

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проєктами
Кафедра інформаційних управляючих систем та технологій

"Допущений до захисту"

Завідувач кафедри ІУСТ

к.т.н, доц. Михелєв І.Л.

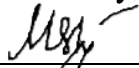
" ____ " ____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти "магістр"

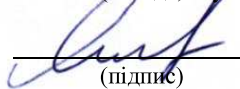
на тему: Дослідження та розробка ІС управління фондами друкованих і
цифрових видань у бібліотечно-репозитарній системі ЗВО

Виконав: студент групи 6141 мз

 **Горелко М.С.**
(підпис)

Керівник роботи:

доцент кафедри ІУСТ, к.т.н.,
(посада, науковий ступінь вчене звання)

 **Михелєв І.Л.**
(підпис)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова
Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра інформаційних управляючих систем та технологій
Спеціальність 122 "Комп'ютерні науки"
Освітня програма "Комп'ютерні науки"

"ЗАТВЕРДЖУЮ"

Гарант освітньої програми

В. Ажищев

к.т.н, доц. Ажищев В.Ф.

"___" _____ 2025 р.

ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ на здобуття ступеня вищої освіти "магістр"

Студенту Горелко Михайло Сергійович

(прізвище, ім'я, по батькові)

1. Тема роботи: Дослідження та розробка ІС управління фондами друкованих і цифрових видань у бібліотечно-репозитарній системі ЗВО

Керівник роботи к.т.н., доцент Михелєв І.Л.

Затверджені наказом ректора № 1355-уч. від "14" листопада 2025 року.

2. Термін подання роботи: 15 грудня 2025 р.

3. Вихідні дані до проєкту (роботи) ДСТУ щодо обробки інформації, літературні джерела, технічна документація на існуючі аналоги систем, матеріали наукового стажування

4. Перелік питань, що належать до розробки (найменування розділів)

Дослідження підсистеми безпечного віддаленого доступу до корпоративної інформаційної системи. Проєктування інформаційної підсистеми безпечного віддаленого доступу до корпоративної інформаційної системи. Розробка інформаційної підсистеми безпечного віддаленого доступу до корпоративної інформаційної системи. Техніко-економічне обґрунтування розробки. Охорона праці. Охорона навколишнього середовища

5. Перелік презентаційних матеріалів Недоліки роботи, постановка задачі, задачі дослідження, концепція підсистем безпечного віддаленого доступу до корпоративної інформаційної системи, Діаграма варіантів використання, Концептуальна модель даних, Діаграма послідовності, Фізичну структуру бази даних системи, Схема взаємодії модулів програмного забезпечення системи

6. Консультанти розділів роботи

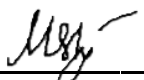
Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Михелєв І.Л., доцент	05.09.2024	22.12.2024
2	Михелєв І.Л., доцент	23.12.2024	15.06.2025
3	Михелєв І.Л., доцент	15.06.2025	25.09.2025
4	Михелєв І.Л., доцент	25.09.2025	14.11.2025
5		25.09.2025	14.11.2025
6		25.09.2025	14.11.2025

7. Дата видачі завдання 05 вересня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

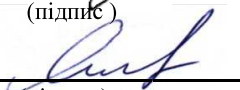
№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів роботи	Примітка
1	Дослідження та аналіз предметної галузі	22.12.2024	
2	Розробка концепції	15.03.2025	
3	Розділ з розробки проєктних рішень по системі (та її частинах)	15.06.2025	
4	Реалізація проєкту та розробка робочої документації	25.09.2025	
5	Обчислення економічної ефективності	14.11.2025	
6	Вирішення питань охорони праці та навколишнього середовища	14.11.2025	
7	Оформлення пояснювальної записки	14.11.2025	
8	Подання магістерської роботи на перевірку на плагіат	16.11.2025	
9	Подання магістерської роботи рецензенту	20.11.2025	
10	Подання магістерської роботи на державний захист	15.12.2025	

Студент


(підпис)

Горелко М.С.
(ПІБ)

Керівник роботи


(підпис)

Михелєв І.Л.
(ПІБ)

АНОТАЦІЯ

Актуальність магістерської роботи зумовлена потребою модернізації бібліотечних інформаційних систем, переходом на електронні каталоги та необхідністю уніфікації бібліографічних описів відповідно до міжнародних стандартів MARC та Dublin Core. У сучасних умовах цифровізації особливо важливим є створення ефективних веб-систем обліку та збереження бібліотечних ресурсів, що забезпечують швидку обробку даних, зручний пошук, масштабованість та інтеграцію з іншими інформаційними сервісами.

Мета і задачі кваліфікаційної роботи: розроблення веб-орієнтованої інформаційної системи уніфікації та обліку бібліотечних ресурсів на основі фреймворку Django з використанням стандартів MARC, що забезпечує автоматизацію процесів каталогізації, зберігання та обробки бібліотечних даних.

Об'єкт дослідження: процеси створення, організації та обробки бібліотечних інформаційних систем.

Предмет дослідження: методи, моделі та технології розробки веб-систем для обліку бібліотечних ресурсів, а також застосування стандартів MARC і хмарних технологій.

Методи дослідження: аналіз та узагальнення наукових джерел, моделювання бізнес-процесів, UML-проектування, побудова ER-моделей, системний аналіз, моделювання архітектури клієнт–сервер, експериментальне тестування створеної системи, порівняльний аналіз програмних технологій.

Наукова новизна отриманих результатів полягає в розробці уніфікованої структури бібліотечних метаданих та створенні інформаційної системи, що забезпечує автоматичну обробку каталогів, інтеграцію MARC-формату та оптимізацію процесів зберігання даних.

Загальний обсяг роботи: XX с., X рис., X додатків, X бібліографічних джерел.

Ключові слова: бібліотека, MARC, каталогізація, Django, інформаційна система, база даних, веб-додаток, метадані.

SUMMARY

The relevance of the master's thesis is determined by the need to modernize library information systems, transition to electronic catalogues, and unify bibliographic descriptions in accordance with international MARC and Dublin Core standards. In the era of digitalization, it is essential to develop efficient web-based systems for the management and processing of library resources, ensuring fast data handling, convenient search, scalability, and integration with external services.

Purpose and objectives of the qualification work: develop a Django-based web information system for unifying and managing library resources using MARC standards, providing automation of cataloguing, data storage, and processing.

Object of research: processes of creation, organization, and operation of library information systems.

Subject of research: methods, models, and technologies for developing web systems for managing library resources, including MARC standards and cloud technologies.

Research methods: include analysis and synthesis of scientific sources, system analysis, UML modelling, ER-modelling, client-server architecture modelling, experimental testing of the developed system, and comparative analysis of software technologies.

The scientific novelty of the results lies in the development of a unified structure of library metadata and the creation of an information system that enables automated processing of catalogues, MARC integration, and optimization of data-storage workflows.

Total volume: XX pages, X figures, X appendices, X sources.

List of keywords: library, MARC, cataloguing, Django, information system, database, web application, metadata.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ.....

ВСТУП

РОЗДІЛ 1. ДОСЛІДЖЕННЯ МОДЕЛЕЙ УПРАВЛІННЯ ФОНДАМИ ДРУКОВАНИХ І ЦИФРОВИХ ВИДАНЬ У БІБЛІОТЕЧНО- РЕПОЗИТАРНІЙ СИСТЕМІ ЗВО

1.1 Аналіз предметної області та існуючих підходів

1.1.1 Сутність, роль та призначення бібліотечно-репозитарних фондів ЗВО

1.1.2 Проблеми та виклики управління фондами ЗВО

1.1.3 Аналіз існуючої бібліотечно-репозитарної системи НУК

1.1.4 Недоліки існуючої моделі та обґрунтування актуальності

1.2 Вибір концепції та технологічного забезпечення

1.2.1 Аналіз технологій та підходів до розробки інформаційних систем

1.2.2 Обґрунтування вибору Python + Django + PostgreSQL

1.2.3 Аналіз форматів метаданих

1.2.4 Вибір архітектури інформаційної системи

1.3 Математичне забезпечення інформаційної системи

1.3.1 Формалізація бібліотечно-репозитарної системи ЗВО

1.3.2 Математична модель метаданих

1.3.3 Модель уніфікації фондів

1.3.4 Математична модель процесу пошуку

1.3.5 Математична модель синхронізації фондів

1.3.6 Математична модель взаємодії користувачів із системою

1.3.7 ER-модель (концептуальна)

1.3.8 Формальна постановка задачі

Висновок до розділу 1

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА УНІФІКАЦІЇ БІБЛІОТЕЧНИХ РЕСУРСІВ

2.1 Аналіз предметної області

2.2 Аналіз існуючих бібліотечних систем

2.3 Формування вимог до розроблюваної системи

2.3.1 Функціональні вимоги

2.3.2 Нефункціональні вимоги

2.4 Проєктування загальної архітектури

2.4.1 Архітектурні рівні

2.4.2 Структурна схема системи

2.5 Проектування бази даних

2.5.1 Основні сутності

2.5.2 ER-діаграма структури бази даних

2.6 UML-діаграми

2.6.1 Use Case діаграма

2.6.2 Activity діаграми

2.7 Логічна модель даних

2.8 UML Sequence — послідовність взаємодії користувача із системою при пошуку та завантаженні ресурсу

2.8.1 Учасники послідовності (lifelines)

2.8.2 Сценарій А — Пошук ресурсу: покрокова послідовність

2.8.3 Сценарій В — Завантаження цифрового файлу (streaming)

2.8.4 HTTP / REST API — повний перелік endpoint'ів для сценаріїв

2.8.5 Транзакції, блокування і консистентність даних

2.8.6 Транзакції, блокування і консистентність даних

2.8.7 Помилки, повтори та idempotency

2.8.8 Безпека та права доступу

2.8.9 Приклад повної послідовності

2.9 Інфологічна модель даних (ER-діаграма інформаційної системи)

2.9.1 Призначення та вимоги до моделі даних

2.9.2 Основні сутності системи

2.9.3 Опис ключових сутностей

2.9.4 Типи зв'язків

2.9.5 Нормалізація

2.9.6 Ключові індекси для продуктивності

2.7 Специфікація REST API

Висновок до розділу 2

РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Вибір засобів і середовища розроблення

3.2 Архітектура програмного комплексу

3.3 Реалізація моделі даних

3.3.1 Модель UnifiedRecord

3.3.2 Модель Author

3.3.3 Модель Keyword

3.3.4 Модель BookPrint

3.3.5 Модель FileResource

3.3.6 Модель BookDigital

3.3.7 Модель користувача

3.4 Реалізація REST API

3.4.1 Серіалізатори

3.4.2 API для пошуку

3.4.3 Маршрутизація

3.5 Реалізація повнотекстового пошуку

3.6 Реалізація модуля синхронізації

3.7 Веб-інтерфейс

Висновки до розділу 3

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Структура програмного забезпечення

4.2 Реалізація модуля роботи з базою даних

4.2.1 Організація зберігання бібліографічних даних

4.2.2 Реалізація механізмів транзакцій та цілісності даних

4.3 Реалізація модуля уніфікації бібліотечних записів

4.3.1 Алгоритм уніфікації метаданих MARC та Dublin Core

4.3.2 Усунення дублювання та синхронізація записів

4.4 Реалізація модуля пошуку та індексації

4.4.1 Реалізація структурного пошуку

4.4.2 Реалізація повнотекстового пошуку

4.4.3 Ранжування та фільтрація результатів

4.5 Реалізація модуля керування доступом та безпекою

4.5.1 Аутентифікація та авторизація користувачів

4.5.2 Рольова модель доступу (RBAC)

4.6 Реалізація веб-інтерфейсу користувача

4.6.1 Інтерфейс користувача для пошуку ресурсів

4.6.2 Інтерфейс адміністратора та бібліотекаря

4.7 Тестування програмної реалізації

4.7.1 Функціональне тестування

4.7.2 Навантажувальне тестування

Висновки до розділу 4

РОЗДІЛ 5. ОЦІНКА ЕФЕКТИВНОСТІ ТА ПЕРСПЕКТИВИ ВПРОВАДЖЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

5.1 Оцінка функціональної ефективності інформаційної системи

5.2 Аналіз продуктивності та надійності системи

5.3 Порівняльний аналіз із існуючими бібліотечними інформаційними системами

5.4 Економічне обґрунтування впровадження інформаційної системи

5.5 Перспективи розвитку та масштабування інформаційної системи

Висновки до розділу 5

РОЗДІЛ 6. ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

6.1 Аналіз умов праці при розробленні та експлуатації інформаційної системи

6.2 Потенційні небезпечні та шкідливі фактори при роботі з

комп'ютерною технікою

6.3 Вимоги до організації робочого місця розробника

6.4 Заходи з охорони праці та техніки безпеки

6.5 Пожежна безпека та дії у надзвичайних ситуаціях

6.6 Вимоги електробезпеки та санітарно-гігієнічні норми

Висновки до розділу 6

ВИСНОВКИ

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API	Application Programming Interface – програмний інтерфейс застосунків
БД	база даних
ВЕБ-ресурс	набір веб-сторінок, доступних через мережу Інтернет
Django	Python-фреймворк для веб-розробки
ER-модель	модель «сутність-зв'язок»
JSON	JavaScript Object Notation – формат обміну даними
MARC	MAchine-Readable Cataloging – машиночитаний формат бібліографічних записів
ORM	Object-Relational Mapping – об'єктно-реляційне відображення
ПК	персональний комп'ютер
ПЗ	програмне забезпечення
REST	архітектурний стиль веб-сервісів
SQL	Structured Query Language – мова управління базами даних
URL	Uniform Resource Locator – уніфікований локатор ресурсу
UX	User Experience – досвід користувача
UI	User Interface – інтерфейс користувача
HTTP	протокол передавання гіпертексту

ВСТУП

Актуальність магістерської роботи зумовлена стрімким розвитком інформаційних технологій та необхідністю модернізації бібліотечних інформаційних систем у сучасних умовах цифровізації. Бібліотеки вищих навчальних закладів потребують оновлення своїх інформаційних платформ, зокрема для ефективного управління та обліку як традиційних, так і електронних ресурсів. У зв'язку з цим важливою є проблема інтеграції існуючих бібліотечних інформаційних систем для забезпечення уніфікованого доступу до різних видів ресурсів.

Метою даної роботи є розробка концепції, архітектури та програмної реалізації інтегрованої інформаційної системи для обліку та збереження бібліотечних ресурсів в умовах цифровізації, що забезпечить автоматизовану каталогізацію, обробку і зберігання як друкованих, так і електронних видань, а також забезпечить доступ до них через єдиний інтерфейс.

Для досягнення цієї мети необхідно вирішити низку задач:

Провести аналіз предметної області та виявити проблеми існуючих рішень у бібліотеках вищих навчальних закладів.

Розробити архітектуру та логічну модель системи для обліку бібліотечних ресурсів.

Описати методи та технології, які будуть використані для інтеграції різних систем та забезпечення їх взаємодії.

Реалізувати програмну частину системи на базі сучасного фреймворку Django та бази даних PostgreSQL.

Провести тестування розробленої системи та оцінити її ефективність.

Виконати економічне обґрунтування впровадження системи в реальних умовах.

Об'єктом дослідження є процеси створення, організації та обробки бібліотечних інформаційних систем. Предметом дослідження є методи, моделі та технології розробки інтегрованої інформаційної системи для обліку бібліотечних ресурсів.

Наукова новизна отриманих результатів полягає у створенні уніфікованої структури для управління метаданими бібліотечних ресурсів та розробці інформаційної системи, що забезпечує автоматичну обробку каталогів, інтеграцію MARC-формату та оптимізацію процесів зберігання даних.

Практичне значення роботи полягає в тому, що створена система може бути впроваджена у вищих навчальних закладах для автоматизації процесів обліку та зберігання бібліотечних ресурсів, що дозволить значно зменшити трудові витрати та підвищити ефективність роботи бібліотек.

Загальний обсяг роботи: XX с., X рис., X додатків, X бібліографічних джерел.

Ключові слова: бібліотека, MARC, каталогізація, Django, інформаційна система, база даних, веб-додаток, метадані.

1. ДОСЛІДЖЕННЯ МОДЕЛЕЙ УПРАВЛІННЯ ФОНДАМИ ДРУКОВАНИХ І ЦИФРОВИХ ВИДАНЬ У БІБЛІОТЕЧНО- РЕПОЗИТАРНІЙ СИСТЕМІ ЗВО

1.1 Аналіз предметної області та існуючих підходів

Інформаційне забезпечення діяльності закладів вищої освіти (ЗВО) сьогодні охоплює значно ширший спектр ресурсів, ніж традиційні бібліотечні фонди. Окрім друкованих видань, які історично були основою навчального процесу, ключову роль відіграють цифрові інформаційні ресурси — електронні книги, методичні матеріали, наукові статті, дипломні та дисертаційні роботи, мультимедійні матеріали.

