

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

**Факультет програмних систем та прикладної математики
Кафедра системного програмування і спеціалізованих
комп'ютерних систем**

«До захисту допущено»

Завідувач кафедри

_____ Віталій РОМАНКЕВИЧ

«___» _____ 2026 р.

**Дипломний проєкт
на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системне програмування та спеціалізовані комп'ютерні системи»
спеціальності 123 «Комп'ютерна інженерія»
на тему: «Інформаційна система управління локалізованою медіа-
бібліотекою»**

Виконав:

студент 4 курсу, групи КВ-21

Саюк Вадим Анатолійович _____

Керівник:

ст. викладач каф. СПіСКС, д.ф.

Радченко Костянтин Олександрович _____

Консультант з нормоконтролю:

доц. каф. СПіСКС, к.т.н., доц.

Клятченко Ярослав Михайлович _____

Рецензент:

ст. викладач. каф. обчислювальної техніки ФІОТ, д.ф.

Коренко Дмитро Володимирович _____

Засвідчую, що у цьому дипломному
проєкті ємає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2026 року

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРОГРАМНИХ СИСТЕМ ТА ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Освітньо-професійною програма

**«Системне програмування та спеціалізовані
комп'ютерні системи»**

Спеціальність 123 «Комп'ютерна інженерія»

ЗАТВЕРДЖУЮ

Завідувач кафедри

(підпис) Віталій РОМАНКЕВИЧ

“ ____ ” _____ 2025 р.

ЗАВДАННЯ

на дипломний проєкт студента

Саюка Вадима Анатолійовича

1. Тема проєкту «Інформаційна система управління локалізованою медіа-бібліотекою» керівник проєкту ст. викладач каф. СПіСКС, д.ф. Радченко Костянтин Олександрович, затверджена наказом по університету від «28» травня 2026 р. № 1984-С
2. Термін подання студентом проєкту _____
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Зміст пояснювальної записки.
 - 1) аналіз проблеми ефективного енергоспоживання побутовими приладами;
 - 2) математичне моделювання, аналіз методів прогнозування та вибір технологій;
 - 3) вибір бібліотеки та проектування інформаційної системи;
 - 4) реалізація системи та аналіз отриманих результатів.
5. Перелік графічного матеріалу (обов'язкові схеми, презентація тощо).
 - 1) схема інформаційної системи медіа-бібліотеки;
 - 2) схема функціональних вимог;

- 3) схема нефункціональних вимог;
- 4) алгоритм обробки медіа-файлів.

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Клятченко Я. М., доцент кафедри СПіСКС, к.т.н., доцент		

7. Дата видачі завдання «31» жовтня 2025 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Дослідження предметної області та аналіз наукових джерел	08.11.2025	
2	Формування вимог до інформаційної системи	18.11.2025	
3	Огляд існуючих систем управління медіа-контентом	02.12.2025	
4	Проектування архітектури локалізованої медіа-бібліотеки	20.12.2025	
5	Розробка структури даних та моделі локалізації контенту	05.01.2026	
6	Реалізація функціональних модулів системи	23.02.2026	
7	Створення користувацького веб-інтерфейсу	12.03.2026	
8	Реалізація механізмів пошуку, фільтрації та локалізації	18.03.2026	
9	Перевірка працездатності та тестування системи	09.04.2026	
10	Підготовка матеріалів першого та другого розділів пояснювальної записки	28.04.2026	
11	Підготовка матеріалів третього та четвертого розділів пояснювальної записки	13.05.2026	
12	Розроблення графічних матеріалів та оформлення документації	22.05.2026	
13	Підготовка дипломного проєкту до попереднього захисту	29.05.2026	
14	Огляд матеріалів на кафедрі	07.05.2026	

Студент

(Підпис)

Вадим САЮК

Керівник проєкту

(Підпис)

Костянтин РАДЧЕНКО

АНОТАЦІЯ

Дипломний проєкт включає пояснювальну записку (98 с., 23рис., 4 табл., список використаних джерел з 21 найменувань, 3 додатки).

Об'єкт розробки – інформаційна система управління локалізованою медіа-бібліотекою.

Розроблена система дозволяє здійснювати централізоване управління мультимедійним контентом, виконувати пошук і фільтрацію медіа-ресурсів, підтримувати багатомовність інтерфейсу та контенту, а також забезпечувати зручний доступ до локалізованих медіа-матеріалів.

