читальні. Через деякий час читальня припинила свою діяльність. ЇЇ робота знову відновилася у 1925 році. В 1926році розпочалося будівництво окремого будинку для читальні, яке було завершене в 1929 році. Його вартість була оцінена в 1500 злотих і він був заінтибульований на членів читальні «Просвіти» у Вільхівчику. Крім читальні, у цьому будинку розмістилася українська споживча кооператива «Поміч», яка платила читальні 48 злотих щорічно. В 1938 році читальня мала 125 членів. Членський внесок становив: для старших – 60 грошів, для молодших – 30 грошів. Читальня проводила велику культурно освітню роботу. В неділю, свята та по вечорах у будні дні тут відбувалися голосні читання газет та часописів. Читальня вела постійну роботу по поповненню своїх фондів. В 1938 році фонд становив 332 книги, в цьому ж році літературою скористався 131 чоловік, прочитано 407 книг. В 1940 році в селі було створено колгосп і бібліотека була заснована при колгоспі. В 1952 році бібліотекарем призначена Скотніцька – Деренівська Олександра Мартинівна яка працювала до 1984 року, вийшовши на пенсію.</p>
                </div>
```

```
                </div>
    <script>
          let map;
          function initMap() {
            myMap = new
google.maps.Map(document.getElementById("map"), {
              center: { lat: 49.096232, lng: 26.196987 },
              zoom: 17,
            });
            var marker = new google.maps.Marker(
            {
              position: { lat: 49.096232, lng: 26.196987 },
              map: myMap,
              title: "Ми тут"
            });}
        </script>
    <h2 class="my-4">Ми тут</h2>
                <div id="map" style="height: 400px; width: 100%;">
                </div>
                <script
    src="https://maps.googleapis.com/maps/api/js?key=AIzaSyABFL41mV
zMWit9n0g_0c2cuLGgKD_FyZk&callback=initMap&libraries=&v=weekly"
          async
        ></script>
            </div>
        </div>
    </div>
    <br><br>

    <?php
     include ("include/footer.php");
     ?>
```

```
     <?php
include("include/header.php");
$post_id = $_GET['post_id'];
if (!is_numeric($post_id)) exit();
$post=get_post_by_id($post_id);
?>
<div class="container">
    <div class="row">
        <!-- Post Content Column -->
        <div class="col-lg-16">
            <!-- Title -->
            <h1 class="mt-4"><?=$post['title']?></h1>


            <hr>
            <!-- Date/Time -->
            <p>Опубліковано  <?=$post['datatime']?></p>
            <hr>
            <!-- Preview Image -->
            <img class="img-fluid rounded" src="<?=$post['image']?>"
alt="">
            <hr>
            <!-- Post Content -->
            <p><?=$post['content']?></p>
            <hr>
<?php
include("include/footer.php");
?>
```

```
     <?php
include("include/header.php");
$category_id = $_GET['id'];
if (!is_numeric($category_id)) exit();

$category=get_category_by_id($category_id);
if($category_id==2):
?>
<div class="container-fluid">

    <div class="row">
        <!-- Post Content Column -->
   <!-- Blog Entries Column -->
        <div class="col-md-2"></div>
        <div class="col-md-8">
            <h1 class="my-4">Новини</h1>
            <?php
                $news = get_news();
            ?>
            <!-- Blog Post -->
            <?php foreach ($news as $new):?>
```