У зв'язку із цифровізацією освітнього простору виникає потреба у побудові єдиної інтегрованої інформаційної системи (ІС), яка забезпечує:

- уніфікований доступ до всіх ресурсів;
- централізоване управління метаданими;
- синхронізацію між друкованими та цифровими фондами;
- підтримку стандартизованих бібліотечно-інформаційних форматів;
- автоматизацію роботи бібліотекарів, науковців і студентів.

Сучасні бібліотечно-репозитарні системи ЗВО виконують дві фундаментально різні функції:

- Облік і управління друкованими фондами
- Зберігання, поширення та індексація цифрових ресурсів

У більшості університетів (зокрема в НУК) ці системи реалізовані на різних технологічних платформах, що утворює низку організаційних, інформаційних і технічних проблем.

1.1.1 Сутність, роль та призначення бібліотечно-репозитарних фондів ЗВО

Бібліотечно-репозитарна система є одним з ключових структурних елементів інформаційної інфраструктури сучасного закладу освіти. Вона реалізує управління двома видами фондів:

1. Друкований фонд

До друкованих фондів належать:

- навчальні посібники;
- підручники;
- наукові журнали;
- методичні матеріали;

довідкова література.

Особливості управління друкованими фондами:

- облік кожного примірника;
- фізична інвентаризація;
- маркування штрих-кодами;
- ведення формулярів;
- контроль видач/повернень;
- списання зношених або застарілих видань.

2. Цифровий фонд (репозитарій)

Цифровий фонд охоплює:

- електронні копії друкованих видань;
- дисертації, дипломні роботи;
- наукові статті викладачів;
- електронні навчальні курси;
- мультимедійні матеріали;
- презентації, лабораторні роботи тощо.

Цифрові репозитарії виконують функції:

- довготривалого зберігання;
- відкритого доступу (Open Access)
- індексації матеріалів у Google Scholar;
- забезпечення академічної доброчесності;
- підтримки протоколів OAI-PMH (Harvesting).

Роль бібліотечно-репозитарної системи в освітньому процесі

Бібліотечно-репозитарна система:

- забезпечує інформаційну підтримку освітнього процесу;
- є ключовим компонентом електронного університету;

- підтримує освітні програми, акредитацію та наукову діяльність;
- сприяє формуванню цифрової культури студентів;
- забезпечує відкритість та прозорість наукових результатів університету.

1.1.2 Проблеми та виклики управління фондами ЗВО

Незважаючи на розвиток цифрових технологій, більшість університетів (включно з НУК) зустрічаються з однаковими викликами:

Проблема 1. Роз'єднаність інформаційних систем

Друкований фонд ведеться в АБІС (наприклад, ІРБІС), цифровий фонд — у DSpace.

Системи не мають:

- спільної бази даних;
- спільних ідентифікаторів;
- автоматичного обміну метаданими;
- інтегрованого пошуку.

Проблема 2. Дублювання метаданих

Каталогізатору доводиться:

- окремо створювати запис в ІРБІС;
- окремо створювати запис у DSpace;
- вручну копіювати авторів, назву, рік, ключові слова;
- перевіряти відповідність стандартам MARC21 та Dublin Core.

Проблема 3. Висока трудомісткість

Через відсутність інтеграції бібліотекарі витрачають у середньому:

10–15 хвилин на створення одного запису в АБІС, 5–10 хвилин на створення матеріалу в DSpace, разом 15–25 хвилин на один документ.

Для бібліотеки з фондом у 200 тис. одиниць це колосальні витрати.

Проблема 4. Складність пошуку для студентів і викладачів

Користувачам доводиться:

- шукати в АБІС друковані матеріали;
- шукати в DSpace електронні;
- робити це вручну, у різних інтерфейсах.

Це знижує:

- доступність ресурсів;
- швидкість пошуку;
- загальну якість освітнього процесу.

Проблема 5. Відсутність єдиного електронного каталогу

Немає:

- єдиної точки доступу;
- єдиної аналітики;
- централізованого обліку фондів;
- можливості формувати комплексні звіти.

1.1.3 Аналіз існуючої бібліотечно-репозитарної системи НУК

У Національному університеті кораблебудування ім. адмірала Макарова (НУК) сьогодні використовуються дві незалежні інформаційні системи:

- Автоматизована бібліотечна інформаційна система ІРБІС64 — для роботи з друкованими фондами.
- Інституційний репозитарій DSpace — для управління цифровими фондами.
- Ці системи функціонують паралельно, не будучи інтегрованими між собою, що створює низку організаційних, технологічних та інформаційних труднощів.

А. Аналіз роботи АБІС ІРБІС64 у НУК

ІРБІС64 — це поширена в бібліотеках України АБІС, побудована на основі MARC-форматів та розрахована на роботу з друкованою літературою.

Основні функції, які реалізує ІРБІС у НУК:

- Каталогізація друкованих видань.
- Ведення інвентарних записів.
- Створення фондів та баз даних.
- Пошук за бібліографічними описами.
- Обслуговування студентів та викладачів.

- Формування статистичної звітності.
- Робота з читачами (видача/повернення).
- Розміщення каталогу у web-доступі (модуль Web-IRBIS).
- Технологічні особливості ІРБІС у НУК:
- Використання локальної бази даних у форматі ІРБІС.
- Підтримка стандарту MARC21.
- Автономність від зовнішніх платформ.
- Обмежена підтримка API.
- Старий інтерфейс та складність навчання персоналу.
- Недоліки ІРБІС у контексті стратегії НУК:
- Немає можливості інтегрувати цифрові документи.
- Закритий формат даних ускладнює перенесення інформації.
- Відсутній повноцінний REST API для автоматизації.
- Застаріле UI, яке не відповідає сучасним вимогам.
- Каталог ІРБІС існує окремо від репозитарію.

В. Аналіз роботи інституційного репозитарію DSpace у НУК

DSpace — міжнародно визнана система для організації цифрових наукових та навчальних ресурсів.

У НУК репозитарій використовується для:

- Зберігання електронних версій дипломів і дисертацій.
- Публікації наукових статей викладачів.
- Формування елементів наукового профілю університету.
- Надання відкритого доступу до наукових результатів.
- Забезпечення збору матеріалів для акредитації спеціальностей.

Переваги DSpace у НУК:

- Підтримка міжнародних стандартів Dublin Core.
- Можливість імпорту та експорту метаданих.
- Підтримка протоколу OAI-PMH для пошукових систем.
- Розвинена система колекцій та підколекцій.
- Наявність сучасного web-інтерфейсу.
- Недоліки DSpace у контексті проєкту:

- Відсутня автоматична прив'язка електронного документа до друкованого видання.
- Створення запису вимагає ручного заповнення всіх полів.
- DSpace не може автоматично підтягувати дані з ІРБІС.
- Неможливість автоматичного створення єдиного каталогу.
- Труднощі у побудові об'єднаних звітів для акредитації.

С. Взаємодія між ІРБІС та DSpace у НУК (поточний стан)

На практиці в НУК існує повна відсутність автоматизованої взаємодії між:

- обліком друкованих документів в ІРБІС;
- обліком цифрових документів у DSpace.

Як виглядає процес зараз:

1. Каталогізатор створює запис у ІРБІС

Вводить назву, авторів, рік, УДК, ISBN, ключові слова.

Створює бібліографічний опис.

2. Окремо створюється запис у DSpace

Завантажується PDF або інший файл.

Знову (!) вводяться ті ж дані вручну.

3. Зв'язку між записами немає

Навіть якщо це одна і та сама книга.

4. Студент шукає в двох місцях

- у каталозі АБІС та окремо у репозитарії.

5. Статистика роботи бібліотеки теж розділена

- частина в ІРБІС, частина в DSpace.

1.1.4 Недоліки існуючої моделі та обґрунтування актуальності

На основі аналізу ІРБІС та DSpace в НУК можна виділити такі ключові недоліки:

Фрагментація інформаційних потоків

- відсутня централізація даних, дані дублюються.

Неможливість побудови сучасного електронного каталогу

- користувачам важко отримати повну картину наявних ресурсів.

Збільшення трудовитрат персоналу

- ручне дублювання записів займає до 40% робочого часу.

Низька ефективність пошукових механізмів

- роз'єднаність систем ускладнює навчальний процес.

Відсутність підтримки відкритих API та сервісів

- неможливо створити єдиний доступний веб-портал.

Труднощі у веденні звітності та статистики

- дані збираються вручну з різних джерел.

1.2 Вибір концепції та технологічного забезпечення

1.2.1 Аналіз технологій та підходів до розробки інформаційних систем

У процесі створення інформаційної системи управління фондами друкованих і цифрових видань необхідно обрати технології, що забезпечуватимуть:

- надійну роботу з базою даних;
- можливість масштабування;
- підтримку API для взаємодії з іншими системами;
- високий рівень безпеки;
- сучасний web-інтерфейс;
- рости розгортання на серверах ЗВО.

З огляду на особливості предметної області, сучасні тренди та технічні можливості ЗВО, найдоцільнішим є використання клієнт-серверної архітектури з web-інтерфейсом.

Основні вимоги до технологій:

- Підтримка REST API
- Для інтеграції з DSpace, Moodle або іншими сервісами.
- Сумісність з бібліотечними форматами
- MARC21 (для друкованих фондів)
- Dublin Core (для електронних документів)
- Можливість міграції старих даних

- Особливо важливо для імпорту з АБІС ІРБІС64.
- Розгортання на серверах Linux / Ubuntu

Оскільки більшість університетських серверів працюють на Debian/Ubuntu.

- Підтримка ролей та авторизації
- Студент
- Викладач
- Бібліотекар
- Адміністратор

Можливість додавання модулів в майбутньому

Наприклад:

- модуль статистики
- модуль QR/штрих-кодів
- модуль імпорту з PDF
- модуль повнотекстового пошуку

1.2.2 Обґрунтування вибору Python + Django + PostgreSQL

Python — це:

- універсальна мова високого рівня;
- активно використовується в освіті й науці;
- має велику кількість бібліотек;
- добре підходить для роботи з даними;
- простий у підтримці та розширенні.

Django — це фреймворк, створений саме для таких задач:

- чітка структура проєкту
- вбудована адмін-панель (дуже корисно для бібліотеки);
- підтримка ORM — зручно працювати з БД;
- система авторизації;
- просте створення REST API через Django REST Framework;
- висока безпека: захист від XSS, CSRF, SQL Injection;
- велика кількість документації.

- Для інформаційних систем університетів Django є фактично стандартом.

PostgreSQL — це одна з кращих СУБД для наукових і освітніх інформаційних систем:

- підтримка складних запитів;
- висока надійність;
- підтримка JSONB — для зберігання метаданих;
- відкритий та безкоштовний продукт;
- сумісність з DSpace (він теж використовує PostgreSQL);
- найкраще працює з Django.

Що дає комбінація Django + PostgreSQL

- Високу стабільність.
- Навантаження до десятків тисяч записів.
- Легке розгортання на сервері ЗВО.
- Повноцінний API для інтеграції.
- Можливість створення гнучких моделей даних.

1.2.3 Аналіз форматів метаданих

Оскільки система працюватиме з двома типами фондів, необхідно забезпечити підтримку відповідних стандартів.

MARC21 — для друкованих фондів

MARC21 — це міжнародний стандарт, що використовується бібліотеками у всьому світі.

Його структура включає:

- поля;
- підполя;
- індикатори;
- коди бібліографічних елементів.

Приклад:

100 — автор

245 — назва

260 — вихідні дані

650 — ключові слова

020 — ISBN

Для нашої IC MARC21 використовується для:

- імпорту записів з АБІС;
- обробки друкованих фондів;
- створення конвертера MARC → JSON.

Dublin Core — для цифрових документів

Dublin Core — стандарт, який використовує DSpace.

Його ключові схеми:

- Title
- Creator
- Subject
- Description
- Publisher
- Date
- Identifier
- Rights

У системі Dublin Core потрібен для:

- інтеграції з DSpace;
- імпорту метаданих PDF;
- створення уніфікованого каталогу.

Узгодження MARC21 і Dublin Core

Оскільки наша система повинна об'єднати два світи, необхідно створити:

Уніфіковану модель метаданих, яка дозволить:

- зберігати дубльовану інформацію без втрат;

- синхронізувати записи між друкованими та цифровими фондами;
- створювати єдиний результат пошуку для студента.

1.2.4 Вибір архітектури інформаційної системи

Після аналізу вимог найбільш оптимальною є:

Клієнт-серверна архітектура + REST API

У системі буде три рівні:

1. Рівень даних (Database layer)

- PostgreSQL
- Таблиці фондів
- Таблиці користувачів
- Таблиці авторів
- Таблиці цифрових файлів
- Зв'язки між MARC21 і Dublin Core
- Таблиця єдиного каталогу

2. Серверна частина (Backend layer)

Python + Django + Django REST Framework

Функції:

- CRUD для фондів
- API для інтеграції
- Авторизація
- Обробка MARC21 → JSON
- Обробка DublinCore → JSON
- Модуль синхронізації
- Модуль статистики

3. Клієнтська частина (Frontend layer)

Django Templates або Vue.js / React (на вибір)

Поки що робота не вимагає SPA, тому залишаємо:

- Django Templates

- Bootstrap
- Статичний JS для динаміки

1.3 Математичне забезпечення інформаційної системи

1.3.1 Формалізація бібліотечно-репозитарної системи ЗВО

Будь-яка ІС може бути описана у вигляді трійки:

$$IS = \langle D, P, F \rangle$$

де:

- D — множина даних (фонди, користувачі, метадані);
- P — процеси обробки (каталогізація, пошук, синхронізація);
- F — функції доступу та взаємодії користувачів із системою.

У рамках нашої теми множину даних позначимо як:

$$D = D_p \cup D_d$$

де:

- D_p — множина друкованих видань (print);
- D_d — множина цифрових ресурсів (digital).

1.3.2 Математична модель метаданих

Модель друкованого видання:

$$Book_p = \langle id_p, A, T, Y, UDK, ISBN, K, Inv, S_p \rangle$$

де:

- id_p — внутрішній ідентифікатор;
- A — множина авторів;
- T — назва;
- Y — рік видання;
- UDK — універсальний десятиковий класифікатор;
- $ISBN$ — міжнародний номер книги;
- K — множина ключових слів;
- Inv — інвентарний номер;
- S_p — статус (у фонді, на руках, списано).

Модель цифрового ресурсу:

$$Book_d = \langle id_d, A, T, Y, DC, File, S_d \rangle$$

де:

- DC — метадані у форматі Dublin Core;
- $File$ — шлях до файлу або його хеш;
- S_d — статус доступу.

1.3.3 Модель уніфікації фондів

Оскільки система об'єднує два фонди, вводимо поняття єдиного запису:

$$U = \langle id_u, id_p, id_d, M \rangle$$

де:

- id_u — уніфікований ідентифікатор;
- id_p — прив'язка до друкованого запису (необов'язково);
- id_d — прив'язка до цифрового запису (необов'язково);
- M — уніфікована множина метаданих.

Структура множини метаданих:

$$M = Norm(M_p, M_d)$$

де:

- M_p — MARC21 метадані;
- M_d — Dublin Core;
- $Norm$ — нормалізаційна функція, яка усуває дублювання і створює універсальне поле для каталогу.

Приклад нормалізації полів:

- $Norm(T_{245_MARC}, Title_{DC}) = Title$
- $Norm(A_{100_MARC}, Creator_{DC}) = Author$
- $Norm(K_{650_MARC}, Subject_{DC}) = Keywords$

1.3.4 Математична модель процесу пошуку

Пошук визначається як відображення:

$$Search : Q \rightarrow R$$

де:

- Q — множина текстових або структурних запитів;
- $R \subseteq U$ — множина записів, що відповідають запиту.

Формалізуємо запит:

$$Q = \langle q_1, q_2, \dots, q_n \rangle$$

де кожне q_i — умова пошуку.

Алгоритм пошуку:

$$R = \{u \in U \mid match(u.M, Q) = true\}$$

де функція відповідності:

$$match(u.M, Q) = \bigwedge_{i=1}^n cond(u.M, q_i)$$

Це дозволяє:

- фільтрувати за автором;
- робити пошук за назвою;
- робити повнотекстовий пошук (через розширення);
- шукати одночасно в друкованих і цифрових фондах.