У ході розробки:

- проведено аналіз предметної області та існуючих систем управління медіа-контентом;
- досліджено підходи до реалізації локалізації та інтернаціоналізації інформаційних систем;
- сформовано функціональні та нефункціональні вимоги до системи;
- розроблено архітектуру та структуру інформаційної системи;
- реалізовано програмний прототип із підтримкою локалізації, пошуку, фільтрації, керування медіа-контентом та ролями користувачів;
- проведено тестування функціональних можливостей системи та оцінено результати її роботи.

Ключові слова:

Медіа-бібліотека, локалізація, інтернаціоналізація, мультимедійний контент, інформаційна система, веб-застосунок, багатомовність.

ANNOTATION

The diploma project includes an explanatory note (94 pages, 23 figures, 4 tables, bibliography containing 21 references, 3 appendices).

The object of development is an information system for managing a localized media library.

The developed system provides centralized management of multimedia content, supports media search and filtering, multilingual user interfaces and content localization, and ensures convenient access to localized media resources.

During the development:

- an analysis of the subject area and existing media content management systems was carried out;
- approaches to localization and internationalization of information systems were investigated;
- functional and non-functional requirements for the system were defined;
- the architecture and structure of the information system were designed;
- a software prototype supporting localization, search, filtering, media content management, and user roles was implemented;
- functional testing of the system was performed and the results were evaluated.

Keywords:

Media library, localization, internationalization, multimedia content, information system, web application, multilingual support.

Поз.	Формат	ПОЗНАЧЕННЯ	НАЙМЕНУВАННЯ	Кількість аркушів	№ прим.	Примітки
	A4	ІАЛЦ.045490.002 ТЗ	Інформаційна система управління локалізованою медіа-бібліотекою.			
			Технічне завдання.			
	A4	ІАЛЦ.045490.003 ТП	Інформаційна система управління локалізованою медіа-бібліотекою.	1		
			Відомість технічного проєкту.			
	A4	ІАЛЦ.045490.004 ПЗ	Інформаційна система управління локалізованою медіа-бібліотекою.	98		
			Пояснювальна записка.			
	A4	ІАЛЦ.045490.005 Д1	Інформаційна система медіа-бібліотеки.	1		
			Схема структурна.			
	A4	ІАЛЦ.045490.006 Д2	Функціональні вимоги.	1		
			Схема структурна.			
			ІАЛЦ.045490.001 ОА			
Змін.	Арк.	№ докум.	Підпис	Дата		
Розробив		Сажок В.А.			Літ.	
Перевірив		Радченко К.О.			Арку	
Консуьт.					Аркушів	
Н. контроль		Клятченко Я.М.			КПП	
Зав. каф.		Романкевич В.О.			ім. Ігоря Сікорського, ФПСМ KB-21	
			Інформаційна система управління локалізованою медіа-бібліотекою Опис альбому			

[illegible]

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ.....	15
2. ПІДСТАВА ДЛЯ РОЗРОБКИ	12
3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ	Помилка! Закладку не визначено.
4. ДЖЕРЕЛА РОЗРОБКИ.....	Помилка! Закладку не визначено.
5. ТЕХНІЧНІ ВИМОГИ.....	Помилка! Закладку не визначено.
5.1 Вимоги до програмного продукту, що розробляється	Помилка! Закладку не визначено.
5.2 Вимоги до апаратного забезпечення	Помилка! Закладку не визначено.
5.3 Вимоги до програмного та апаратного забезпечення користувача	Помилка! Закладку не визначено.
6. ЕТАПИ РОЗРОБКИ	14

					ІАЛЦ.045490.002 ТЗ			
Зм.	Лист	№ докум.	Підп.	Дата				
Розробив	Саюк В.А.				Інформаційна система управління локалізованою медіа-бібліотекою Технічне завдання	Лит.	Лист	Листків
Перев.	Радченко К.О.						1	4
						КПІ ім. Ігоря Сікорського, ФПСМ КВ-21		
Н. контр.	Клятченко Я.М.							
Затвер.	Романкевич В.О.							

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Інформаційна система аналізу ефективності використання енергії в побутових приладах».

Галузь застосування: енергоменеджмент, автоматизовані системи контролю побутового енергоспоживання, розумні будинки, аналітичні сервіси для оцінки енергоефективності.

2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на дипломне проектування на здобуття другого (магістерського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського».