```
            <div class="card mb-4">
                <img class="card-img-top"
src="<?=$new['image']?>"style="height: 450px;">
                <div class="card-body">
                    <h2 class="card-title"><?=$new['title']?></h2>
                    <p class="card-
text"><?=mb_substr($new['content'],0, 360, 'UTF-8').'...'?></p>
                    <a href="post.php?post_id=<?=$new["id"]?>"
class=" btn btn-primary">Читати більше &rarr;</a>
                </div>
                <div class="card-footer text-muted">
                    <svg class="bi bi-calendar" width="1em"
height="1em" viewBox="0 0 16 16" fill="currentColor"
xmlns="http://www.w3.org/2000/svg">
                        <path fill-rule="evenodd" d="M14 0H2a2 2 0
00-2 2v12a2 2 0 002 2h12a2 2 0 002-2V2a2 2 0 00-2-2zM1 3.857C1 3.384
1.448 3 2 3h12c.552 0 1 .384 1 .857v10.286c0 .473-.448.857-1
.857H2c-.552 0-1-.384-1-.857V3.857z" clip-rule="evenodd"/>
                        <path fill-rule="evenodd" d="M6.5 7a1 1 0
100-2 1 1 0 000 2zm3 0a1 1 0 100-2 1 1 0 000 2zm3 0a1 1 0 100-2 1 1
0 000 2zm-9 3a1 1 0 100-2 1 1 0 000 2zm3 0a1 1 0 100-2 1 1 0 000
2zm3 0a1 1 0 100-2 1 1 0 000 2zm3 0a1 1 0 100-2 1 1 0 000 2zm-9 3a1
1 0 100-2 1 1 0 000 2zm3 0a1 1 0 100-2 1 1 0 000 2zm3 0a1 1 0 100-2
1 1 0 000 2z" clip-rule="evenodd"/>
                    </svg>
                    Опубліковано <?=$new['datatime']?>
                </div>
            </div>
            <?php endforeach; ?>
 <?php endif; ?>
<?php

$category_id = $_GET['id'];
if (!is_numeric($category_id)) exit();

$category=get_category_by_id($category_id);
if($category_id==3):
?>
<div class="container-fluid">
    <div class="row">
        <div class="col-md-2"></div>
        <div class="col-md-8">
            <div class="row">
                <div class="col-md-3"><h1 class="my-
3">Книги</h1></div>
            <div class="col-md-3"></div>
            <div class="col-md-6" style="margin-top: 5px;">
                <div class="my-4"><div><form method="POST"
action="search.php">  <input type="text" name="text"
```

```
placeholder="Пошук" value= "<?=$_COOKIE['name']?>" required> <input
type="submit" name="enter" value="Пошук"></form></div>

                </div></div></div>


            <?php
                $books = get_books();
            ?>
            <div class="card mb-4">
            <?php foreach ($books as $book):?>
                <div class="row">
                    <div class="card mb-1" style="margin-left:
15px"><img class="card-img-top" src="<?=$book['img']?>"
style="height: 450px; width: 270px;"></div>
                    <div class="card-body" style="width: 40%;">
                    <h4 class="card-title"><?=$book['name']?></h4>
                    <p class="card-text">Автор:
<?=$book['avtor']?></p>
                    <p class="card-text">Рік випуску:
<?=$book['rik_vypusku']?></p>
                    <p class="card-text">Видавництво:
<?=$book['publication']?></p>
                    <p class="card-text">Кількість сторінок:
<?=$book['kil_stor']?></p>
                    <p class="card-text">Кількість книг:
<?=$book['kil_books']?></p>
                     <?php
            if(empty($_COOKIE['user'])):
                ?>
                <p class="card-text">Авторизуйтеся щоб читати</p>
            <?php else:?>
                    <a href="<?=$book["text"]?>" download="" class="
btn btn-primary">Завантажити</a>
                </div>
                 <?php endif;?>
            </div>

            <hr>
            <?php endforeach; ?>
 <?php endif; ?>
                </div>

 <?php
 include ("include/footer.php");
 ?>

     <!DOCTYPE html>
<html lang="uk">
<head>
```

```
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Адмін Панель</title>
</head>
<body>

  <?php
        if(empty($_COOKIE['admin'])):
            ?>
            <?php  for($n=1; $n<1000;$n++){?>
             ти не адмін
          <? }?>
            <?php else:?>
<h1>Адмін Панель</h1>
<a href="index.php">На головну сторінку</a>
<?php
$servername = "localhost";
    $username = "root";
    $pas = "root";
    $dbname = "biblioteka";