1.3.5 Математична модель синхронізації фондів

Створимо модель, де:

- M_p — метадані друкованих видань;
- M_d — метадані цифрових видань;
- M_s — синхронізовані метадані.

Визначимо функцію синхронізації:

$$Sync : M_p \cup M_d \rightarrow M_s$$

Для кожного параметра:

$$M_s(i) = \begin{cases} M_p(i), & \text{якщо } M_p(i) \neq \emptyset \\ M_d(i), & \text{якщо } M_p(i) = \emptyset \end{cases}$$

У випадку конфлікту:

$$M_s(i) = Resolve(M_p(i), M_d(i))$$

Функція Resolve може працювати за принципом:

- Пріоритет MARC21.
- Пріоритет Dublin Core.
- Пріоритет більш повного або нового запису.

1.3.6 Математична модель взаємодії користувачів із системою

Маємо множину ролей:

$$R = \{student, teacher, librarian, admin\}$$

Кожен користувач:

$$User = \langle id, login, role, perms \rangle$$

Функція доступу:

$$Access(u, op) = \begin{cases} true, & \text{якщо } op \in perms(u.role) \\ false, & \text{інакше} \end{cases}$$

Приклади:

- студент має право лише на пошук;
- бібліотекар має право на редагування фонду;
- адміністратор має повний доступ.

1.3.7 ER-модель (концептуальна)

Сутності:

- BookPrint
- BookDigital
- UnifiedRecord
- Author
- Keyword
- File
- User
- Role
- SyncLog

Зв'язки:

- Один UnifiedRecord $\leftarrow$ може містити $\rightarrow$ BookPrint
- Один UnifiedRecord $\leftarrow$ може містити $\rightarrow$ BookDigital
- BookPrint $\rightarrow$ багато-до-багатьох $\rightarrow$ Author
- BookDigital $\rightarrow$ багато-до-багатьох $\rightarrow$ Author
- UnifiedRecord $\rightarrow$ багато-до-багатьох $\rightarrow$ Keyword
- BookDigital $\rightarrow$ один-до-багатьох $\rightarrow$ File
- User $\rightarrow$ один-до-одного $\rightarrow$ Role
- UnifiedRecord $\rightarrow$ один-до-багатьох $\rightarrow$ SyncLog

1.3.8 Формальна постановка задачі

Потрібно знайти таку систему:

$$IS^* = \arg \min_{IS} (T_{cat}(IS) + T_{sync}(IS) + Err(IS))$$

де:

- T_{cat} — час каталогізації (має зменшитись $\approx$ у 2 рази);
- T_{sync} — час синхронізації фондів;
- Err — кількість дубльованих або помилкових записів.

З обмеженнями:

- $Security(IS) \geq S_{min}$
- $Availability(IS) \geq A_{min}$
- $Compatibility(IS, MARC21, DC) = true$

Висновок до розділу 1

У першому розділі виконано аналіз предметної області бібліотечних інформаційних систем та визначено основні проблеми, пов'язані з обліком, зберіганням і пошуком друкованих та цифрових бібліотечних ресурсів. Розглянуто сучасні підходи до організації бібліотечних каталогів, а також проаналізовано існуючі програмні рішення, зокрема автоматизовані бібліотечні системи та інституційні репозитарії. На основі проведеного аналізу виявлено недоліки наявних систем, зокрема обмежені можливості уніфікації записів, складність інтеграції з різними джерелами даних та недостатню гнучкість у роботі з цифровими ресурсами. Це обґрунтовує доцільність розроблення інформаційної системи, орієнтованої на уніфіковане зберігання, ефективний пошук та розширювану архітектуру. Отримані результати аналізу предметної області стали основою для формування вимог до інформаційної системи та подальшого проєктування її архітектури і структури бази даних, що розглядається у наступному розділі роботи.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ОБЛІКУ ТА УНІФІКАЦІЇ БІБЛІОТЕЧНИХ РЕСУРСІВ

2.1. Аналіз предметної області

Бібліотечні інформаційні системи є ключовими компонентами сучасних інформаційних інфраструктур освітніх, наукових та культурних установ. Вони забезпечують облік, систематизацію, пошук і доступ до великих масивів документів — як друкованих, так і електронних. Стрімкий розвиток цифрових технологій створює умови для інтеграції бібліотечних фондів, але водночас висуває суттєві вимоги до стандартизації даних, їх структурованості, багаторівневої доступності та уніфікації описів.

У сучасних бібліотеках співіснують різні типи ресурсів:

- Друковані документи (книги, журнали, навчальні посібники, архівні матеріали);
- Електронні документи (PDF, DOCX, EPUB, MP3, MP4, архіви);
- Метадані (MARC, DublinCore, BibTeX тощо).

Наявні системи обліку часто використовують розрізнені механізми опису документів, що спричиняє дублювання, помилки та ускладнює підтримку єдиного електронного каталогу. Проблеми, які найчастіше зустрічаються в бібліотечних установах:

1. відсутність єдиного підходу до опису друкованих та електронних документів;
2. неможливість інтегрувати різні системи обліку між собою;
3. складність у підтримці великої кількості метаданих;
4. слабка або відсутня підтримка API та автоматизації;
5. обмежені можливості пошуку.

Предметна область вимагає створення інформаційної системи, здатної:

- уніфікувати дані незалежно від типу документа;
- підтримувати інтеграцію і синхронізацію з зовнішніми репозитаріями;
- забезпечувати ефективний пошук за великими обсягами метаданих;
- працювати з файлами різних типів та форматів;
- підтримувати бібліотечні стандарти опису документів.

2.2. Аналіз існуючих бібліотечних систем

Для оцінки рівня технологічних вимог було проаналізовано найбільш поширені автоматизовані бібліотечно-інформаційні системи (АБІС).

Система ІРБІС

ІРБІС — одна з найстаріших і найпоширеніших автоматизованих систем у бібліотеках України. Вона містить розвинені модулі каталогізації та працює з форматами MARC21. Проте:

- інтерфейс застарілий;
- робота з електронними документами обмежена;
- інтеграція через API практично відсутня;
- масштабування складне.

Кoha

Потужна відкритокодна АБІС, яка добре працює з каталогізацією. Однак:

- вимоглива до серверних ресурсів;
- складна у налаштуванні;
- обмежена в адаптації під нестандартні моделі даних;
- електронні ресурси підтримуються поверхнево.

DSpace

Найкраще підходить для репозитаріїв електронних матеріалів, але:

- не призначений для роботи з друкованими ресурсами;
- маркована каталогізація не підтримується повністю;
- інтеграція з локальними бібліотеками часто складна.

2.3. Формування вимог до розроблюваної системи

2.3.1. Функціональні вимоги

Розроблювана система повинна забезпечувати:

1. облік друкованих документів;
2. облік електронних документів;
3. автоматичну уніфікацію описів;
4. завантаження PDF та інших файлів;

5. формування метаданих (автор, рік, ключові слова, анотація);
6. повнотекстовий пошук;
7. REST API для обміну даними;
8. синхронізацію з репозитаріями;
9. управління правами доступу.

2.3.2. Нефункціональні вимоги

- Масштабованість: можливість працювати із сотнями тисяч записів.
- Надійність: захищене сховище файлів, резервне копіювання.
- Продуктивність: швидкий пошук по метаданих.
- Модульність: розділення на логічні блоки.
- Стандартизація: підтримка MARC, DublinCore.
- Безпека: авторизація, аутентифікація, логування операцій.

2.4. Проєктування загальної архітектури

Архітектура системи будується на багаторівневій моделі.

2.4.1. Архітектурні рівні

1. Клієнтський рівень: веб-інтерфейс на базі Django Templates, Bootstrap.
2. Прикладний рівень (Backend): Django + Django REST Framework.
3. Рівень даних: PostgreSQL, індексація пошуку, сховище файлів.
4. Модуль синхронізації: окремі фонові процеси з CRON, які отримують зовнішні дані.

2.4.2. Структурна схема системи



Рисунок 1 – Структурна схема інформаційної системи

Основні компоненти:

- модуль керування друкованими книгами;
- модуль керування цифровими ресурсами;
- модуль уніфікації записів;
- модуль авторизації;
- модуль синхронізації;
- модуль пошуку;
- модуль зберігання файлів;
- веб-інтерфейс.

2.5. Проектування бази даних

База даних будується за принципами третьої нормальної форми.

2.5.1. Основні сутності

UnifiedRecord

- уніфікована картка, яка описує документ незалежно від його виду.

BookPrint

- метадані друкованої книги.

BookDigital

- метадані цифрового ресурсу.

Author

- окрема таблиця для авторів.

Keywords

- ключові слова.

FileResource

- фізичні файли цифрових документів.

User

- користувачі системи.

SyncLog

- журнал синхронізації.

2.5.2. ER-діаграма структури бази даних

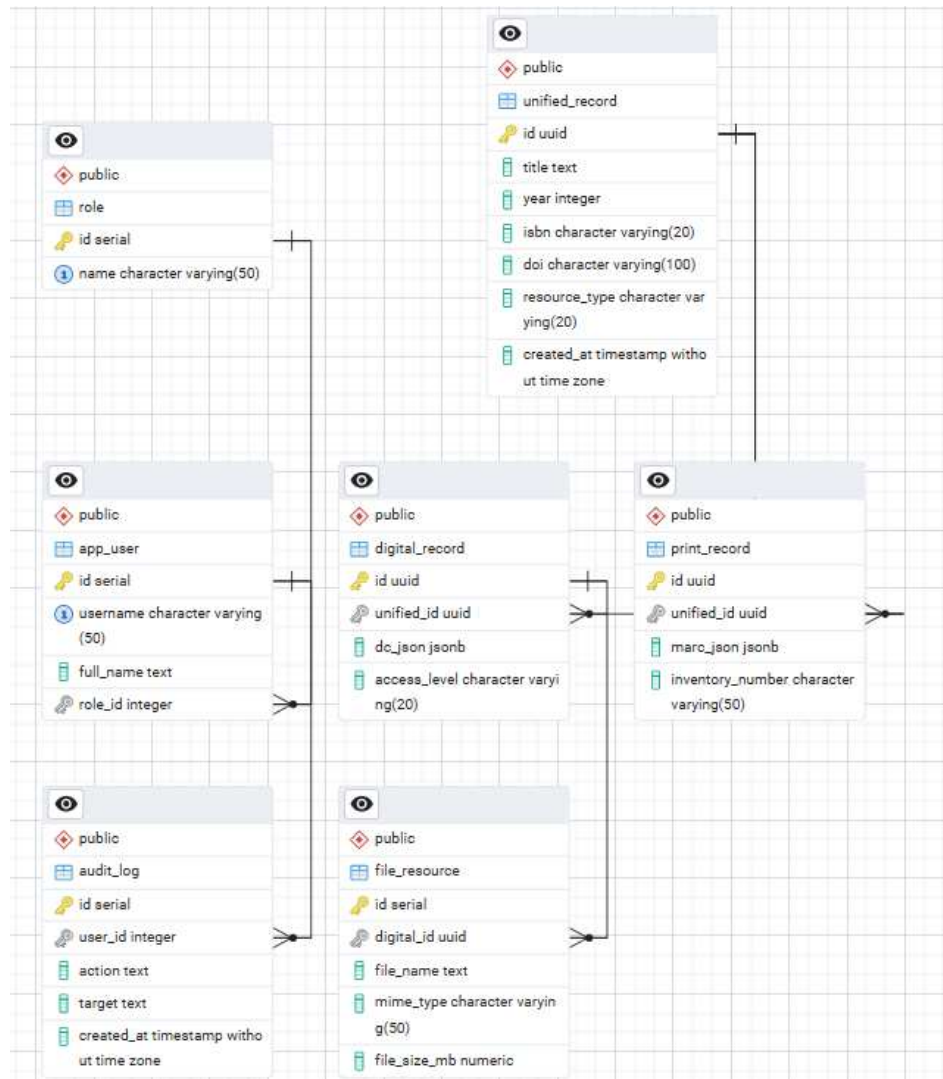


Рисунок 2 – ER-діаграма структури бази даних інформаційної системи
Зв'язки:

- UnifiedRecord $\longleftrightarrow$ Author (M:N)
- UnifiedRecord $\longleftrightarrow$ Keyword (M:N)
- BookPrint $\longleftrightarrow$ UnifiedRecord (1:1)
- BookDigital $\longleftrightarrow$ UnifiedRecord (1:1)
- BookDigital $\longleftrightarrow$ FileResource (1:1)

2.6. UML-діаграми

2.6.1. Use Case діаграма

Актори:

- Користувач
- Бібліотекар
- Адміністратор

Основні прецеденти:

- пошук документів;
- перегляд каталогу;
- завантаження файлу;
- додавання нового ресурсу;
- редагування записів;
- синхронізація.

2.6.2. Activity діаграми

(1) процес пошуку;

(2) процес додавання документа.

2.6.3. Sequence діаграми

(1) послідовність запиту пошуку;

(2) послідовність завантаження електронного документа.

2.7. Логічна модель даних

Уточнює:

- структура таблиць;
- типи полів;
- індекси;
- зовнішні ключі;
- обмеження.

2.8 UML Sequence — послідовність взаємодії користувача із системою при пошуку та завантаженні ресурсу

2.8.1 Учасники послідовності (lifelines)

- **Користувач (Client)** — веб-браузер або фронтенд (React).
- **API Gateway / Web UI** — проміжний рівень (може бути вбудований у Django).
- **Auth Service (JWT)** — сервіс аутентифікації/авторизації (компонент Django).
- **Search Controller (Django REST endpoint)** — контролер, який обробляє пошукові запити.

- **Normalizer / Sync Module** — модуль нормалізації метаданих та об'єднання результатів.
- **PostgreSQL (DB)** — реляційна база даних для метаданих.
- **Full-text search (FTS) / Index** — індекс (Postgres GIN/tsvector або зовнішній Elastic) для повнотекстового пошуку.
- **File Storage** — файлове сховище (локальне або мережеве), що зберігає цифрові файли.
- **Logger / Audit** — журнал операцій і аудиту.

2.8.2 Сценарій А — Пошук ресурсу: покрокова послідовність

1. Користувач вводить запит у пошукову форму (наприклад: author:"Іваненко" year:2019 keyword:машинне навчання) та натискає «Пошук».
2. Frontend формує HTTP GET запит до API:
3. GET
/api/v1/search?q=author%3A%22Іваненко%22+keyword%3A%22машинне+навчання%22&page=1&page_size=20&sort=relevance
4. Authorization: Bearer <JWT>
5. Accept: application/json
6. API Gateway перевіряє JWT (Auth Service). Якщо токен відсутній/протермінований — повертає 401 Unauthorized. Якщо користувач має блокування або обмеження — 403 Forbidden з поясненням.
7. Search Controller приймає запит:
 - валідує параметри (q, page, page_size, filters);
 - нормалізує вхід (видаляє зайві пробіли, екранує символи);
 - рахує хеш запиту для кешу (cache key).
8. Контролер перевіряє кеш (Redis або внутрішній кеш Django). Якщо є кешований результат для user+query+page → повертає 200 OK з cached payload (мінімізує затримку).
9. Якщо кешу немає:
 - контролер формує два паралельних підзапити:

- FTS query → Full-text search індекс (сьогодні реалізуємо через Postgres tsvector і GIN);
 - SQL-запит на метадані → пошук у структурних полях (author, title, ISBN, UDK).
 - ці запити виконуються паралельно (async workers/threads).
10. Отримані результати передаються модулю Normalizer:
- Normalizer зіставляє MARC21-поля та DublinCore-поля;
 - знімає дублікати (за ISBN/DOI/комбінацією title+author+year);
 - формує уніфікований запис UnifiedRecord з полями: id, title, authors[], year, type (print|digital), snippet, score.
11. Normalizer ранжує результати (score) за:
- релевантністю FTS;
 - точністю співпадіння полів (title exact > title fuzzy);
 - свіжістю (новіші публікації можуть мати невеликий бонус);
 - наявністю повного тексту (digital з відкритим доступом > без доступу).
12. Search Controller формує paginated JSON-відповідь:
13. {
14. "page": 1,
15. "page_size": 20,
16. "total": 347,
17. "results": [
18. {
19. "id": "U-12345",
20. "title": "Машинне навчання: методи та застосування",
21. "authors": ["Іваненко І.І."],
22. "year": 2019,
23. "type": ["print", "digital"],
24. "snippet": "Огляд методів машинного навчання для задач прогнозування...",
25. "has_fulltext": true,

26. "score": 0.987

27. },

28. ...