3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою проекту є розробка інформаційної системи управління локалізованою медіа-бібліотекою для зберігання, пошуку, фільтрації та перегляду мультимедійного контенту.

Система призначена для централізованого керування медіа-ресурсами, підтримки багатомовності, локалізації контенту та забезпечення зручного доступу користувачів до інформаційних ресурсів.

4. ДЖЕРЕЛА РОЗРОБКИ

Джерелами розробки є науково-технічна література з проектування інформаційних систем, матеріали з локалізації програмного забезпечення, документація веб-технологій (HTML, CSS, JavaScript), а також електронні ресурси мережі Інтернет.

5. ТЕХНІЧНІ ВИМОГИ

5.1 Вимоги до програмного продукту, що розробляється:

- підтримка роботи з мультимедійним контентом (зображення, відео, аудіо та документи);
- реалізація пошуку та фільтрації медіа-ресурсів за визначеними параметрами;
- підтримка локалізації інтерфейсу українською та англійською мовами;
- підтримка завантаження та відображення медіа-контенту;
- реалізація системи ролей користувачів та розмежування доступу;
- забезпечення зручного та інтуїтивно зрозумілого користувацького інтерфейсу;

5.2 Вимоги до апаратного забезпечення:

- центральний процесор не нижче Intel Core i3 або аналогічний AMD;
- обсяг оперативної пам'яті не менше 4 ГБ;
- вільне місце на жорсткому диску не менше 100 МБ;
- дисплей з роздільною здатністю не менше 1366×768 пікселів;
- наявність доступу до мережі Інтернет для роботи із зовнішніми медіа-ресурсами.

5.3 Вимоги до програмного та апаратного забезпечення користувача:

- операційна система Windows 10/11, Linux або macOS;
- сучасний веб-браузер (Google Chrome, Microsoft Edge, Mozilla Firefox, Opera);
- підтримка JavaScript та HTML5;
- стабільне інтернет підключення;

6. ЕТАПИ РОЗРОБКИ

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Дослідження предметної області та аналіз наукових джерел	08.11.2025	
2	Формування вимог до інформаційної системи	18.11.2025	
3	Огляд існуючих систем управління медіа-контентом	02.12.2025	
4	Проектування архітектури локалізованої медіа-бібліотеки	20.12.2025	
5	Розробка структури даних та моделі локалізації контенту	05.01.2026	
6	Реалізація функціональних модулів системи	23.02.2026	
7	Створення користувацького веб-інтерфейсу	12.03.2026	
8	Реалізація механізмів пошуку, фільтрації та локалізації	18.03.2026	
9	Перевірка працездатності та тестування системи	09.04.2026	
10	Підготовка матеріалів першого та другого розділів пояснювальної записки	28.04.2026	
11	Підготовка матеріалів третього та четвертого розділів пояснювальної записки	13.05.2026	
12	Розроблення графічних матеріалів та оформлення документації	22.05.2026	
13	Підготовка дипломного проєкту до попереднього захисту	29.05.2026	
14	Огляд матеріалів на кафедрі	07.05.2026	

Студент

(Підпис)

Вадим САЮК

Керівник проєкту

(Підпис)

Костянтин РАДЧЕНКО

Поз.	Формат	НАЙМЕНУВАННЯ	НАЙМЕНУВАННЯ	Кількіс ть	№	Примітки	
		ІАЛЦ.045490.004 ПЗ	Інформаційна система управління локалізованою медіа-бібліотекою.	94			
			Пояснювальна записка.				
		ІАЛЦ.045490.005 Д1	Інформаційна система медіа-бібліотеки.	1			
			Схема структурна.				
		ІАЛЦ.045490.006 Д2	Функціональні вимоги.	1			
			Схема структурна.				
		ІАЛЦ.045490.007 Д3	Нефункціональні вимоги.	1			
			Схема структурна.				
		ІАЛЦ.045490.008 Д4	Обробка медіа-файлів.	1			
			Схема алгоритму.				
		Диск CD-ROM	Текст пояснювальної записки у двох форматах.	1			
			Код програми.				
			Презентація дипломної роботи. Відео матеріал.				
			ІАЛЦ.045490.003 ТП				
Зм.	Лист	№ докум.	Підп.	Дата			
Розробив	Саюк В.А.				Лит.	Лист	Листків
Перев.	Радченко К.О.					1	1
Н. контр.	Клятченко Я.М.				КПІ ім. Ігоря Сікорського,		
Затвер.	Романкевич В.О.				ФПСМ KB-21		
					Відомість технічного проєкту		
					Інформаційна система управління локалізованою медіа-бібліотекою		

Пояснювальна записка
до дипломного проєкту
на тему: «Інформаційна система управління локалізованою медіа-
бібліотекою».