    $conn = mysqli_connect($servername,$username,$pas,$dbname);
    if (mysqli_connect_errno()) {
        echo 'Помилка підключення до БД:
('.mysqli_connect_errno().'):'.mysqli_connect_error();
        exit();
    }
 ?>

<?php


include ("del/del_regist.php");
include ("del/del_news.php");
include ("del/del_books.php");
include ("del/del_categorias.php");
include ("del/del_janr.php");
include ("del/del_coment.php");

if (isset($_POST["title"])) {

      if (isset($_GET['red_id'])) {
          $sql = mysqli_query($conn, "UPDATE `news` SET `title` =
'{$_POST['title']}',`image` = '{$_POST['image']}'
            ,`content` = '{$_POST['content']}' ,`datatime` =
'{$_POST['datatime']}' WHERE `id`={$_GET['red_id']}");
      } else {
```

```
          $sql = mysqli_query($conn, "INSERT INTO `news` (`title`,
`image`,`content`,`datatime`) VALUES ('{$_POST['title']}',
            '{$_POST['image']}' , '{$_POST['content']},
'{$_POST['datatime']}')");
      }


      if ($sql) {
        echo '<p>Успешно!</p>';
      } else {
        echo '<p>Произошла ошибка: ' . mysqli_error($conn) . '</p>';
      }
    }

if (isset($_GET['red_id'])) {
      $sql = mysqli_query($conn, "SELECT `id`, `title`,
`image`,`content`,`datatime` FROM `news` WHERE
`id`={$_GET['red_id']}");
      $news = mysqli_fetch_array($sql);
    }

  if (isset($_GET['del_news_id'])) {

    $sql = mysqli_query($conn, "DELETE FROM `news` WHERE `id` =
{$_GET['del_news_id']}");
    if ($sql) {
      echo "<p>Користувач видалений.</p>";
    } else {
      echo '<p>Сталася помилка: ' . mysqli_error($conn) . '</p>';
    }
  }

  if (isset($_GET['del_news_id'])) {
    $sql = mysqli_query($conn, "DELETE FROM `news` WHERE `id` =
{$_GET['del_news_id']}");
    if ($sql) {
      echo "<p>Користувач видалений.</p>";
    } else {
      echo '<p>Сталася помилка: ' . mysqli_error($conn) . '</p>';
    }
  }

?>
 <h2>Таблиця користувачів</h2>
    <table border='1'>
  <tr>
    <td>Номер</td>
    <td>Логін</td>
    <td>Пароль</td>
```

```
    <td>Вид користувача</td>
  </tr>
  <?php
    $sql = mysqli_query($conn, 'SELECT `id`, `login`,
`pass`,`name`,`admin` FROM `regist`');
    while ($result = mysqli_fetch_array($sql)) {


      echo

            "<tbody>".
        '<tr>' .
             "<td>{$result['id']}</td>" .
             "<td>{$result['login']}</td>".
             "<td>{$result['pass']}</td>" .
             "<td>{$result['name']}</td>" .
             "<td>{$result['admin']}</td>".
              "<td><a
href='?del_regist_id={$result['id']}'>Видалити</a></td>".
        '</tr>'.
        "</tbody>";

      }

  ?>
</table>

<h3>Форма додавання</h3>
<form action="add/add_regist.php" method="post">
  <table>
    <tr>
      <td>Логін:</td>
      <td><input type="text" name="login" id="title"></td>
    </tr>
    <tr>
      <td>Пароль
      <td><input type="text" name="pass" id="image"> </td>
    </tr>
    <tr>
      <td>Імя
      <td><input type="text" name="name" id="name"> </td>
    </tr>
    <tr>
      <td>Адмін
      <td><input type="text" name="admin" id="admin"> </td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" value="OK"></td>
    </tr>
```