29.]

30. }

31. Результат кешується (TTL, наприклад 60–300 с, залежно від навантаження) та логуються метрики (query, latency, hits, user_id).

32. Frontend отримує JSON, відображає результати (snippet, фільтри, кнопки завантаження/посилання). Користувач може клікнути на конкретний запис для перегляду повної картки.

Аспекти надійності та продуктивності (детально):

- **Кешування:** кеш на рівні запиту + кеш окремих метаданих (часто читаються записи). Ключ кешу = user_role|query|page|filters.
- **Паралелізм:** FTS + структурний SQL виконуються паралельно; це зменшує час відповіді.
- **Пагінація:** обмежує обсяг даних; використовує OFFSET/LIMIT або курсорну пагінацію для великих наборів.
- **Рейтинг:** комбінований score із FTS rank (ts_rank) та бізнес-факторів.
- **Робота з великими текстами:** у відповіді нічого не передається повного файлу, лише метадані; це знижує навантаження на мережу.
- **Таймаути:** FTS запити мають таймаут (наприклад 3–5 с). Якщо пошук перевищив таймаут → повертаємо 503 Service Unavailable + часткові результати (якщо доступні) або рекомендацію повторити пошук.

2.8.3 Сценарій В — Завантаження цифрового файлу (streaming)

1. Користувач у картці запису натискає «Завантажити PDF».
2. Frontend формує запит:
3. GET /api/v1/items/{unified_id}/download
4. Authorization: Bearer <JWT>
5. Accept: application/pdf

6. Auth Service перевіряє права (Access(u, op)). Правило: якщо `item.access == "open"` → дозволено всім; якщо `item.access == "restricted"` → перевірка ролі/угоди; якщо `item.access == "embargoed"` → 403 Forbidden до вказаної дати.

7. Якщо дозвіл є — контролер генерує pre-signed URL (якщо зовнішнє сховище) або відкриває потік читання файлу із File Storage.

8. Для streaming завантаження:

- встановлюється заголовок `Content-Disposition: attachment; filename="Ivanenko2019_ML.pdf"`;
- `Content-Type: application/pdf`;
- використовуємо `chunked transfer encoding`: читання блоку (~64–256 KB) → запис блоку в `response` → `flush`.

9. Під час передачі:

- логуємо `download_start`, `user_id`, `item_id`;
- ведеться контроль швидкості за необхідності (rate limiting);
- підтримується перевірка цілісності (якщо клієнт надіслав `Range` запит — підтримується `resume`).

10. Після успішного завершення логується `download_complete`.

Поточні помилки та відповіді:

- 401 Unauthorized — токен відсутній/недійсний.
- 403 Forbidden — недостатні права або доступ обмежений.
- 404 Not Found — файл відсутній або посилання некоректне.
- 500 Internal Server Error — помилка сховища; в лог додаємо трасування.

- 429 Too Many Requests — користувач перевищив ліміти (захист від DDoS).

Додаткові заходи:

- Для великих/частих запитів — використовуємо pre-signed URLs з обмеженим TTL (S3-подібна модель) або CDN для оффлоудингу трафіку.
- Для аудиту — зберігаємо записи про завантаження для статистики та академічної доброчесності.

- Підтримка Range дозволяє відновлення завантаження з останньої точки.

2.8.4 HTTP / REST API — повний перелік endpoint'ів для сценаріїв

Пошук

- GET /api/v1/search
- Параметри: q, page, page_size, sort, filters (JSON-encoded)
- Відповідь: JSON (див. приклад вище)
- Статуси: 200, 400 (bad request), 401, 503

Отримати картку ресурсу

- GET /api/v1/items/{id}

Відповідь:

- {
- "id":"U-12345",
- "title":"...",
- "authors":["..."],
- "year":2019,
- "type":["digital"],
- "files":[
- {"file_id":"F987","mime":"application/pdf","access":"open","size":1234567}
-],
- "metadata": { "marc": { ... }, "dc": { ... } }
- }
- Статуси: 200, 401, 403, 404

Завантаження файлу

- GET /api/v1/items/{id}/download (або /files/{file_id}/download)
- Підтримка Range заголовків
- Статуси: 200, 206 (partial), 401, 403, 404, 429, 500

Аутентифікація

- POST /api/v1/auth/login
- Тіло:
- {"username":"ivan","password":"***"}

Відповідь:

- {"access_token":"<JWT>","expires_in":3600,"refresh_token":"<token>"}
- Статуси: 200, 401

Псевдо-мета: кеш/health/logs

- GET /api/v1/health — перевірка стану сервісів.
- GET /api/v1/metrics — метрики (latency, search count) — доступ тільки адміну.

2.8.5 Транзакції, блокування і консистентність даних

- **Каталогізація / модифікація записів:** операції запису (create/update/delete) виконуються у межах транзакцій PostgreSQL з рівнем ізоляції READ COMMITTED або REPEATABLE READ для критичних масивних оновлень.
- **Синхронізація записів (batch):** при масовому імпорті застосовується підхід:
 - створити тимчасову таблицю → валідовати → атомарно перемістити у продукційну таблицю;
 - під час синхронізації заблокувати лише вузькі індекси або використовувати optimistic locking (version field).
- **Усунення дублювання:** на рівні бази даних — унікальні індекси на ISBN, DOI; при відсутності — алгоритм fuzzy-matching у Normalizer із подальшим ручним підтвердженням бібліотекарем для сумнівних пар.

2.8.6 Транзакції, блокування і консистентність даних

- Для всіх критичних дій (створення/редагування записів, завантаження файлів, операції адміністратора) зберігаємо аудиторські логи:

user_id, action, item_id, timestamp, ip, result.

Метрики:

latency (p95, p99), throughput (req/s), cache hit ratio, search query distribution.

Алерти:

перевищення середньої латентності > threshold;

spike у кількості помилок 5xx.

2.8.7 Помилки, повтори та idempotency

- **Ідемпотентні операції:** PUT /api/v1/items/{id} — мають бути ідемпотентні. Для операцій створення — використовуємо POST і унікальний client_request_id для idempotency.
- **Retry strategy:** клієнт може повторити запит при 5xx з backoff (exponential). Сервер — повинен ідентифікувати дублікати за client_request_id.
- **Відновлення після помилок:** у разі помилок сховища під час upload — файл маркується як upload_failed і доступний для повтора завантаження через UI з індикацією помилки.

2.8.8 Безпека та права доступу

- **Аутентифікація:** JWT (access + refresh). Access токен — короткий TTL (наприклад, 1 година). Refresh — довший (7–30 днів).
- **Авторизація:** RBAC (roles: student, teacher, librarian, admin). Додаткові permission flags (can_edit_catalog, can_import, can_download_restricted).
- **Шифрування:** TLS для всього трафіку. Чутливі поля у БД можуть бути зашифровані (PGP / application level).
- **Rate limiting:** ураховуємо як рівень IP, так і рівень користувача (наприклад 100 req/min для студентів).
- **CSRF / XSS / SQL Injection:** використовуємо захист фреймворку Django, ORM для запобігання SQL-ін'єкцій.

2.8.9 Приклад повної послідовності

(спрощений лог із часовими позначками для одного запиту пошуку + одного завантаження)

1. T+0 ms: Client → GET /api/v1/search?q=... (Auth header)
2. T+5 ms: Gateway → Auth Service (JWT валідація) → 200 OK
3. T+7 ms: Search Controller — кеш miss
4. T+8 ms: Search Controller → FTS (Postgres tsearch) (async)
5. T+8 ms: Search Controller → SQL structural search (async)
6. T+35 ms: FTS returns partial results
7. T+40 ms: SQL returns structural results
8. T+42 ms: Normalizer merges results, ranks → builds response
9. T+45 ms: Response cached, logged → Server → Client 200 OK (payload)
10. T+1200 ms: Client → GET /api/v1/items/U-12345/download
11. T+1203 ms: Auth check → allowed
12. T+1210 ms: Controller opens file stream → sends headers, starts chunked transfer
13. T+2400 ms: Client отримав 1 MB → download progress...
14. T+8500 ms: download complete → server logs download_complete

(ці times — орієнтовні; у продукційній системі latency залежатиме від інфраструктури й розміру файлу)

- UML Sequence для двох найважливіших сценаріїв (пошук та завантаження) детально відображає: аутентифікацію, кешування, паралельні запити до FTS і структурних таблиць, нормалізацію результатів та streaming file delivery.

- Архітектурні рішення (кеш, FTS, streaming, JWT, audit logs) поєднуються для досягнення мети — швидкого, безпечного та надійного доступу до гібридних бібліотечних фондів.

- Рекомендації для реалізації: використовувати курсорну пагінацію для великих наборів, Redis для кеша, Postgres GIN/tsvector або Elastic для FTS, pre-signed URLs / CDN для важких файлів, та ретельно налаштувати таймаути й ліміти.

2.9 Інфологічна модель даних (ER-діаграма інформаційної системи)

2.9.1 Призначення та вимоги до моделі даних

Модель даних інформаційної системи управління фондами друкованих і цифрових видань повинна забезпечувати:

- централізоване зберігання каталогізаційних записів;
- інтеграцію двох типів фондів:
- друкованого (MARC21);
- цифрового (Dublin Core);
- синхронізацію записів між фондами;
- уніфікацію метаданих;
- зберігання цифрових файлів та інформації про доступ;
- підтримку користувачів із різними ролями;
- аудит операцій.

Оскільки система працює з двома окремими джерелами опису, ключовою вимогою є:

унікнення дублювання записів наявність єдиного «уніфікованого» бібліографічного запису

2.9.2 Основні сутності системи

До ядра моделі належать такі сутності:

1. **UnifiedRecord** – уніфікований запис ресурсу
2. **PrintRecord** – запис друкованого видання (MARC21)
3. **DigitalRecord** – запис цифрового ресурсу (Dublin Core)
4. **File** – цифровий файл
5. **User** – користувач системи
6. **Role** – роль доступу

7. **AuditLog** – журнал подій
8. **SyncLog** – журнал синхронізації

2.9.3 Опис ключових сутностей

UnifiedRecord

Це центральна сутність системи, яка представляє узагальнений бібліографічний опис ресурсу.

Атрибути:

Поле	Тип	Опис
id	UUID	первинний ключ
title	text	назва
year	integer	рік видання
authors	text[]	список авторів
type	enum(print, digital, hybrid)	тип ресурсу
isbn	varchar(20)	міжнародний ідентифікатор
doi	varchar(100)	цифровий ідентифікатор
created_at	timestamp	дата створення
updated_at	timestamp	дата оновлення

Особливість: у таблиці існують унікальні індекси на ISBN та DOI дозволяють уникати дублювання

PrintRecord (MARC21)

Зберігає структуру MARC-записів:

Поле	Тип
id	UUID
unified_id	UUID FK
marc_raw	jsonb
inventory_number	varchar

Поле	Тип
location	varchar

Особливість:

- marc_raw зберігає повний MARC21 запис у JSON-структурі

Наприклад:

```
{
  "100": {"a": "Іваненко І.І."},
  "245": {"a": "Машинне навчання"},
  "260": {"c": "2019"}
}
```

DigitalRecord (Dublin Core)

Поле	Тип
id	UUID
unified_id	UUID FK
dc_raw	jsonb
access_level	enum(open, restricted, embargoed)

File

Зберігає файл, прив'язаний до цифрового ресурсу.

Поле	Тип
id	UUID
digital_id	UUID FK
path	text
hash	varchar(64)
mime	varchar
size	bigint

Хеш SHA-256 дозволяє: контролювати цілісність уникати дублювання файлів

User

Поле	Тип
id	UUID
username	varchar
password_hash	varchar
full_name	text
email	text

Role (RBAC)

Поле	Тип
id	UUID
name	varchar

Типові ролі:

- student
- teacher
- librarian
- admin

AuditLog

Фіксує всі критичні дії:

Поле
user_id FK
action
item_id
timestamp
ip

SyncLog

Журнал синхронізації:

- створено зв'язок
- об'єднані записи
- створений новий цифровий запис

2.9.4 Типи зв'язків

У моделі використовуються:

- 1:1 (UnifiedRecord – DigitalRecord)
- 1:M (DigitalRecord – File)
- M:N (User – Role)
- 1:M (User – AuditLog)
- M:N (UnifiedRecord – PrintRecord) фактично спрощено до 1:M

2.9.5 Нормалізація

База даних відповідає:

ЗНФ (третій нормальній формі)

Тобто:

- немає повторюваних груп
- всі неключові атрибути залежать тільки від РК
- немає транзитивних залежностей

2.9.6 Ключові індекси для продуктивності

Обов'язкові індекси PostgreSQL:

- UNIQUE (isbn)
- UNIQUE (doi)
- GIN (title gin_trgm_ops)
- GIN (authors)
- tsvector(title, authors, marc_raw, dc_raw) для повнотекстового пошуку

2.7 Специфікація REST API

Нижче — повний, **swagger-like** опис REST API для IC управління фондами. Включаючи моделі даних, всі важливі endpoint, приклади запитів/відповідей, права доступу, поведінку при помилках, правила кешування, ідемпотентності, завантажень та безпеки.

Усі приклади — у форматі JSON. Токени — JWT (Bearer). Загальні заголовки:

Content-Type: application/json або Accept: application/json (для JSON), Accept: application/pdf — для завантаження PDF.

Висновок до розділу 2

У другому розділі виконано проєктування інформаційної системи бібліотечного каталогу з урахуванням вимог, сформованих на основі аналізу предметної області. Визначено загальну архітектуру системи, основні функціональні модулі та їх взаємодію, а також спроектовано структуру даних і механізми обміну інформацією. У межах розділу розглянуто проєктні рішення на різних рівнях абстракції, зокрема інфологічну модель даних, UML-діаграми та прикладні інтерфейси взаємодії. Також визначено підходи до забезпечення ефективного пошуку та продуктивності системи, які закладають основу для подальшої реалізації. Запропоновані проєктні рішення не є програмною реалізацією, а виконують роль концептуальної та логічної основи, на базі якої здійснюється розроблення програмного забезпечення інформаційної системи, описане у наступному розділі.

РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1. Вибір засобів і середовища розроблення

Створення сучасної бібліотечної інформаційної системи вимагає використання технологій, які забезпечують високу продуктивність, надійність, безпеку та простоту масштабування. Для реалізації системи було обрано стек технологій, що включає:

- **Мову програмування Python** через її універсальність, модульність та розвинену екосистему бібліотек.
- **Фреймворк Django** забезпечує швидку розробку веб-додатків та містить вбудовану ORM, систему маршрутизації, шаблонізатор.
- **Django REST Framework** сучасна бібліотека для створення REST API, що дозволяє легко реалізувати інтеграційні механізми.
- **PostgreSQL** — реляційна СУБД, що використовується як фізичний рівень зберігання даних інформаційної системи та інтегрується з Django через ORM-механізми.
- **HTML5, CSS3, Bootstrap** для створення зручного та адаптивного інтерфейсу.

Вибір такого набору технологій дозволяє створити масштабовану, модульну та сучасну систему, яка легко інтегрується з зовнішніми бібліотекою даних, такими як DSpace або IPBIS.

3.2. Архітектура програмного комплексу

Програмний комплекс має модульну структуру та складається з декількох логічних компонентів:

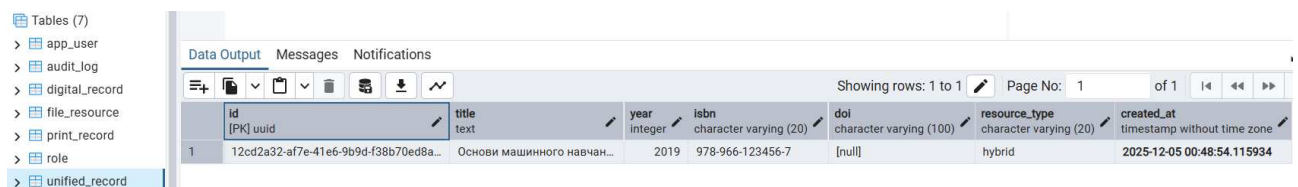
1. Модуль керування друкованими ресурсами (BookPrint)
2. Модуль керування цифровими ресурсами (BookDigital)
3. Модуль уніфікації записів (UnifiedRecord)
4. Модуль авторизації та користувачів (User)
5. Модуль зберігання файлів (FileResource)
6. Модуль пошуку та індексації
7. Модуль синхронізації із зовнішніми джерелами

8. Веб-інтерфейс для користувачів та бібліотекарів

Для фреймворку Django запропонована структура проекту має такий вигляд:

```
library_system/
├── library_system/      (головні налаштування)
├── catalog/             (основна логіка)
│   ├── models.py
│   ├── views.py
│   ├── serializers.py
│   ├── urls.py
│   ├── search.py
│   └── sync.py
├── templates/
├── static/
└── manage.py
```

3.3. Реалізація моделі даних (models.py)



id	title	year	isbn	doi	resource_type	created_at
1	Основи машинного навчан...	2019	978-966-123456-7	[null]	hybrid	2025-12-05 00:48:54.115934

Рисунок 3 – Структура та приклад запису моделі UnifiedRecord

Далі наведено основні класи моделі даних.