Київ – 2026

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ	3
ВСТУП	5
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ. 7	
1.1. Поняття та класифікація медіа-бібліотек	7
1.2. Поняття локалізації та інтернаціоналізації	13
1.3. Огляд існуючих систем управління медіа.....	17
1.4. Аналіз вимог до системи.....	22
1.5. Постановка задачі	26
Висновки до розділу 1	30
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ	32
2.1. Загальна архітектура системи.....	32
2.2. Моделювання системи	35
2.3. Проєктування бази даних.....	44
2.4. Модель локалізації контенту	49
2.5. Проєктування API.....	52
Висновки до розділу 2	54
РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ	56
3.1. Вибір технологій	56
3.2. Реалізація серверної частини.....	59
3.3. Реалізація клієнтської частини	64
3.4. Інтеграція системи	69
3.5. Безпека системи	73
Висновки до розділу 3	76

					ІАЛЦ.045490.004 ПЗ			
Змін.	Арк.	№ докум.	Підп.	Дата				
Розробив	Саюк В.А.				Інформаційна система управління локалізованою медіа-бібліотекою Пояснювальна записка	Лнт.	Лист	Листків
Перев.	Радченко К.О.						1	96
Н. контр.	Клятченко Я. М.					КПІ ім. Ігоря Сікорського, ФПСМ KB-21		
Затвер.	Романкевич В. О.							

РОЗДІЛ 4. ТЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ.....	78
4.1. Методика тестування.....	78
4.2. Тестування функціональності	82
4.3. Оцінка продуктивності.....	87
4.4. Аналіз результатів.....	89
Висновки до розділу 4	93
ВИСНОВКИ.....	95
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	97

Додаток 1. Копії графічного матеріалу

ІАЛЦ.045490.005 Д1 Інформаційна ситема медіа - бібліотеки.

Схема структурна.

ІАЛЦ.045490.006 Д2 Функціональні вимоги. Схема структурна.

ІАЛЦ.045490.007 Д3 Нефункціональні вимоги. Схема структурна.

ІАЛЦ.045490.008 Д4 Обробка медіа - файлів. Схема алгоритму.

Додаток 2. Лістинг

Додаток 3. Презентація

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Медіа-контент	цифрові дані, що містять аудіо-, відео-, графічні або текстові матеріали, призначені для зберігання, обробки та відтворення в інформаційних системах.
Медіа-бібліотека	інформаційна система або сховище, призначене для централізованого управління мультимедійними ресурсами з можливістю їх пошуку, класифікації та доступу.
Метадані	структурована інформація, що описує характеристики медіа-об'єкта (назва, автор, дата створення, ключові слова тощо).
Контент-менеджмент	процес створення, редагування, зберігання та організації цифрового контенту в інформаційній системі.
Цифрові активи (Digital Assets)	будь-які цифрові файли, що мають цінність для організації (зображення, відео, документи, аудіо).
Повнотекстовий пошук	механізм пошуку інформації за вмістом текстових полів та метаданих.
Фільтрація даних	процес відбору підмножини даних із загального набору за визначеними критеріями.
CMS (Content Management System)	Система управління контентом — програмна платформа, яка дозволяє створювати, редагувати, зберігати та публікувати веб-контент без необхідності прямого програмування.
DAM (Digital Asset Management)	Система управління цифровими активами — спеціалізована система для централізованого зберігання, організації та управління мультимедійними файлами.
i18n (Internationalization)	Інтернаціоналізація — процес підготовки програмного забезпечення до підтримки різних мов і регіонів без необхідності змінювати його код.