```
  </table>
</form>

 <h2>Таблиця новин</h2>
    <table border='1'>
  <tr>
    <td>Номер</td>
    <td>Заголовок</td>
    <td>Зображення</td>
    <td>Зміст</td>
    <td>Дата</td>
  </tr>
  <?php
    $sql = mysqli_query($conn, 'SELECT `id`, `title`,
`image`,`content`,`datatime` FROM `news`');
    while ($result = mysqli_fetch_array($sql)) {


      echo
      "<tbody>".
        '<tr>' .
             "<th scope='row'>{$result['id']}</td>" .
             "<td>{$result['title']}</td>" .
             "<td>{$result['image']}</td>" .
             "<td>{$result['content']}</td>" .
             "<td>{$result['datatime']}</td>" .
             "<td><a
href='?del_news_id={$result['id']}'>Видалити</a></td>" .
             "<td><a
href='?red_id={$result['id']}'>Редагувати</a></td>" .
        '</tr>'.
        "</tbody>";
      }

  ?>
</table>
<h3>Форма змін</h3>
<form action="" method="post">
    <table>
      <tr>
        <td>Заголовок:</td>
        <td><input type="text" name="title" value="<?=
isset($_GET['red_id']) ? $news['title'] : ''; ?>"></td>
      </tr>
      <tr>
        <td>Зображення:</td>
        <td><input type="text" name="image"  value="<?=
isset($_GET['red_id']) ? $news['image'] : ''; ?>"></td>
      </tr>
```

```
      <tr>
        <td>Зміст:</td>
        <td><input type="text" name="content"  value="<?=
isset($_GET['red_id']) ? $news['content'] : ''; ?>"></td>
      </tr>
      <tr>
        <td>Дата:</td>
        <td><input type="text" name="datatime" value="<?=
isset($_GET['red_id']) ? $news['datatime'] : ''; ?>"></td>
      </tr>
      <tr>
        <td colspan="2"><input type="submit" value="OK"></td>
      </tr>
    </table>
  </form>

<h3>Форма додавання</h3>
<form action="add/add_news.php" method="post">
  <table>
    <tr>
      <td>Заголовок:</td>
      <td><input type="text" name="title" id="title"></td>
    </tr>
    <tr>
      <td>Зображення
      <td><input type="text" name="image" id="image"> </td>
    </tr>
    <tr>
      <td>Зміст
      <td><input type="text" name="content" id="content"> </td>
    </tr>
    <tr>
      <td>Дата
      <td><input type="text" name="datatime" id="datatime"> </td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" value="OK"></td>
    </tr>
  </table>
</form>

<h2>Таблиця категорій</h2>
    <table border='1'>
  <tr>
    <td>Номер</td>
    <td>Назва</td>

  </tr>
  <?php
```

```
    $sql = mysqli_query($conn, 'SELECT `id`, `title` FROM
`categories`');
    while ($result = mysqli_fetch_array($sql)) {


      echo  "<tbody>".
        '<tr>' .
             "<th scope='row'>{$result['id']}</td>" .
             "<td>{$result['title']}</td>" .
             "<td><a
href='?del_category_id={$result['id']}'>Видалити</a></td>" .

        '</tr>'.
        "</tbody>";


      }

  ?>
</table>

<h3>Форма додавання</h3>
<form action="add/add_categories.php" method="post">
  <table>
    <tr>
      <td>Назва:</td>
      <td><input type="text" name="title" id="title"></td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" value="OK"></td>
    </tr>
  </table>
</form>

<h2>Таблиця книг</h2>
    <table border='1'>
  <tr>
    <td>Номер</td>
    <td>Назва</td>
    <td>Автор</td>
    <td>Рік випуску</td>
    <td>Видавництво</td>
    <td>Кількість сторінок</td>
    <td>Кількість книг</td>
    <td>Номер жанру</td>
    <td>Зображення</td>
    <td>Текст</td>
  </tr>
  <?php
```