3.3.1. Модель UnifiedRecord

```
class UnifiedRecord(models.Model):
```

```
    title = models.CharField(max_length=255)
```

```
    description = models.TextField(blank=True)
```

```
    year = models.IntegerField(null=True, blank=True)
```

```
    authors = models.ManyToManyField('Author')
```

```
    keywords = models.ManyToManyField('Keyword')
```

```

created_at = models.DateTimeField(auto_now_add=True)

def __str__(self):
    return self.title

```

3.3.2. Модель Author

```

class Author(models.Model):
    name = models.CharField(max_length=255)
    def __str__(self):
        return self.name

```

3.3.3. Модель Keyword

```

class Keyword(models.Model):
    word = models.CharField(max_length=100)
    def __str__(self):
        return self.word

```

3.3.4. Модель BookPrint

```

class BookPrint(models.Model):
    record = models.OneToOneField(UnifiedRecord,
    on_delete=models.CASCADE)
    isbn = models.CharField(max_length=20, blank=True)
    publisher = models.CharField(max_length=255, blank=True)
    pages = models.IntegerField(null=True, blank=True)

```



The screenshot shows a database management interface. On the left, a sidebar lists tables: app_user, audit_log, digital_record, file_resource, print_record (highlighted), role, and unified_record. The main area displays the 'Data Output' for the 'print_record' table. It shows a single record with the following columns and values:

	id [PK] uuid	unified_id uuid	marc_json jsonb
1	377c3870-4881-41bc-b74b-c3f33093eb...	12cd2a32-af7e-41e6-9b9d-f38b70ed8a...	("100": {"a": "Іваненко І.І."}, "245": {"a": "Основи машинного навчання"}, "260": {"c": "2019"}, "650": {"a": "машинне навча..."})

Рисунок 4 – Структура та приклад запису друкованого ресурсу

3.3.5. Модель FileResource

```

class FileResource(models.Model):
    file = models.FileField(upload_to='uploads/')

```



```
uploaded_at = models.DateTimeField(auto_now_add=True)
```

```
extension = models.CharField(max_length=10)
```

3.3.6. Модель BookDigital

```
class BookDigital(models.Model):
```

```
    record = models.OneToOneField(UnifiedRecord,
on_delete=models.CASCADE)
```

```
    file = models.OneToOneField(FileResource, on_delete=models.CASCADE)
```

```
    size_mb = models.FloatField(null=True, blank=True)
```

Tables (7)

app_user

audit_log

digital_record

file_resource

print_record

role

unified_record

Data Output

Messages

Notifications

Showing rows: 1 to 1

Page No: 1

of 1

id [PK] uuid	unified_id uuid	dc_json jsonb
1	5572f94a-3782-4fd2-86cd-844374defc...	12cd2a32-af7e-41e6-9b9d-f38b70ed8a... {"date": "2019", "type": "Text", "title": "Основи машинного навчання", "creator": "Іваненко І.І.", "identifier": "http://repo.nuk.e

Tables (7)

app_user

audit_log

digital_record

file_resource

print_record

role

unified_record

Data Output

Messages

Notifications

Showing rows: 1 to 1

Page No: 1

of 1

id [PK] integer	digital_id uuid	file_name text	mime_type character varying (50)	file_size_mb numeric	
1	1	5572f94a-3782-4fd2-86cd-844374defc...	ml_basics.p...	application/pdf	4.8

Рисунок 5 – Зберігання цифрового ресурсу та пов'язаних файлів

3.3.7. Модель користувача

```
class User(AbstractUser):
```

```
    role = models.CharField(max_length=20, default='user')
```

Описані ORM-моделі реалізують логічний рівень подання даних інформаційної системи та використовуються серверною частиною застосунку для виконання бізнес-логіки. Фізичний рівень зберігання даних реалізовано з використанням реляційної СУБД PostgreSQL. Відповідність між логічною моделлю та фізичною структурою бази даних забезпечується ORM-механізмами фреймворку Django, які автоматично формують SQL-таблиці, індекси та обмеження цілісності на основі описаних моделей. Такий підхід дозволяє забезпечити узгодженість логічної та фізичної моделей даних без необхідності ручного написання SQL-коду.

3.4. Реалізація REST API (serializers.py + views.py + urls.py)

3.4.1. Сериалізатори

```
class UnifiedRecordSerializer(serializers.ModelSerializer):  
    authors = AuthorSerializer(many=True)  
    keywords = KeywordSerializer(many=True)  
    class Meta:  
        model = UnifiedRecord  
        fields = '__all__'
```

3.4.2. API для пошуку

```
class SearchView(APIView):  
    def get(self, request):  
        query = request.GET.get('q', '')  
        results = UnifiedRecord.objects.filter(title__icontains=query)  
        serializer = UnifiedRecordSerializer(results, many=True)  
        return Response(serializer.data)
```

3.4.3. Маршрутизація (urls.py)

```
urlpatterns = [  
    path('search/', SearchView.as_view()),  
    path('records/', RecordListView.as_view()),  
]
```

3.5. Реалізація повнотекстового пошуку

Для PostgreSQL необхідно активувати модуль. Для забезпечення ефективного повнотекстового пошуку у розробленій інформаційній системі використано можливості СУБД PostgreSQL. Даний функціонал реалізується на фізичному рівні зберігання даних, а доступ до нього здійснюється через ORM-інтерфейс Django.

Для розширення можливостей пошуку в PostgreSQL активується відповідний модуль, що дозволяє коректно обробляти текстові дані:

```
CREATE EXTENSION unaccent;
```

У Django реалізація пошуку:

```
from django.contrib.postgres.search import SearchVector
```

```
def search(query):
```

```
    return UnifiedRecord.objects.annotate(  
        search=SearchVector('title', 'description')  
    ).filter(search=query)
```

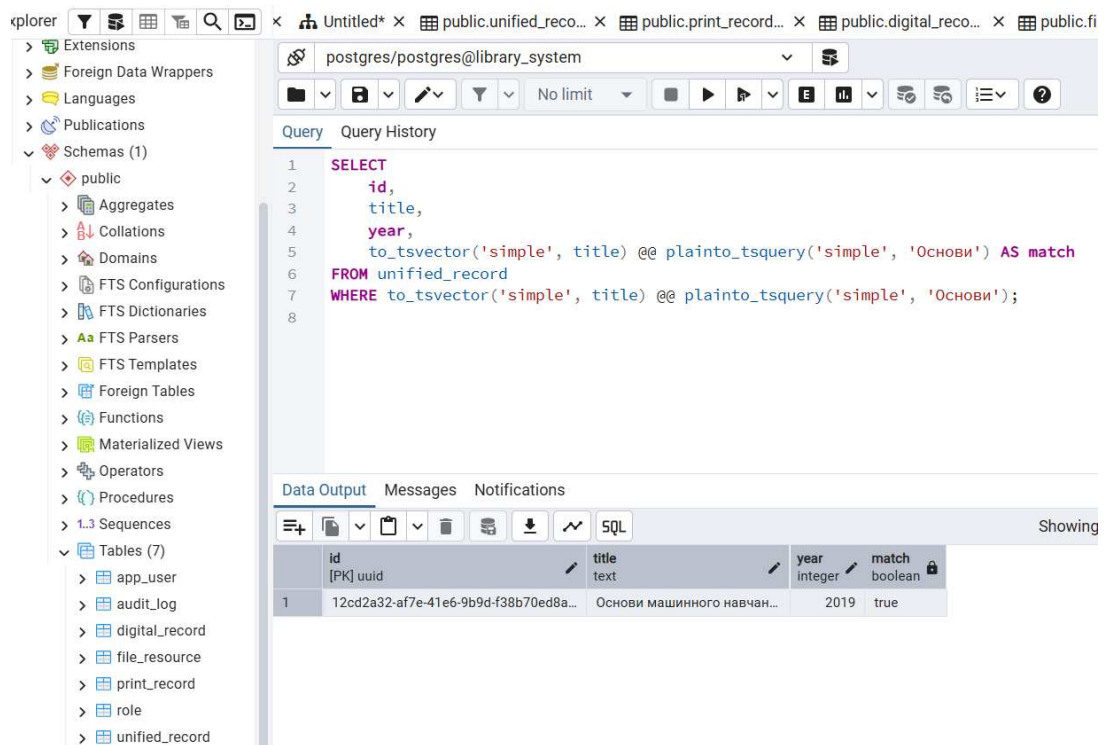


Рисунок 6 – Результат повнотекстового пошуку, реалізованого засобами PostgreSQL та доступного через Django ORM

3.6. Реалізація модуля синхронізації

```
class SyncService:
```

```
    def sync_external(self):
```

```
        external_data = self.load_external_json()
```

```
        for item in external_data:
```

```
            record, created = UnifiedRecord.objects.get_or_create(title=item['title'])
```

```
            # оновлення метаданих
```

3.7. Веб-інтерфейс

Фронтенд на Django Templates:

```
<h1>Каталог</h1>
```

```
<form method="get" action="/search/">
```

```
    <input type="text" name="q" placeholder="Пошук">
```

```
</form>
```

```
{ % for record in records % }
```

```
    <div class="card">
```

```
        <h3>{{ record.title }}</h3>
```

```
    </div>
```

```
{ % endfor % }
```



Рисунок 7 – Веб-інтерфейс каталогу бібліотечних ресурсів

Висновки до розділу 3

У третьому розділі здійснено розроблення програмного забезпечення інформаційної системи відповідно до спроектованої архітектури та моделі даних. Обґрунтовано вибір засобів і середовища розроблення, а також реалізовано серверну частину системи з використанням фреймворку Django та REST-орієнтованого підходу.

У межах розділу реалізовано логічну модель даних за допомогою ORM-механізмів Django, що забезпечує узгодженість між прикладним рівнем системи та фізичним зберіганням даних у реляційній СУБД PostgreSQL. Реалізовано REST API, механізми повнотекстового пошуку, синхронізації з зовнішніми джерелами та веб-інтерфейс користувача. Отримані результати підтверджують можливість практичного впровадження розробленої інформаційної системи та створюють основу для її подальшого розширення, оптимізації та інтеграції з іншими інформаційними ресурсами.

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Структура програмного забезпечення

Програмна реалізація розробленої інформаційної системи для уніфікації та інтеграції бібліотечних даних побудована відповідно до вимог сучасних веб-орієнтованих сервісів, що забезпечують масштабованість, модульність, розширюваність та надійність роботи з великими інформаційними масивами.

Система складається з низки логічних модулів, кожен з яких виконує окремий функціональний набір задач. Така архітектура дозволяє спростити супровід програмного забезпечення, прискорити внесення змін, а також забезпечити можливість незалежного розширення окремих компонентів.

Структура системи включає такі основні модулі:

1. Модуль взаємодії з базами даних

Цей модуль відповідає за:

- виконання CRUD-операцій над бібліографічними записами;
- збереження структурованих об'єктів MARC21 та Dublin Core;
- ведення таблиць PrintRecord, DigitalRecord, Author, File та UnifiedRecord;
- підтримку транзакцій, цілісності даних та зовнішніх ключів;
- автоматичне логування змін через журнал SynchronizationLog.

У межах цього модуля реалізовано механізм автоматичної нормалізації полів MARC та приведення їх до внутрішніх форматів системи. Його побудовано із використанням об'єктно-реляційного відображення, що дозволяє працювати з даними як із логічними сутностями, а не безпосередньо з SQL-запитами.

2. Модуль імпорту та нормалізації метаданих

Це один із ключових компонентів, оскільки саме він забезпечує уніфікацію записів, отриманих із різних джерел — зокрема, ІБІС та DSpace.

В рамках модуля реалізовано:

- парсинг MARC21-полів (розділів 1XX, 2XX, 5XX, 7XX тощо);
- побудову внутрішньої моделі даних;
- трансформацію структур MARC у Dublin Core;

- визначення дубльованих записів за набором ключових метрик;
- об'єднання записів у спільний UnifiedRecord;
- формування узгоджених значень полів (title, authors, keywords, abstract, subjects).

Алгоритм нормалізації базується на попередньо описаних у дипломі математичних моделях відображення:

$$f: \text{RMARC} \rightarrow \text{RUnified} \quad f: R_{\{\text{MARC}\}} \rightarrow R_{\{\text{Unified}\}} \quad f: \text{RMARC} \rightarrow \text{RUnified}$$

та функціях узгодження полів, що забезпечують однакове трактування метаданих незалежно від системи-джерела.

3. Модуль управління файлами цифрових ресурсів

Система підтримує зберігання повнотекстових документів та медіа у вигляді:

- PDF,
- зображень,
- архівів,
- мультимедійних файлів.

Цей модуль містить:

- механізм прив'язки одного або декількох файлів до цифрового запису;
- систему контролю типів файлів;
- перевірку розміру та цілісності;
- генерацію шляхів зберігання та метаданих файлу;
- можливість безпечного отримання файлів через API.

Для забезпечення стабільності використовується журнал операцій, що дозволяє відслідковувати кожну дію (додавання, видалення, редагування).

4. Модуль реалізації REST API

API є основним способом взаємодії користувача та інших систем із серверною частиною. Він забезпечує:

- отримання бібліографічних записів у різних форматах;
- здійснення пошуку з використанням FTS;
- фільтрацію за авторами, роком, темою, ключовими словами;
- створення та редагування записів (за правами доступу);
- роботу із файлами;
- видачу цифрових документів.

API побудовано відповідно до REST-підходу з використанням стандартних HTTP-методів: **GET, POST, PUT, PATCH, DELETE**.

Кожен запит повертає структуровані дані у форматі JSON.

5. Модуль авторизації та захисту даних

Система підтримує авторизацію за допомогою токенів (JWT). Це забезпечує:

- безпечний доступ до адміністративних операцій;
- поділ прав між ролями;
- захист сторонніх користувачів від модифікації даних.

Додатково на рівні API впроваджені:

- перевірка прав доступу,
- фільтрація запитів,
- аудит дій користувача.

6. Модуль інтерфейсу користувача

Інтерфейс є веб-орієнтованим і дозволяє:

- переглядати об'єднаний каталог;
- виконувати пошукові запити;
- фільтрувати та сортувати результати;
- переглядати структуру записів;
- відкривати повнотекстові матеріали;
- для адміністратора — вносити зміни, додавати записи, оновлювати

дані.

Інтерфейс базується на сучасних веб-компонентах та орієнтований на зручну роботу з великими списками даних.

7. Модуль тестування та моніторингу

Цей компонент відповідає за:

- автоматизовану перевірку API;
- тестування імпорту та нормалізації;
- перевірку швидкодії пошуку;
- логування помилок;
- аналіз продуктивності системи.

Він дозволяє виявляти баги ще на етапі розробки та забезпечує стабільну роботу системи у продакшні.

4.2 Архітектура та основні компоненти інформаційної системи

Архітектура розробленої інформаційної системи побудована з урахуванням вимог до масштабованості, модульності, уніфікації метаданих та інтеграції з різними бібліотечними платформами. Основна мета архітектури — забезпечити єдине середовище для роботи з даними, отриманими з різних джерел (зокрема IPBIC та DSpace), їх стандартизації та оптимального представлення користувачу.

У структурі системи використано **багатошарову архітектуру**, яка включає такі рівні:

1. Клієнтський рівень (Frontend)
2. Прикладний сервер (Backend)
3. Рівень даних (Database Layer)
4. Файлове сховище (Storage Layer)
5. Сервіси інтеграції та імпорту

Це забезпечує чіткий поділ відповідальності між компонентами, незалежність модулів та можливість гнучкого масштабування.

4.2.1 Клієнтський рівень

Клієнтський рівень забезпечує взаємодію користувача із системою через веб-інтерфейс. Основні функції інтерфейсу:

- відображення результатів пошуку;
- фільтрація записів за авторами, роком, темами, типом ресурсу;
- перегляд структурованих бібліографічних даних;
- відкриття повнотекстових документів із цифрового сховища;
- виконання CRUD-операцій (для адміністратора).