I10n (Localization)	процес адаптації системи або контенту до конкретної мови, культури та регіональних особливостей користувача (переклад, формати дат, валюти тощо).
API (Application Programming Interface)	Програмний інтерфейс застосунків — набір правил і методів, що дозволяють різним програмам взаємодіяти між собою.
UI (User Interface)	Користувацький інтерфейс — сукупність елементів, через які користувач взаємодіє з програмною системою (кнопки, меню, форми).
UX (User Experience)	Користувацький досвід — загальне враження користувача від взаємодії із системою, включаючи зручність, швидкість і зрозумілість інтерфейсу.
RPS (Requests Per Second)	Кількість запитів за секунду — показник продуктивності системи, що відображає, скільки запитів вона здатна обробити за одиницю часу.
CDN (Content Delivery Network)	Мережа доставки контенту — розподілена інфраструктура серверів, яка забезпечує швидке завантаження медіа шляхом передачі даних із найближчого до користувача вузла.
CRUD	Базові операції з даними (Create, Read, Update, Delete): створення, читання, оновлення та видалення записів у базі даних.
JWT (JSON Web Token)	Стандарт безпечної передачі інформації у вигляді токена, який використовується для автентифікації та авторизації користувачів.
NoSQL (Not Only SQL)	Тип баз даних, що не використовує класичну реляційну модель і дозволяє зберігати дані у гнучких структурах (документи, графи, ключ-значення).

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій та цифрових комунікацій особливого значення набувають системи управління мультимедійним контентом. Зростання обсягів цифрових даних, зокрема аудіо-, відео- та графічних матеріалів, зумовлює необхідність створення ефективних засобів їх зберігання, обробки, класифікації та швидкого доступу. Одним із перспективних напрямів у цій сфері є розробка інформаційних систем управління медіа-бібліотеками, здатних забезпечити не лише централізоване керування контентом, але й його адаптацію до потреб різних категорій користувачів.

Актуальність теми дослідження визначається зростаючими вимогами до персоналізації інформаційного середовища та підтримки багатомовності. У глобалізованому цифровому просторі користувачі очікують отримання контенту рідною мовою з урахуванням регіональних особливостей, культурного контексту та індивідуальних вподобань. Це зумовлює необхідність впровадження механізмів локалізації (l10n) та інтернаціоналізації (i18n) у системах управління медіа-ресурсами. Проте більшість існуючих рішень або не забезпечують достатнього рівня гнучкості в управлінні локалізованим контентом, або мають обмежені можливості інтеграції та масштабування.

Об'єктом дослідження є процеси управління цифровим мультимедійним контентом у рамках інформаційних систем. Предметом дослідження є методи, моделі та програмні засоби реалізації локалізованих медіа-бібліотек, що забезпечують ефективне зберігання, обробку, пошук і відображення контенту відповідно до мовних та регіональних параметрів користувача.

Метою дипломного проєкта є розробка інформаційної системи управління локалізованою медіа-бібліотекою, яка забезпечує ефективну

організацію, обробку та адаптацію мультимедійного контенту з урахуванням мовних і регіональних характеристик користувачів.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- провести аналіз предметної області та існуючих систем управління медіа-контентом;
- дослідити підходи до реалізації локалізації та інтернаціоналізації;
- сформулювати функціональні та нефункціональні вимоги до системи;
- розробити архітектуру інформаційної системи;
- спроектувати модель даних для зберігання медіа-контенту та його локалізованих версій;
- реалізувати програмний прототип системи;
- провести тестування та оцінку ефективності розробленого рішення.

У роботі використано методи системного аналізу, об'єктно-орієнтованого проєктування, UML-моделювання, а також сучасні підходи до розробки веб-застосунків та баз даних. Для оцінки ефективності системи застосовано методи функціонального та навантажувального тестування.

Практичне значення отриманих результатів полягає у можливості використання розробленої інформаційної системи в освітніх платформах, медіа-сервісах, корпоративних інформаційних системах та інших програмних продуктах, де необхідне управління великими обсягами локалізованого контенту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Поняття та класифікація медіа-бібліотек

У сучасному інформаційному суспільстві значна частина даних представлена у вигляді мультимедійного контенту, що включає текстові, аудіо-, відео- та графічні ресурси. Для ефективного управління такими ресурсами використовуються спеціалізовані інформаційні системи – медіа-бібліотеки.

Медіа-бібліотека – це організована сукупність цифрових мультимедійних ресурсів, що зберігаються, обробляються та надаються користувачам за допомогою інформаційної системи. Основною метою медіа-бібліотеки є забезпечення швидкого доступу до контенту, його структуризація, пошук, фільтрація та керування [1].

На відміну від традиційних бібліотек, медіа-бібліотеки оперують не лише текстовими документами, а й різноманітними форматами мультимедіа, що вимагає більш складних підходів до організації даних, включаючи метадані, індексацію та механізми доставки контенту.