```
    $sql = mysqli_query($conn, 'SELECT `id`, `name`,
`avtor`,`rik_vypusku`,`publication`,`kil_stor`,`kil_books`,`id_janr`
,`img`,`text` FROM `books`');
    while ($result = mysqli_fetch_array($sql)) {


      echo
      "<tbody>".
        '<tr>' .
             "<th scope='row'>{$result['id']}</td>" .
             "<td>{$result['name']}</td>" .
             "<td>{$result['avtor']}</td>" .
             "<td>{$result['rik_vypusku']}</td>" .
             "<td>{$result['publication']}</td>" .
             "<td>{$result['kil_stor']}</td>" .
             "<td>{$result['kil_books']}</td>" .
             "<td>{$result['id_janr']}</td>" .
             "<td>{$result['img']}</td>" .
             "<td>{$result['text']}</td>" .
             "<td><a
href='?del_books_id={$result['id']}'>Видалити</a></td>" .

        '</tr>'.
        "</tbody>";


      }

  ?>
</table>

<h3>Форма додавання</h3>
<form action="add/add_books.php" method="post">
  <table>
    <tr>
      <td>Назва:</td>
      <td><input type="text" name="name" id="name"></td>
    </tr>
    <tr>
      <td>Автор
      <td><input type="text" name="avtor" id="avtor"> </td>
    </tr>
    <tr>
      <td>Рік випуску
      <td><input type="text" name="rik_vypusku" id="rik_vypusku">
</td>
    </tr>
    <tr>
      <td>Видавництво
```

```
      <td><input type="text" name="publication" id="publication">
</td>
    </tr>
    <tr>
      <td>Кількість сторінок
      <td><input type="text" name="kil_stor" id="kil_stor"> </td>
    </tr>
    <tr>
      <td>Кількість книг
      <td><input type="text" name="kil_books" id="kil_books"> </td>
    </tr>
    <tr>
      <td>Номер жанру
      <td><input type="text" name="id_janr" id="id_janr"> </td>
    </tr>
    <tr>
      <td>Зображення
      <td><input type="text" name="img" id="img"> </td>
    </tr>
    <tr>
      <td>Текст
      <td><input type="text" name="text" id="text"> </td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" value="OK"></td>
    </tr>
  </table>
</form>
<h2>Таблиця жанрів</h2>
    <table border='1'>
  <tr>
    <td>Номер</td>
    <td>Назва</td>

  </tr>
  <?php
    $sql = mysqli_query($conn, 'SELECT `id`, `name` FROM `janr`');
    while ($result = mysqli_fetch_array($sql)) {

      echo
      "<tbody>".
        '<tr>' .
             "<th scope='row'>{$result['id']}</td>" .
             "<td>{$result['name']}</td>" .
             "<td><a
href='?del_janr_id={$result['id']}'>Видалити</a></td>" .

        '</tr>'.
        "</tbody>";
```

```
      }

  ?>
</table>
<h3>Форма додавання</h3>
<form action="add/add_janr.php" method="post">
  <table>
    <tr>
      <td>Назва:</td>
      <td><input type="text" name="name" id="name"></td>
    </tr>
    <tr>
      <td colspan="2"><input type="submit" value="OK"></td>
    </tr>
  </table>
</form>
<h2>Таблиця відгуків</h2>
    <table border='1'>
  <tr>
    <td>Номер</td>
    <td>Відгук</td>
    <td>Імя</td>


  </tr>
  <?php
    $sql = mysqli_query($conn, 'SELECT `id`, `text`, `name` FROM
`coment`');
    while ($result = mysqli_fetch_array($sql)) {


      echo "<tbody>".
        '<tr>' .
             "<td>{$result['id']}</td>" .
             "<td>{$result['text']}</td>".
             "<td>{$result['name']}</td>" .
              "<td><a
href='?del_coment_id={$result['id']}'>Видалити</a></td>".
        '</tr>'.
        "</tbody>";
      }