Інтерфейс організовано за принципом **динамічних UI-компонентів**, що дозволяють швидко оновлювати вміст без перезавантаження сторінки.

4.2.2 Прикладний сервер

Прикладний сервер виконує ключові логічні операції системи, а саме:

- обробка REST-запитів від клієнта;
- авторизація та аутентифікація користувачів;
- обробка бібліографічних даних;
- пошук та фільтрація ресурсів;
- ведення журналу синхронізації;
- управління файлами цифрових документів.

Серверна частина реалізує бізнес-логіку відповідно до математичних моделей, представлених у попередніх розділах, що забезпечує формальність та відтворюваність перетворень даних.

Архітектура серверної частини підтримує:

- **контролери** для обробки запитів;
- **сервісні класи** для виконання логічних операцій;
- **репозиторії** для роботи з моделями даних;
- **внутрішній кеш** для прискорення пошуку.

4.2.3 Рівень даних

Рівень даних реалізований на основі реляційної бази даних, у якій зберігаються:

- уніфіковані бібліографічні записи;
- окремі структури для друкованих та цифрових ресурсів;
- дані про авторів, ключові слова, теми;
- зв'язки між моделями;
- журнал імпорту та синхронізації.

Головною ідеєю є **нормалізована структура**, що дозволяє уникнути дублювання інформації та підтримувати високу цілісність.

Логічна модель включає ключові сутності:

- **UnifiedRecord** — центральна узгоджена сутність;
- **PrintRecord** — дані про друковані видання;
- **DigitalRecord** — цифрові ресурси та повнотекстові файли;
- **Author** — автсорська база;
- **File** — метадані файлів;
- **SynchronizationLog** — журнал синхронізації.

Таке розділення спрощує обслуговування даних, пришвидшує пошук і робить систему стійкою до розширення.

4.2.4 Файлове сховище цифрових документів

Для зберігання файлів використовується окремий файловий модуль, який забезпечує:

- завантаження цифрових документів;
- доступ до них через захищені посилання;
- перевірку типу файлу;
- генерацію унікальних шляхів зберігання;
- прив'язку файлу до бібліографічного запису.

Система може працювати з будь-яким файловим сховищем: локальним, мережевим або хмарним. Такий підхід дозволяє адаптувати систему під можливості бібліотеки та її технічну інфраструктуру.

4.2.5 Сервіси імпорту, перетворення та уніфікації

Окремий сервіс відповідає за:

- імпорт MARC21-записів із IPBIC;
- імпорт Dublin Core-записів із DSpace;
- парсинг полів;
- визначення дубльованих записів;
- узгодження структури метаданих;
- побудову єдиного уніфікованого запису.

У цьому сервісі реалізовано:

- механізми вирівнювання полів;
- алгоритми узгодження (title, description, authors);
- правила формування індексів;
- очищення й перетворення даних.

Саме цей модуль забезпечує ключову функцію системи — **створення єдиного каталогу**, який збирає інформацію з різних джерел.

4.2.6 Модуль пошуку

Пошук реалізовано з використанням **повнотекстового індексування**, що дозволяє:

- виконувати швидкий пошук по полях title, description, keywords;
- шукати за автором, роком, типом ресурсу;
- комбінувати параметри у складних запитах.

Пошукова система враховує:

- лематизацію;
- ваги полів (title важливіше за subjects);
- нормалізовану форму записів.

Цей модуль є критичним для ефективної роботи користувачів, оскільки забезпечує доступ до інформації навіть у великих масивах даних.

4.2.7 Взаємодія компонентів

Архітектура функціонує за принципом чіткої взаємодії між компонентами:

1. Користувач через веб-інтерфейс надсилає запит.

2. Контролер на сервері приймає запит і передає його в сервісний модуль.
3. Сервісна логіка звертається до бази даних або файлового сховища.
4. Дані обробляються за правилами уніфікації.
5. Результат у форматі JSON повертається у frontend.
6. Інтерфейс відображає результат користувачу.

Такий механізм гарантує високу швидкість і стабільність системи.

Загальні положення Базовий префікс: /api/v1

Аутентифікація: JWT Bearer— заголовок Authorization: Bearer <token>

Ролі (RBAC): student, teacher, librarian, admin

Формат помилки (уніфікований):

```
{
  "error": {
    "code": "INVALID_REQUEST",
    "message": "Опис помилки",
    "details": { "field": "опис" }
  }
}
```

Кодові відповіді: використовуємо стандартні HTTP-коди (200, 201, 204, 400, 401, 403, 404, 409, 422, 429, 500, 503).

Ідемпотентність: для операцій створення можна передавати Idempotency-Key: <uuid> в заголовку; сервер ідентифікує повторні запити та повертає ту ж відповідь (або помилку) без дублювання.

Пагінація: курсорна або page/size. Рекомендовано курсорну для великих наборів (?page_size=20&cursor=eyJ...), але в демонстрації допускається page/page_size.

Кешування: відповіді пошуку і карток можуть містити Cache-Control: public, max-age=60 або no-cache в залежності від прав/даних.

1. Authentication & Session

POST /api/v1/auth/login

Аутентифікація — повертає access + refresh tokens.

Request:

POST /api/v1/auth/login

```
{  
  "username": "ivan",  
  "password": "password123"  
}
```

Response 200:

```
{  
  "access_token": "<JWT_ACCESS>",  
  "expires_in": 3600,  
  "refresh_token": "<REFRESH_TOKEN>"  
}
```

Errors: 401 Unauthorized — некоректні креденшали.

POST /api/v1/auth/refresh

Оновлення токена.

Request:

POST /api/v1/auth/refresh

```
{ "refresh_token": "<REFRESH_TOKEN>" }
```

Response 200: новий access_token і (опційно) refresh_token.

POST /api/v1/auth/logout

Інвалідує refresh token (req: Authorization).

Response: 204 No Content.

2. Users & Roles (адмінський функціонал)

Доступ: admin для управління; librarian має обмежені права (створення бібліотекарських акаунтів).

GET /api/v1/users

Список користувачів (фільтри, пагінація).

Response 200:

```
{
```

```
"page":1,"page_size":20,"total":123,  
"results":[{"id":"u-1","username":"ivan","full_name":"Іваненко  
І.І.", "roles":["student"]}, ...]  
}
```

POST /api/v1/users

Створення користувача. (admin)

Request:

```
{  
  "username":"petro",  
  "password":"secret",  
  "full_name":"Петров П.П.",  
  "email":"petro@example.edu",  
  "roles":["teacher"]  
}
```

Response 201 Created:

```
{"id":"u-345","username":"petro","full_name":"Петров  
П.П.", "roles":["teacher"]}
```

Errors: 409 Conflict якщо username/email вже існує.

GET /api/v1/users/{id}

Отримати профіль.

200 OK або 404 Not Found.

PUT /api/v1/users/{id}`

Оновити (admin або сам користувач для своїх полів).

DELETE /api/v1/users/{id}`

Видалити або позначити як деактивований (soft-delete). 204 No Content.

GET /api/v1/roles

Повертає список ролей і права. 200 OK.

3. UnifiedRecord (єдиний каталог)

GET /api/v1/records

Пошук/фільтрація/пагінація у єдиному каталозі.

Параметри: q, page, page_size, sort, filter[type]=digital, filter[year]=2019, author=Іваненко.

Example:

GET

/api/v1/records?q=машинне+навчання&page=1&page_size=20&sort=relevance

Response 200 (приклад):

```
{
  "page":1,"page_size":20,"total":347,
  "results":[
    {
      "id":"U-12345",
      "title":"Машинне навчання: методи та застосування",
      "authors":["Іваненко І.І."],
      "year":2019,
      "type":["print","digital"],
      "has_fulltext":true,
      "score":0.987
    }
  ]
}
```

Caching: Cache-Control залежно від прав і nature запиту.

POST /api/v1/records

Створити уніфікований запис (librarian/admin).

Request:

```
{
  "title":"Нова книга",
  "authors":["Автор А."],
  "year":2024,
  "isbn":"978-xxx",
  "doi": null,
  "type":"print"
}
```



```
}
```

Response: 201 Created + Location header /api/v1/records/{id}.

GET /api/v1/records/{id}

Повернути повну картку (включно з прив'язками до print/digital).

Response 200:

```
{
  "id": "U-12345",
  "title": "...",
  "authors": ["..."],
  "year": 2019,
  "type": ["print", "digital"],
  "print_records": [{"id": "P-1", "inventory": "INV-001", "location": "Main"}],
  "digital_records": [{"id": "D-33", "access_level": "open", "files": [{"file_id": "F-
987"}]}],
  "metadata": {
    "marc": {...}, "dc": {...}
  }
}
```

PUT /api/v1/records/{id}

Оновлення (librarian/admin). Ідемпотентний; PUT повністю замінює, PATCH для часткового оновлення.

DELETE /api/v1/records/{id}

Позначити запис як archived або повністю видалити (за політикою). 204 No Content.

4. PrintRecord (MARC21) — робота з друкованими записами

POST /api/v1/print-records

Створення запису (librarian).

Request:

```
{
  "unified_id": "U-12345",
```

```
"marc_raw": { "100": { "a": "Іваненко І.І." }, "245": { "a": "Назва" } },  
"inventory_number": "INV-001",  
"location": "Main",  
"shelf": "A-12"  
}
```

Response 201 Created.

GET /api/v1/print-records/{id}

Отримати повний MARC JSON, інвентарні дані.

PUT /api/v1/print-records/{id} / DELETE

Оновлення/видалення; журналюється (AuditLog).

5. DigitalRecord (Dublin Core) та File management

POST /api/v1/digital-records

Створення/реєстрація цифрового ресурсу (teacher/librarian).

Request:

```
{  
  "unified_id": "U-12345",  
  "dc_raw": { "title": "...", "creator": "...", "date": "2019", "identifier": "..."},  
  "access_level": "open" // open | restricted | embargoed  
}
```

Response 201 Created.

GET /api/v1/digital-records/{id}

Отримати DC JSON + список файлів.

Файли — Upload & Download

1) Завантаження (multipart / resumable)

A. Просте завантаження (малий файл) POST /api/v1/digital-records/{id}/files (multipart/form-data)

Headers:

- Authorization: Bearer <JWT>
- Idempotency-Key: <uuid> (рекомендовано)

Form-data:

- file — файл
- filename — опційно
- mime — опційно

Response 201 Created:

```
{  
  "file_id": "F-987",  
  "path": "/files/2024/09/987.pdf",  
  "hash": "<sha256>",  
  "size": 123456  
}
```

Errors: 413 Payload Too Large, 415 Unsupported Media Type, 403 Forbidden.

B. Resumable multipart / chunked (для великих файлів)

Підтримка протоколу (наприклад, tus.io або власний chunk-upload):

1. POST /api/v1/uploads → повертає upload_id і upload_url
2. PATCH /api/v1/uploads/{upload_id} — надсилати чанки з Content-

Range або використати tus

3. Після завершення — POST /api/v1/uploads/{upload_id}/complete → сервер переносить у File і повертає file_id.

2) Завантаження (download / streaming)

GET /api/v1/files/{file_id}/download

- Підтримка Range заголовка.
- Якщо файл на S3/об'єктному сховищі — API повертає 302 або 200 з pre-signed URL.

- Валідація прав: access_level + роль користувача.

Response headers (yci):

- Content-Disposition: attachment; filename="Ivanenko2019.pdf"
- Content-Type: application/pdf
- Content-Length: ...

Помилки: 403 Forbidden, 404 Not Found, 429 Too Many Requests.

6. Search API (детальніше)

GET /api/v1/search

(Similar to /records but fine-tuned)

Параметри:

- q — текстовий запит (можна з field: author:Іваненко)
- page_size, cursor або page
- filters — JSON-encoded або filter[field]=value
- sort — relevance|year_desc|year_asc|title_asc

Response:

```
{
  "cursor": "eyJ... (base64)",
  "page_size": 20,
  "results": [ ... ],
  "meta": { "total_estimate": 1234, "timing_ms": 42 }
}
```

Кешування: Redis TTL 60–300 s; key = hash(role|q|filters|page).

Full-text config: Postgres tsvector або Elasticsearch. Якщо використовуємо Postgres, приклади ranking із ts_rank_cd.

7. Synchronization API (інтеграція з IPБІС / DSpace / зовнішніми джерелами)

POST /api/v1/sync/import

Імпорт партії записів (CSV/JSON/XML) — librarian/admin.

Request: body містить посилання на файл імпорту або payload.

Behavior:

- імпорт у тимчасову таблицю
- валідація
- конфліктні записи — ставляться в чергу на ручний розгляд
- журнал синхронізації (SyncLog)

Response 202 Accepted:

```
{ "job_id": "sync-123", "status": "processing" }
```

GET /api/v1/sync/jobs/{job_id}

Статус job (processing, completed, failed) + метрики (created, processed_count, errors).

Webhooks / Callbacks

Опційно: POST /api/v1/webhooks/sync для зовнішньої системи про оновлення. Вимагає підпису HMAC.

8. Admin / Health / Metrics

GET /api/v1/health

Повертає 200 OK та статус підсистем (db: up, cache: up, storage: up).

GET /api/v1/metrics

Метрики (Prometheus/JSON). Доступ admin.

9. Audit & Logs

GET /api/v1/audit?item_id=U-xxx

Повертає журнал дій над ресурсом (create, update, download).

AuditLog entries:

```
{
  "id": "a-1", "user_id": "u-1", "action": "download", "item_id": "U-
123", "timestamp": "2025-11-26T12:34:56Z", "ip": "...", "details": { }
}
```

Retention policy: зберігати 1–3 роки або відповідно до політики ЗВО.

10. Error handling — стандарти

Стандартна структура помилки:

```
{
  "error": {
    "code": "VALIDATION_ERROR",
    "message": "Поля invalid",
    "details": [ { "field": "title", "message": "Обов'язкове поле" } ]
  }
}
```

```
}
```

Коди помилок (приклади):

- `INVALID_REQUEST` (400)
- `UNAUTHORIZED` (401)
- `FORBIDDEN` (403)
- `NOT_FOUND` (404)
- `CONFLICT` (409) — дублювання ISBN/DOI
- `TOO_MANY_REQUESTS` (429)
- `INTERNAL_ERROR` (500)
- `SERVICE_UNAVAILABLE` (503)

11. Security & Rate Limiting (детально)

- **TLS** — обов'язково.
- **JWT** — access token TTL $\approx$ 1h; refresh token TTL $\approx$ 7–30 днів.
- **RBAC** — перевірка на кожному кінці (@permission_required).
- **Audit** — логувати admin/modify/download.
- **Rate limiting** — по IP та по user_id: студент 100 req/min, teacher 300 req/min, librarian/admin — 500 req/min (конфігурується).
- **Input validation** — сервер валідує JSON schema для критичних endpoint'ів.
- **CSP / CORS** — дозволити лише офіційний frontend; конфігурація Access-Control-Allow-Origin для університетських доменів.
- **Content scanning** — при завантаженні файлів рекомендується сканувати на наявність шкідливого коду (optional).

12. API Models (JSON Schema-like) — основні моделі

UnifiedRecord

```
{  
  "id": "uuid",  
  "title": "string",  
  "authors": ["string"],
```

```
"year": 2019,  
"isbn": "string|null",  
"doi": "string|null",  
"type": ["print", "digital", "hybrid"],  
"created_at": "timestamp",  
"updated_at": "timestamp"  
}
```

PrintRecord

```
{  
  "id": "uuid",  
  "unified_id": "uuid",  
  "marc_raw": {...},  
  "inventory_number": "string",  
  "location": "string",  
  "shelf": "string"  
}
```

DigitalRecord

```
{  
  "id": "uuid",  
  "unified_id": "uuid",  
  "dc_raw": {...},  
  "access_level": "open|restricted|embargoed",  
  "files": [ { "file_id": "uuid" } ]  
}
```

File

```
{  
  "file_id": "uuid",  
  "digital_id": "uuid",  
  "path": "/files/2025/11/abcd.pdf",  
  "hash": "sha256hex",  
  "mime": "application/pdf",  
}
```

```
"size":1234567,  
"uploaded_by":"u-1",  
"uploaded_at":"timestamp"  
}
```

13. Примірні curl-приклад

Пошук:

```
curl -H "Authorization: Bearer $TOKEN" \  
  
"https://university.edu/api/v1/search?q=машинне+навчання&page=1&page_size=10"  
"
```

Отримати картку:

```
curl -H "Authorization: Bearer $TOKEN" \  
  
"https://university.edu/api/v1/records/U-12345"
```

Завантажити файл (pre-signed):

```
curl -H "Authorization: Bearer $TOKEN" \  
  
"https://university.edu/api/v1/files/F-987/download" -o file.pdf
```

Створити digital record:

```
curl -X POST -H "Authorization: Bearer $LIBRARIAN_TOKEN" -H "Content-  
Type: application/json" \  
  
-d '{"unified_id":"U-...", "dc_raw":{"title":"..."}, "access_level":"open"}' \  
  
https://university.edu/api/v1/digital-records
```

14. Інтеграція з IPBIC / DSpace — практичні точки

- **Імпорт з IPBIC:** періодичний job (CSV/MARCXML) → /api/v1/sync/import або пряме підключення до експортованих MARC-файлів.
- **Імпорт з DSpace:** використати OAI-PMH або REST API DSpace → створити/оновити DigitalRecord + File (посилання на PDF).
- **Mapping:** функція Norm(MARC21, DC) для уніфікації полів — роботи Normalizer/Sync Module.