Ключовими характеристиками медіа-бібліотек є:

- наявність структурованого сховища мультимедійних даних;
- підтримка метаданих для опису контенту (назва, автор, мова, формат тощо);
- механізми пошуку та навігації;
- можливість масштабування та інтеграції з іншими системами;
- підтримка багатокористувацького доступу.

Медіа-бібліотеки можуть бути класифіковані за низкою ознак, які визначають їх структуру, функціональність та принципи організації даних.

Такий поділ дозволяє краще зрозуміти різноманіття існуючих рішень і особливості їх застосування в різних предметних областях.

Однією з базових ознак класифікації є тип контенту, який зберігається в системі. Залежно від цього виділяють аудіо-бібліотеки, що містять музичні записи, подкасти та інші звукові матеріали; відео-бібліотеки, орієнтовані на фільми, навчальні ролики та потоковий контент; графічні бібліотеки, які працюють із зображеннями, фотографіями та ілюстраціями; а також документні бібліотеки, що включають електронні книги й текстові файли. Окремо виділяють гібридні медіа-бібліотеки, які поєднують кілька типів контенту в межах однієї системи.

Іншим важливим критерієм є спосіб організації доступу до даних. У цьому контексті розрізняють локальні медіа-бібліотеки, які функціонують у межах одного пристрою або локальної мережі, хмарні системи, де дані зберігаються на віддалених серверах із доступом через Інтернет, а також розподілені системи, в яких інформація зберігається на кількох вузлах із механізмами синхронізації.

За функціональним призначенням медіа-бібліотеки поділяються на освітні, що використовуються у навчальному процесі; корпоративні, які застосовуються всередині організацій для внутрішніх потреб; публічні, доступні широкому колу користувачів; та спеціалізовані, які орієнтовані на конкретні предметні області, наприклад медицину або мистецтво [2].

Окрему роль відіграє рівень інтерактивності системи. Пасивні медіа-бібліотеки забезпечують лише перегляд і доступ до контенту, тоді як інтерактивні системи дозволяють користувачам взаємодіяти з матеріалами, залишати коментарі, оцінювати або редагувати інформацію.

Важливим сучасним аспектом є підтримка локалізації. Одномовні системи працюють лише з однією мовою, багатомовні забезпечують можливість перемикання між кількома мовами інтерфейсу та контенту, а

локалізовані медіа-бібліотеки додатково враховують культурні та регіональні особливості користувачів, адаптуючи інформацію під конкретний контекст.

Ще однією класифікаційною ознакою є архітектура системи. Монолітні рішення передбачають реалізацію всіх компонентів в одному програмному середовищі, мікросервісні системи розподіляють функціональність між незалежними сервісами, а гібридні архітектури поєднують властивості обох підходів, забезпечуючи баланс між простотою та масштабованістю.

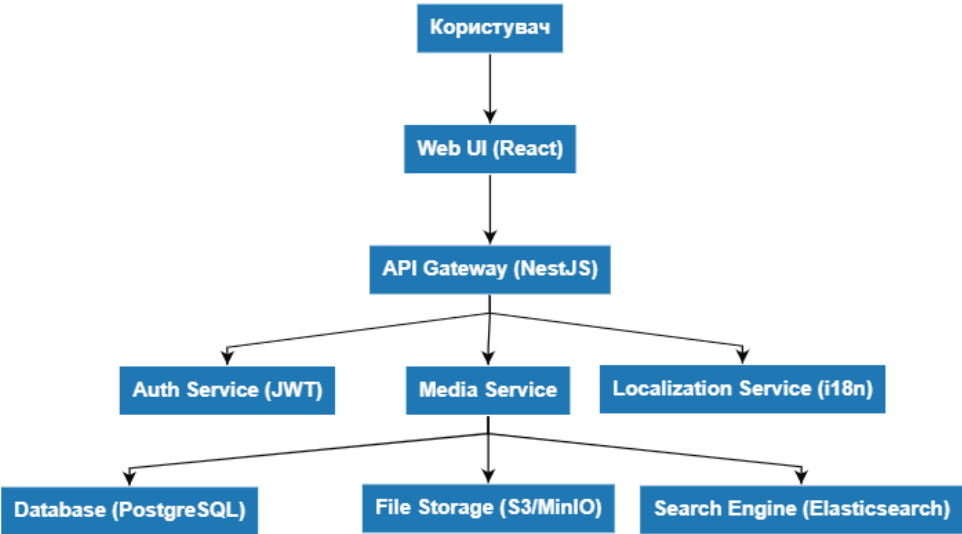


Рис. 1.1. Інформаційна система медіа-бібліотеки.