  ?>
</table>
<?php endif;?>

</body>
</html>
```

ДОДАТОК Б

Тези

УДК 004.41

М. П. Бесащук, В. М. Бревус, PhD.

Тернопільський національний технічний університет імені Івана Пулюя, Україна

АРХІТЕКТУРНЕ ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБ-ПЛАТФОРМИ ДЛЯ УПРАВЛІННЯ БІБЛІОТЕЧНИМИ ПРОЦЕСАМИ НА ОСНОВІ ВІДКРИТИХ ПРОГРАМНИХ РІШЕНЬ.

M. P. Besashchuk, V. M. Brevus, PhD.

ARCHITECTURAL DESIGN AND DEVELOPMENT OF A WEB PLATFORM FOR LIBRARY PROCESSES MANAGEMENT BASED ON OPEN SOFTWARE SOLUTIONS.

У сучасних умовах цифровізації освіти та науки бібліотеки відіграють важливу роль як інформаційні центри доступу до знань. Ефективне управління бібліотечними процесами потребує використання сучасних веб-технологій та інтеграції з міжнародними інформаційними ресурсами. Саме тому актуальним є завдання створення веб-платформ на основі відкритих програмних рішень.

Проблема автоматизації бібліотечних процесів полягає у необхідності обробки великих обсягів бібліографічних даних, забезпеченні швидкого доступу до електронних каталогів, підтримці стандартів опису документів і взаємодії з зовнішніми сервісами. Існуючі комерційні системи часто є дорогими та закритими для модифікації, що обумовлює доцільність використання open-source рішень.

У роботі проведено аналіз та адаптацію відкритих архітектур бібліотечних систем, зокрема Koha, Evergreen та VuFind. Визначено їх основні переваги, серед яких модульність, масштабованість, підтримка стандартів MARC та можливість інтеграції через API. На основі цього запропоновано власну модульну архітектуру веб-платформи, що забезпечує гнучке розширення функціональності та незалежність окремих компонентів.

Головні особливості запропонованої платформи:

- Модульність – розподіл системи на незалежні функціональні компоненти.
- Інтеграція – підтримка MARC-каталогів, DOI та API Google Books.
- Масштабованість – можливість розширення системи без зміни її ядра.
- Доступність – використання через веб-браузер з будь-якого пристрою.

Для розробки та проєктування використано такі технології:

- Open-source фреймворки для серверної та клієнтської частини.
- REST API для обміну даними між модулями.
- Формати MARC21 для обміну бібліографічними записами.
- API Google Books для отримання анотацій, зображень та метаданих книг.
- DOI-сервіси для автоматизованого наповнення наукових публікацій.

У результаті розроблено архітектурну модель веб-платформи для управління

бібліотечними процесами, що забезпечує автоматизацію каталогізації, пошуку, обліку та інтеграції з міжнародними бібліографічними джерелами. Система дозволяє значно підвищити ефективність роботи бібліотеки та якість обслуговування користувачів.

Практична значущість роботи полягає у можливості використання розробленої платформи в навчальних, наукових та публічних бібліотеках для створення сучасного електронного інформаційного середовища.

Таким чином, запропоноване архітектурне рішення відповідає сучасним вимогам до інформаційних систем, базується на відкритих технологіях та має високий потенціал для подальшого розвитку.

Література

1. Петрик М.Р., Михалик Д.М., Петрик О.Ю., Цуприк Г.Б. Методичні вказівки до виконання атестаційної роботи магістра за спеціальністю 121 – «Інженерія програмного забезпечення» для усіх форм навчання – Тернопіль: Тернопільський національний технічний університет імені Івана Пулюя – 2020 – 27 с.
2. Koha Open Source Integrated Library System. Official Documentation. – URL: https://koha-community.org
3. MARC 21 Format for Bibliographic Data. Library of Congress. – URL: https://www.loc.gov/marc/
4. Бойко І.В., Петрик М.Р., Цуприк Г.Б. Моделювання та видобуток даних (високопродуктивні обчислення у великих та числових системах, комбінаторному аналізі): навчальний посібник. Тернопіль: : ТНТУ 2019 – 62 с.
5. Бойко І.В., Петрик М.Р., Цуприк Г.Б. Інформаційні технології видобутку даних (Data mining, високопродуктивні обчислення у складних системах): навчальний посібник. Тернопіль: : ТНТУ 2020 – 62 с.