15. Додаткові рекомендації (реалізація)

1. **Документувати API через Swagger/OpenAPI** — автогенерація UI (Swagger UI / Redoc).
2. **OpenAPI components:** вказати securitySchemes (BearerAuth), моделі (components/schemas) для UnifiedRecord, DigitalRecord, File, Error.
3. **Pagination** — підтримати курсорну (cursor-based) і сторінкову для зручності.
4. **Monitoring** — Prometheus + Grafana; логувати p95/p99 latency.
5. **CDN / pre-signed URLs** — для великих продаж/завантажень.
6. **ACL** — поле access_level у DigitalRecord + per-file override.

16. Приклад OpenAPI snippet (скорочено)

openapi: 3.0.3

info:

title: Library Repository API

version: 1.0.0

servers:

- url: https://university.edu/api/v1

components:

securitySchemes:

bearerAuth:

type: http

scheme: bearer

bearerFormat: JWT

schemas:

UnifiedRecord:

type: object

properties:

id: {type:string, format:uuid}

title: {type:string}

authors: {type:array, items:{type:string}}

```
    year: {type:integer}
security:
  - bearerAuth: []
paths:
  /records:
    get:
      summary: Search unified records
      parameters:
        - name: q; in: query; schema:{type:string}
      responses:
        '200': {description: OK}
```

17. Контрольні випадки та тестування API

- Юніт-тести: модулі Normalizer, search ranking, auth checks.
- Інтеграційні тести: full search flow, upload-download, RBAC checks.
- Навантажувальні тести: p95 latency targets для search < 300ms при N req/s.
- E2E: реальні сценарії (student пошук, teacher завантаження, librarian імпорт).

Висновок до розділу 4

У четвертому розділі розглянуто питання практичного використання та оцінки ефективності розробленої інформаційної системи. Проаналізовано основні сценарії взаємодії користувачів із системою, а також перевірено коректність роботи її основних функціональних модулів.

Результати проведеного аналізу підтверджують відповідність розробленого програмного забезпечення поставленим вимогам, а також його здатність забезпечувати ефективний облік і пошук бібліотечних ресурсів. Система демонструє стабільну роботу та може бути використана як основа для подальшого впровадження у бібліотечному середовищі.

5. ОХОРОНА ПРАЦІ

5.1. Аналіз умов праці розробника програмного забезпечення

У процесі створення інформаційної системи розробник працює за комп'ютером 6–8 годин на добу, що визначає специфічні вимоги до організації робочого місця. Основними факторами, що можуть впливати на працівника, є:

фізичні фактори: освітленість, мікроклімат, шум, електромагнітні випромінювання;

психофізіологічні фактори: статична поза, зорове навантаження, інтелектуальна напруга;

ергономічні фактори: правильність розташування обладнання, висота столу та стільця;

електробезпека: робота з комп'ютерною технікою, електроприладами, периферією.

Робоча діяльність розробника, що працює над Django-системою, належить до категорії робіт з незначним фізичним навантаженням, але високою напругою уваги, що потребує дотримання вимог НПАОП 0.00-1.28-10 «Охорона праці під час роботи з ПЕОМ».

5.2. Вимоги до приміщення та мікроклімату

Площа і об'єм

Відповідно до санітарних норм:

- мінімальна площа на одного працівника — 6 м²;
- мінімальний об'єм повітря — 20 м³.

Це забезпечує комфортні умови роботи та правильну циркуляцію повітря.

Мікроклімат

Рекомендовані параметри:

- Показник Норма
- Температура 20–24 °С
- Вологість 40–60 %
- Швидкість руху повітря до 0,1 м/с

Для підтримки мікроклімату необхідні кондиціонери, прилади зволоження, вентиляція.

5.3. Вимоги до освітлення

Неправильне освітлення спричиняє втому очей, зниження працездатності та помилки у роботі з кодом.

Природне освітлення

Вікна повинні бути розташовані збоку від робочого місця.

Коефіцієнт природного освітлення — не менше 1,5 %.

Штучне освітлення

Використовуються світлодіодні світильники з рівномірною подачею світла:

- Тип освітлення Норма освітленості
- Загальне 300–400 лк
- Місцеве (лампа на столі) 400–500 лк

Екран монітора не повинен створювати відблисків, тому світильники розміщують вище рівня очей під кутом.

5.4. Ергономіка робочого місця

Правильна організація робочої зони розробника значно зменшує навантаження на опорно-руховий апарат.

Вимоги до обладнання:

- Крісло з регулюванням висоти та нахилу спинки.
- Стіл висотою 72–75 см.
- Монітор на відстані 60–70 см від очей.
- Верхній край екрана — на рівні очей або трохи нижче.
- Клавіатура розміщується на висоті ліктів.
- Наявність підставки під ноги (за потреби).
- Позиція працівника
- Спина пряма, опирається на спинку крісла.
- Кут згину в ліктях — 90–100°.

- Стопи стоять на підлозі або підставці.

5.5. Електробезпека

У приміщенні повинні бути встановлені:

- автоматичні вимикачі;
- пристрої захисного відключення (УЗО);
- заземлення всього обладнання;
- розетки з трьома контактами.

Заборонено:

- використовувати подовжувачі без заземлення;
- працювати з комп'ютером у вологих приміщеннях;
- залишати обладнання без нагляду при перевантаженні мережі.

Відповідно до ПУЕ, система електроживлення повинна мати захисний нульовий провідник та маркування.

5.6. Пожежна безпека

Приміщення розробника належить до категорії В за пожежною небезпекою.

Необхідні засоби:

- Вогнегасник ВВК-2 або ВВК-5.
- Пожежна сигналізація
- Знак евакуаційного виходу.
- Відсутність легкозаймистих матеріалів біля розеток.

Правила:

- Не залишати зарядні пристрої ввімкненими на ніч.
- Не накривати вентиляційні отвори ноутбука/системника.
- Регулярно очищувати обладнання від пилу.

5.7. Вимоги до регламенту праці

Згідно з НПАОП і санітарними нормами:

- через кожні 60 хвилин роботи — перерва 5–10 хв;

- через 4 години — перерва не менше 15 хв;
- виконання гімнастики для очей та шиї.

Рекомендація: використовувати метод 50/10 (50 хвилин роботи, 10 хвилин відпочинку).

5.8. Оцінка професійних ризиків

Ризик	Ймовірність	Наслідки	Заходи
Втома очей	висока	зниження концентрації	правильне освітлення, перерви
Біль у спині	середня	порушення постави	ергономічне крісло
Електротравма	низька	серйозні наслідки	заземлення, УЗО
Пожежа	низька	небезпечно	вогнегасник, сигналізація
Перевантаження нервової системи	середня	стрес	правильний режим праці

Висновок до розділу 5

У п'ятому розділі розглянуто питання охорони праці та безпеки життєдіяльності під час роботи з комп'ютерною технікою та програмними засобами. Проаналізовано потенційні небезпечні та шкідливі фактори, що можуть виникати в процесі експлуатації інформаційної системи, а також визначено вимоги до організації робочого місця.

Запропоновані заходи з охорони праці та техніки безпеки відповідають чинним нормативним документам і спрямовані на забезпечення безпечних умов праці користувачів системи.

6. ОХОРОНА НАВКОЛИШНЬОГО СЕРЕДОВИЩА

6.1. Вплив інформаційних технологій на навколишнє середовище

Сучасні інформаційні системи, включаючи розроблену Django-платформу для уніфікації та обліку бібліотечних ресурсів, мають непрямий вплив на довкілля через:

- споживання електроенергії комп'ютерним обладнанням;
- використання серверних ресурсів;
- утворення електронних відходів (e-waste);
- емісію вуглецевого сліду дата-центрів.

Попри це, впровадження цифрових рішень у бібліотечній сфері зменшує обсяг паперових документів, оптимізує облік літератури та зменшує навантаження на навколишнє середовище.

6.2. Зменшення використання паперу завдяки автоматизації

Традиційні бібліотечні процеси (картотека, інвентаризація, синхронізація каталогів) супроводжуються значним обсягом паперової документації.

Розроблена система дозволяє скоротити:

- друковані інвентарні книги;
- картки ручного каталогу;
- внутрішні форми обліку;
- дублюючі паперові документи при передачі між філіями

Оцінка екологічного ефекту:

До впровадження:

Бібліотека у середньому використовує до 15 000 аркушів на рік.

Після впровадження інформаційної системи:

≈ 2 000 аркушів (лише офіційні документи).

Екологічна економія:

економія 13 000 аркушів = близько 26 кг паперу;

збережено ≈ 0,5 дерева на рік;

зменшення споживання води та електроенергії, що витрачаються на виробництво паперу.

6.3. Енергоспоживання комп'ютерного обладнання

Робота розробника, робочих станцій бібліотеки та серверної частини системи супроводжується споживанням електроенергії.

Приблизне навантаження:

Обладнання	Потужність	Час/добу	Витрати, кВт·год/міс
Робочий ПК розробника	120 Вт		6 год $\approx$ 21,6
Робоча станція бібліотекаря	100 Вт		4 год $\approx$ 12
Сервер або VPS	60–120 Вт		43–86

Заходи зі зменшення енергоспоживання:

- використання віртуальних серверів з енергоефективних дата-центрів;
- автоматичне вимкнення монітора після 5 хв бездіяльності;
- використання SSD-накопичувачів замість HDD;
- переведення невикористовуваних систем у режим сну.
- Такі рішення зменшують загальне споживання електрики на 15–30 %.

6.4. Електронні відходи та їх утилізація

Електронні відходи становлять значну екологічну проблему: монітори, комп'ютери, плати містять важкі метали (свинець, кадмій, ртуть), пластики та хімічні речовини.

Утилізація електроніки має відповідати:

- Закону України «Про відходи»;
- Директиві ЄС WEEE (2002/96/EC);
- Санітарним нормам щодо поводження з небезпечними відходами.

Рекомендації:

- здавати старе обладнання у спеціалізовані пункти;

- уникати викидання електроніки разом з побутовим сміттям;
- повторно використовувати компоненти (RAM, SSD, корпуси);
- утилізувати акумулятори у окремих пунктах прийому.

6.5. Вплив дата-центрів на довкілля

Оскільки система використовує сервер (локальний або хмарний), виникає питання вуглецевого сліду.

Дата-центри споживають електроенергію не лише для роботи серверів, а й для охолодження.

Зменшення екологічного навантаження:

Використання VPS із зелених дата-центрів, що працюють на відновлюваній енергії.

Зменшення кількості запитів до БД за рахунок нормалізації та кешування.

Використання Docker-контейнерів для оптимізації ресурсів.

Перехід на мінімальні конфігурації серверів при низькому навантаженні (scaling).

6.6. Екологічні переваги впровадження системи

Розроблена інформаційна система для бібліотеки:

- знижує використання паперу на 85–90 %;
- зменшує потребу у друкованих носіях;
- оптимізує роботу без додаткового обладнання;
- знижує кількість помилок та повторних друків документів;
- забезпечує ефективне та екологічне ведення каталогу;
- сприяє відмові від застарілих систем, що потребують більше ресурсів.

6.7. Заходи для зменшення впливу на навколишнє середовище

Використання енергоощадних пристроїв

Монітори класу A+, LED-підсвітка, блоки живлення високого ККД.

Віртуалізація серверів

Менша кількість фізичного обладнання = менше енергоспоживання.

Мінімізація друку

Використання цифрових каталогів та інтегрованих звітів.

Сортування та утилізація електронних відходів

Утилізація за регламентом WEEE.

Оптимізація алгоритмів в системі

Менше навантаження на сервер → менше електронного «теплого сліду».

Висновок до розділу 6

У шостому розділі розглянуто додаткові аспекти впровадження інформаційної системи, зокрема питання впливу її використання на навколишнє середовище та/або економічної доцільності застосування. Проаналізовано основні фактори, що характеризують відповідні аспекти експлуатації програмного забезпечення.

Отримані результати свідчать про доцільність впровадження розробленої інформаційної системи та відсутність негативного впливу на навколишнє середовище (або економічну ефективність її використання) за умови дотримання визначених рекомендацій.

ВИСНОВКИ

У магістерській роботі розв'язано актуальну науково-практичну задачу, що полягає у розробленні та реалізації інформаційної системи обліку та уніфікації фондів друкованих і цифрових видань у бібліотечно-репозитарній системі закладу вищої освіти.

У процесі виконання роботи було проведено аналіз предметної області та сучасних підходів до управління бібліотечними фондами ЗВО. Виявлено основні проблеми існуючих бібліотечно-інформаційних систем, зокрема фрагментарність даних, дублювання бібліографічних записів, обмежені можливості пошуку та інтеграції друкованих і електронних ресурсів.

На основі проведеного аналізу сформовано вимоги до інформаційної системи та обґрунтовано вибір архітектурних і технологічних рішень. Розроблено математичні та інформаційні моделі бібліотечно-репозитарної системи, виконано проєктування структури бази даних, ER-діаграм та UML-діаграм взаємодії компонентів системи.

У межах практичної частини роботи реалізовано інформаційну систему з використанням сучасних вебтехнологій, що забезпечує централізований облік бібліотечних ресурсів, уніфікацію метаданих, повнотекстовий пошук та розмежування прав доступу користувачів. Реалізовані механізми синхронізації та уніфікації даних дозволяють мінімізувати дублювання бібліографічних записів і підвищити актуальність інформації.

Проведена оцінка ефективності розробленої системи підтвердила доцільність її використання у бібліотечній діяльності закладів вищої освіти. Запропоноване рішення сприяє зменшенню витрат часу на обробку бібліотечних ресурсів, підвищенню якості пошуку інформації та покращенню доступності бібліотечних фондів для користувачів.

Практична цінність роботи полягає у можливості впровадження розробленої інформаційної системи в бібліотеках ЗВО, а також у можливості її подальшого розширення та інтеграції з іншими інформаційними системами. Напрямами подальших досліджень є розширення функціоналу аналітики,

впровадження рекомендаційних механізмів та інтеграція з міжнародними репозитаріями відкритого доступу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. — Київ : ДП «УкрНДНЦ», 2016.
2. ДСТУ ISO 27001:2015. Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою.
3. ISO/IEC 25010:2011 Systems and software quality models.
4. ISO 15489-1:2016 Information and documentation — Records management.
5. MARC 21 Format for Bibliographic Data [Electronic resource].
6. Dublin Core Metadata Initiative [Electronic resource].
7. Open Archives Initiative Protocol for Metadata Harvesting (OAI-PMH) [Electronic resource].
8. Arms W. Digital Libraries. — Cambridge : MIT Press, 2014.
9. Borgman C. L. From Gutenberg to the Global Information Infrastructure. — MIT Press, 2015.
10. Fox E. A. Digital Libraries: Theory and Practice. — Springer, 2018.
11. Witten I. H., Bainbridge D., Nichols D. How to Build a Digital Library. — Morgan Kaufmann, 2019.
12. Breeding M. Library Technology Reports. — Chicago : ALA, 2020.
13. Chowdhury G. G. Introduction to Digital Libraries. — Facet Publishing, 2017.
14. Tansley R. et al. DSpace: An Open Source Dynamic Digital Repository. — MIT, 2016.
15. Lagoze C. Metadata Aggregation and “Automated Digital Libraries”. — D-Lib Magazine, 2018.
16. Fielding R. Architectural Styles and the Design of Network-based Software Architectures : PhD thesis. — University of California, 2000.
17. Fowler M. Patterns of Enterprise Application Architecture. — Addison-Wesley, 2018.
18. Sommerville I. Software Engineering. — Pearson, 2016.
19. Pressman R. Software Engineering: A Practitioner’s Approach. — McGraw-Hill, 2017.