Медіа-бібліотеки є складними інформаційними системами, що забезпечують управління мультимедійним контентом у різних формах і середовищах (рис. 1.1). Їх класифікація дозволяє визначити ключові характеристики та особливості реалізації залежно від вимог користувачів і сфери застосування. Особливе значення в сучасних умовах набувають локалізовані медіа-бібліотеки, які забезпечують адаптацію контенту до мовних і культурних потреб користувачів, що є важливим аспектом у глобалізованому інформаційному просторі.

Медіа-контент є основним інформаційним наповненням медіа-бібліотек і охоплює різні типи цифрових ресурсів, що відрізняються за форматом, структурою, способом відтворення та обробки. Кожен тип контенту має свої особливості зберігання, індексації та використання в інформаційних системах.

Аудіо-контент включає звукову інформацію, представлену у цифровому вигляді. До нього належать музичні композиції, подкасти, аудіокниги, записи лекцій, інтерв'ю та інші звукові матеріали. Основними характеристиками аудіо є: формат зберігання (MP3, WAV, AAC тощо); бітрейт та частота дискретизації; тривалість запису; метадані (виконавець, альбом, жанр, мова).

Особливістю аудіо-контенту є його відносно невеликий обсяг порівняно з відео та можливість потокового відтворення (streaming), що широко використовується у сучасних системах [2].

Відео-контент являє собою поєднання візуальної та аудіоінформації, синхронізованих у часі. Це найбільш ресурсоємний тип медіа, який включає фільми, серіали, навчальні відео, вебінари та стримінгові трансляції. Основні характеристики: формат (MP4, AVI, MKV тощо); роздільна здатність (SD, HD, Full HD, 4K); частота кадрів; кодеки стиснення; супровідні дані (субтитри, аудіодоріжки, локалізації).

Відео-контент вимагає значних обчислювальних ресурсів для обробки, зберігання та передачі, а також підтримки адаптивного стримінгу.

Графічний контент включає статичні візуальні об'єкти: фотографії, ілюстрації, інфографіку, скановані документи, дизайнерські матеріали. Основні характеристики: формат (JPEG, PNG, GIF, SVG); роздільна здатність і розмір; глибина кольору; метадані (автор, дата створення, геолокація) [3].

Зображення широко використовуються у веб-застосунках, маркетингу, освіті та потребують оптимізації для швидкого завантаження та відображення.

Документний контент представлений текстовими або змішаними файлами, що можуть містити текст, таблиці, зображення та інші елементи. До

нього належать електронні книги, наукові статті, звіти, презентації, інструкції. Основні формати: PDF, DOCX, TXT, HTML, EPUB; структуровані документи (XML, JSON). Особливістю документного контенту є його орієнтація на семантичний зміст, що дозволяє реалізовувати повнотекстовий пошук, аналіз та автоматичну обробку інформації [4].

Різноманітність видів медіа-контенту визначає складність побудови інформаційних систем медіа-бібліотек. Кожен тип контенту потребує специфічних підходів до зберігання, обробки та відображення. Ефективна система повинна забезпечувати універсальні механізми роботи з усіма типами медіа, враховуючи їх технічні характеристики та особливості використання, а також підтримувати можливості локалізації та адаптації контенту для різних категорій користувачів.

Однією з ключових характеристик інформаційних систем управління медіа-контентом є спосіб організації зберігання та обробки даних. За цією ознакою медіа-бібліотеки поділяються на централізовані та розподілені. Кожен із підходів має свої особливості, переваги та обмеження, що впливають на ефективність функціонування системи [4].

Централізовані медіа-бібліотеки передбачають зберігання всього контенту та метаданих у єдиному сховищі або на одному сервері (чи кластері, що логічно виступає як єдине ціле). Управління ресурсами, обробка запитів та контроль доступу здійснюються з одного центру.

Розподілені медіа-бібліотеки базуються на зберіганні та обробці даних на декількох незалежних вузлах (серверах), які можуть бути географічно віддаленими. Дані можуть бути розподілені, репліковані або кешовані між вузлами для підвищення доступності та продуктивності.