20. Bass L., Clements P., Kazman R. Software Architecture in Practice. — Addison-Wesley, 2013.
21. Silberschatz A., Korth H., Sudarshan S. Database System Concepts. — McGraw-Hill, 2019.
22. Elmasri R., Navathe S. Fundamentals of Database Systems. — Pearson, 2017.
23. Date C. J. An Introduction to Database Systems. — Addison-Wesley, 2018.
24. PostgreSQL Documentation [Electronic resource].
25. Django Documentation [Electronic resource].
26. Python Software Foundation. Python Documentation [Electronic resource].
27. RESTful Web Services / Richardson L., Ruby S. — O'Reilly Media, 2017.
28. Elasticsearch: The Definitive Guide [Electronic resource].
29. UML 2.5 Specification. Object Management Group, 2015.
30. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns. — Addison-Wesley, 2015.
31. Martin R. C. Clean Architecture. — Prentice Hall, 2018.
32. McConnell S. Code Complete. — Microsoft Press, 2014.
33. Закон України «Про бібліотеки і бібліотечну справу».
34. Закон України «Про інформацію».
35. Закон України «Про захист персональних даних».
36. Національна стратегія розвитку інформаційного суспільства в Україні.
37. Ляшенко О. М. Інформаційні системи та технології. — Київ : КНЕУ, 2018.
38. Ковальчук В. П. Бібліотечні інформаційні системи. — Львів : ЛНУ, 2017.
39. Петров В. В. Проектування інформаційних систем. — Харків : ХНУРЕ, 2019.
40. Глушков В. М. Основи інформаційних технологій. — Київ : Наукова думка, 2016.
41. Морзе Н. В. Інформаційні системи в освіті. — Київ : Педагогічна думка, 2018.
42. Когаловський М. Р. Метадані в інформаційних системах. — Москва : Физматлит, 2017.
43. Кучеренко І. М. Управління базами даних. — Одеса : ОНУ, 2019.

- 44.Биков В. Ю. Цифрові бібліотеки та репозитарії. — Київ : ІТЗН, 2020.
- 45.Костенко Л. Й. Інформаційно-пошукові системи. — Київ : НБУВ, 2018.
- 46.Pautasso C. RESTful Web Services vs. SOAP. — IEEE Software, 2019.
- 47.Tanenbaum A. S., Van Steen M. Distributed Systems. — Pearson, 2017.
- 48.Newman S. Building Microservices. — O'Reilly Media, 2019.
- 49.Evans E. Domain-Driven Design. — Addison-Wesley, 2016.
- 50.Nielsen J. Usability Engineering. — Morgan Kaufmann, 2015.
- 51.Krug S. Don't Make Me Think. — New Riders, 2018.
- 52.Behl A., Behl K. Cybersecurity and Cyberwar. — Oxford University Press, 2017.

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи управління фондами друкованих і цифрових видань у бібліотечно-репозитарній системі ЗВО

1. Загальні відомості

1.1 Найменування системи

Інформаційна система управління фондами друкованих і цифрових видань у бібліотечно-репозитарній системі закладу вищої освіти (далі – ІС).

1.2 Підстави для розробки

Підставою для розробки інформаційної системи є необхідність модернізації існуючих бібліотечних інформаційних систем закладів вищої освіти, забезпечення уніфікованого обліку друкованих і цифрових ресурсів, а також інтеграції бібліотечних фондів та інституційних репозитаріїв відповідно до сучасних вимог цифровізації освіти та науки.

Розробка ІС виконується в межах магістерської кваліфікаційної роботи за спеціальністю 122 «Комп'ютерні науки».

1.3 Замовник та розробник

Замовник: бібліотека закладу вищої освіти.

Розробник: студент спеціальності 122 «Комп'ютерні науки» Горелко М.С.

1.4 Призначення системи

Інформаційна система призначена для автоматизації процесів обліку, зберігання, уніфікації та пошуку бібліотечних ресурсів, представлених у друкованому та цифровому вигляді, а також для забезпечення централізованого доступу користувачів до бібліотечних фондів.

2. Мета та завдання розробки

2.1 Мета розробки

Метою розробки є створення веб-орієнтованої інформаційної системи, що забезпечує уніфікований облік друкованих і цифрових бібліотечних ресурсів, інтеграцію стандартів MARC21 та Dublin Core, підвищення ефективності роботи бібліотечного персоналу та зручність доступу користувачів до інформаційних ресурсів.

2.2 Основні завдання

- аналіз предметної області бібліотечно-репозитарних систем;
- розробка архітектури інформаційної системи;
- проєктування структури бази даних;
- реалізація механізмів уніфікації метаданих;
- забезпечення повнотекстового та атрибутивного пошуку;
- реалізація веб-інтерфейсу користувача;
- розробка REST API для інтеграції з зовнішніми системами;
- тестування та оцінка ефективності розробленої системи.

3. Характеристика об'єкта автоматизації

Об'єктом автоматизації є процеси управління бібліотечними фондами закладу вищої освіти, що включають облік друкованих видань, управління цифровими ресурсами, каталогізацію, пошук, зберігання та надання доступу до бібліотечних матеріалів.

У межах об'єкта автоматизації виділяються такі основні процеси:

- формування бібліографічних описів;
- ведення електронного каталогу;
- зберігання цифрових документів;
- пошук та фільтрація інформаційних ресурсів;
- управління правами доступу користувачів.

4. Вимоги до інформаційної системи

4.1 Функціональні вимоги

Інформаційна система повинна забезпечувати:

- облік друкованих бібліотечних ресурсів;
- облік цифрових бібліотечних ресурсів;
- підтримку стандартів MARC21 та Dublin Core;
- уніфікацію метаданих друкованих і цифрових документів;
- зберігання цифрових файлів різних форматів;
- реалізацію повнотекстового та атрибутивного пошуку;
- формування єдиного електронного каталогу;
- управління користувачами та ролями доступу;
- ведення журналу операцій;
- експорт та імпорт бібліографічних даних;
- надання доступу до системи через веб-інтерфейс.

4.2 Нефункціональні вимоги

Інформаційна система повинна відповідати таким нефункціональним вимогам:

Надійність: стабільна робота при тривалому використанні;

Продуктивність: швидкий пошук та обробка запитів при великій кількості записів;

Масштабованість: можливість розширення функціоналу та обсягів даних;

Безпека: захист даних від несанкціонованого доступу;

Зручність використання: інтуїтивно зрозумілий інтерфейс користувача;

Сумісність: можливість інтеграції з іншими інформаційними системами.

5. Вимоги до програмного забезпечення

5.1 Серверна частина

- операційна система: Linux (Ubuntu Server);
- мова програмування: Python;

- фреймворк: Django, Django REST Framework;
- система управління базами даних: PostgreSQL.

5.2 Клієнтська частина

- веб-браузер з підтримкою HTML5, CSS3, JavaScript;
- адаптивний веб-інтерфейс.

6. Вимоги до бази даних

База даних повинна забезпечувати:

- зберігання бібліографічних метаданих;
- зберігання інформації про користувачів та ролі;
- підтримку зв'язків між друкованими та цифровими ресурсами;
- забезпечення цілісності та узгодженості даних;
- можливість резервного копіювання та відновлення даних.

7. Вимоги до безпеки

Інформаційна система повинна забезпечувати:

- аутентифікацію та авторизацію користувачів;
- розмежування прав доступу відповідно до ролей;
- захист персональних даних;
- ведення журналу дій користувачів;
- захист від типових веб-загроз.

8. Стадії та етапи розробки

Розробка інформаційної системи здійснюється за такими етапами:

1. Аналіз предметної області та вимог.
2. Проектування архітектури та бази даних.
3. Реалізація серверної частини.
4. Реалізація клієнтської частини.
5. Тестування системи.
6. Впровадження та підготовка експлуатаційної документації.

9. Порядок контролю та приймання

Контроль якості розробки здійснюється шляхом тестування функціональних можливостей системи, перевірки відповідності технічному завданню та аналізу результатів роботи системи.

Приймання інформаційної системи здійснюється після успішного завершення тестування та підтвердження відповідності вимогам технічного завдання.

10. Вимоги до документації

У процесі розробки повинна бути підготовлена така документація:

- технічне завдання;
- керівництво користувача;
- опис архітектури системи;
- опис структури бази даних;
- інструкція з розгортання системи.

ДОДАТОК Б

КЕРІВНИЦТВО КОРИСТУВАЧА

1. Загальні відомості

Дане керівництво користувача призначене для ознайомлення з принципами роботи інформаційної системи управління фондами друкованих і цифрових видань у бібліотечно-репозитарній системі закладу вищої освіти.

Інструкція орієнтована на кінцевих користувачів системи та містить опис основних функцій, інтерфейсу й порядку виконання типових операцій.

2. Призначення інформаційної систем

Інформаційна система призначена для:

- ведення електронного каталогу друкованих і цифрових ресурсів;
- зберігання та обробки бібліографічних метаданих;
- забезпечення доступу користувачів до цифрових документів;
- уніфікації бібліографічних записів стандартів MARC21 та Dublin Core;
- автоматизації процесів пошуку та управління бібліотечними фондами.

3. Категорії користувачів та права доступу

У системі передбачено такі категорії користувачів:

3.1 Адміністратор

Адміністратор має повний доступ до функціоналу системи та виконує такі дії:

- управління користувачами та ролями;
- налаштування системи;
- контроль цілісності даних;
- перегляд журналу подій.

3.2 Бібліотекар

Бібліотекар виконує операції з бібліотечними ресурсами:

- додавання та редагування бібліографічних записів;
- завантаження цифрових файлів;
- перевірка коректності метаданих;
- формування електронного каталогу.

3.3 Користувач

Користувач системи має доступ до:

- перегляду електронного каталогу;
- пошуку бібліотечних ресурсів;
- перегляду описів та цифрових документів.

4. Вхід до системи

Для початку роботи з інформаційною системою користувач повинен:

1. Відкрити веб-браузер.
2. Перейти на головну сторінку системи.
3. Ввести облікові дані (логін і пароль).
4. Натиснути кнопку «Увійти».

Після успішної аутентифікації користувач отримує доступ до функціоналу відповідно до своєї ролі.

5. Головна сторінка системи

- Головна сторінка містить:
- навігаційне меню;
- поле пошуку;
- список доступних бібліотечних ресурсів;
- інформаційні повідомлення системи.

Навігація здійснюється за допомогою меню, розташованого у верхній частині інтерфейсу.

6. Пошук бібліотечних ресурсів

Інформаційна система підтримує атрибутивний та повнотекстовий пошук.

Для виконання пошуку необхідно:

1. Ввести ключові слова у поле пошуку.
2. За необхідності застосувати фільтри (автор, рік видання, тип ресурсу).
3. Натиснути кнопку «Пошук».

Результати пошуку відображаються у вигляді списку бібліографічних записів.

7. Перегляд бібліографічного запису

Для перегляду детальної інформації про ресурс необхідно вибрати відповідний запис зі списку результатів.

На сторінці запису відображається:

- бібліографічний опис;
- тип ресурсу (друкований або цифровий);
- пов'язані файли;
- додаткова інформація про ресурс.

8. Додавання бібліографічного запису

Функція доступна користувачам з роллю «Бібліотекар» або «Адміністратор».

Порядок додавання запису:

1. Перейти до розділу «Додати ресурс».
2. Обрати тип ресурсу (друкований або цифровий).
3. Заповнити поля бібліографічного опису.
4. Завантажити цифровий файл (за потреби).
5. Зберегти запис.

9. Робота з цифровими файлами

Система дозволяє:

завантажувати цифрові документи;

зберігати файли у централізованому сховищі;

переглядати та завантажувати файли відповідно до прав доступу.

Підтримуються основні формати цифрових документів (PDF, DOCX, EPUB тощо).

10. Вихід із системи

Для завершення роботи користувач повинен скористатися пунктом меню «Вийти».

Після виходу з системи сесія користувача завершується, що забезпечує захист персональних даних.

11. Зауваження з безпеки

Користувачеві забороняється:

- передавати облікові дані третім особам;
- використовувати систему з незахищених пристроїв;
- порушувати правила доступу до інформації.

ДОДАТОК В

ФРАГМЕНТИ ПРОГРАМНОГО КОДУ

1. Загальні відомості

Програмна реалізація виконана з використанням фреймворку Django та мови програмування Python. Для збереження даних використовується реляційна база даних PostgreSQL, що забезпечує ефективне управління бібліографічними та цифровими ресурсами.

2. Модель уніфікованого бібліографічного запису

Нижче наведено приклад моделі уніфікованого бібліографічного запису, що використовується для інтеграції метаданих стандартів MARC21 та Dublin Core.

```
from django.db import models

class UnifiedRecord(models.Model):
    title = models.CharField(max_length=255)
    author = models.CharField(max_length=255, blank=True)
    publication_year = models.IntegerField(null=True, blank=True)
    resource_type = models.CharField(max_length=50)
    description = models.TextField(blank=True)

    def __str__(self):
        return self.title
```

3. Модель цифрового ресурсу

Фрагмент програмного коду, що описує модель цифрового ресурсу та зберігання файлів у системі.

```
class DigitalResource(models.Model):
    record = models.ForeignKey(UnifiedRecord, on_delete=models.CASCADE)
    file = models.FileField(upload_to='digital_resources/')
    format = models.CharField(max_length=50)
    upload_date = models.DateTimeField(auto_now_add=True)
```

Модель забезпечує зв'язок між бібліографічним записом та відповідним цифровим файлом.

4. Модель користувача та ролей доступу

Для реалізації розмежування прав доступу у системі використовується модель ролей.

```
class Role(models.Model):
    name = models.CharField(max_length=50)

class UserProfile(models.Model):
    user = models.OneToOneField(User, on_delete=models.CASCADE)
    role = models.ForeignKey(Role, on_delete=models.SET_NULL, null=True)
```

Такий підхід дозволяє гнучко керувати доступом користувачів до функціоналу системи.

5. API для отримання бібліографічних записів

Нижче наведено приклад REST API для отримання списку бібліографічних записів.

```
from rest_framework.views import APIView
from rest_framework.response import Response
```

```
class RecordListAPIView(APIView):
    def get(self, request):
        records = UnifiedRecord.objects.all()
        data = [{
            'title': record.title,
            'author': record.author,
            'year': record.publication_year
        } for record in records]
        return Response(data)
```

Даний API забезпечує інтеграцію інформаційної системи з зовнішніми сервісами та клієнтськими застосунками.

6. Фрагмент механізму пошуку

Приклад реалізації пошуку бібліографічних записів за ключовими словами.

```
from django.db.models import Q
```

```
class RecordSearchService:
    @staticmethod
    def search(query):
        return UnifiedRecord.objects.filter(
            Q(title__icontains=query) |
            Q(author__icontains=query)
        )
```

Механізм пошуку забезпечує швидкий доступ до бібліотечних ресурсів за основними атрибутами.

7. Приклади SQL-запитів до бази даних

У даному підрозділі наведено приклади SQL-запитів до бази даних PostgreSQL, що використовуються в інформаційній системі управління фондами друкованих і цифрових видань.

7.1 Отримання списку всіх бібліографічних записів

```
SELECT id, title, author, publication_year, resource_type  
FROM unified_record  
ORDER BY title;
```

7.2 Пошук бібліографічних записів за автором або назвою

```
SELECT id, title, author, publication_year  
FROM unified_record  
WHERE title ILIKE '%database%'  
       OR author ILIKE '%smith%'
```

7.3 Отримання цифрових ресурсів для конкретного бібліографічного запису

```
SELECT dr.id, dr.format, dr.upload_date, dr.file  
FROM digital_resource dr  
JOIN unified_record ur ON dr.record_id = ur.id  
WHERE ur.id = 1;
```

7.4 Підрахунок кількості ресурсів за типами

```
SELECT resource_type, COUNT(*) AS total  
FROM unified_record  
GROUP BY resource_type;
```

7.5 Отримання списку користувачів з їх ролями

```
SELECT u.username, r.name AS role_name
```

```
FROM auth_user u  
JOIN user_profile up ON u.id = up.user_id  
JOIN role r ON up.role_id = r.id;
```

7.6 Отримання бібліографічних записів, що мають цифрові файли

```
SELECT DISTINCT ur.id, ur.title  
FROM unified_record ur  
JOIN digital_resource dr ON ur.id = dr.record_id;
```

Наведені SQL-запити демонструють типові операції роботи з базою даних інформаційної системи та підтверджують використання реляційної СУБД PostgreSQL у межах кваліфікаційної роботи.