Вибір між централізованою та розподіленою архітектурою залежить від масштабу системи, вимог до продуктивності та доступності (табл. 1.1). Для локалізованих медіа-бібліотек, орієнтованих на широку аудиторію, доцільним

є використання розподіленого підходу, оскільки він дозволяє ефективно обслуговувати користувачів з різних регіонів і забезпечувати адаптивну доставку контенту відповідно до їхніх мовних і географічних особливостей.

Таблиця 1.1.

Порівняльний аналіз архітектури бібліотек

Критерій	Централізовані системи	Розподілені системи
Масштабованість	Обмежена	Висока
Відмовостійкість	Низька	Висока
Складність	Низька	Висока
Узгодженість даних	Висока	Потребує контролю
Продуктивність	Середня	Висока

Цифрові медіа-сховища є ключовим компонентом інформаційних систем управління медіа-бібліотеками, оскільки забезпечують зберігання, обробку та доступ до великих обсягів мультимедійного контенту. На відміну від традиційних файлових систем, вони орієнтовані на роботу з різнорідними даними, що мають значний обсяг, складну структуру та потребують ефективної організації [5].

Для ефективного керування медіа-контентом використовуються метадані, тобто структуровані описи інформаційних ресурсів, які характеризують їхній основний зміст. До таких метаданих відносять назву, автора та дату створення матеріалу, а також формат і технічні параметри файлу. Сучасні цифрові медіа-сховища використовують складні механізми індексації, які забезпечують швидкий доступ до інформації. Зокрема, реалізується повнотекстовий пошук для текстових документів, пошук за метаданими, а також контент-орієнтований пошук, який може враховувати

вміст зображень або аудіофайлів. Такий підхід дозволяє значно підвищити ефективність взаємодії користувача із системою та скоротити час отримання необхідної інформації.

Оскільки медіа-контент часто має обмеження доступу, пов'язані з авторськими правами або конфіденційністю, система повинна забезпечувати надійний контроль доступу. Для цього використовуються механізми автентифікації користувачів, авторизації прав доступу, шифрування даних, а також аудит дій користувачів, що дозволяє відстежувати всі операції в системі.

У контексті локалізованих медіа-бібліотек важливою є можливість зберігання різних мовних версій одного ресурсу (наприклад, субтитрів, аудіодоріжок, перекладів документів) та швидкий вибір відповідної версії залежно від користувача.

1.2. Поняття локалізації та інтернаціоналізації

У сучасних інформаційних системах, орієнтованих на глобальну аудиторію, важливе значення мають процеси адаптації програмного забезпечення та контенту до різних мовних і культурних середовищ. Така адаптація забезпечується за допомогою двох взаємопов'язаних, але принципово різних, підходів – інтернаціоналізації та локалізації (табл. 1.2).

Інтернаціоналізація є основою для локалізації. Якщо система не була спроєктована з урахуванням і18n, процес і10n стає складним, дорогим і технічно обмеженим. У правильно побудованій системі локалізація виконується шляхом додавання або зміни ресурсів без втручання у програмний код [6].

Локалізація контенту в інформаційних системах, зокрема в медіа-бібліотеках, може реалізовуватися за різними підходами залежно від архітектури системи, типу контенту та вимог до масштабованості. Вибір

підходу впливає на ефективність управління перекладами, продуктивність системи та зручність супроводу.

Таблиця 1.2.

Ключові відмінності між i18n та l10n

Критерій	Інтернаціоналізація (i18n)	Локалізація (l10n)
Етап розробки	На етапі проєктування та програмування	Після реалізації базової системи
Основна мета	Забезпечити універсальність системи	Адаптувати систему до конкретного регіону
Рівень реалізації	Архітектура, код, структура даних	Контент, інтерфейс, медіа
Зміни у коді	Потребує змін у коді	Не повинна потребувати змін у коді
Об'єкт роботи	Технічна база системи	Мовні та культурні ресурси
Приклад	Використання Unicode, ресурсних файлів	Переклад текстів, додавання субтитрів

Існує кілька основних підходів до реалізації локалізації в інформаційних системах, кожен з яких має свої особливості, переваги та обмеження. Вибір конкретного підходу залежить від архітектури системи, обсягу контенту та вимог до гнучкості й продуктивності.

Файлово-ресурсний підхід ґрунтується на зберіганні перекладів у зовнішніх файлах ресурсів, таких як JSON, XML або PO-файли. Такі файли підключаються залежно від обраної користувачем мови. У цьому випадку текстові елементи інтерфейсу виносяться з програмного коду, а доступ до них здійснюється через унікальні ключі. Перевагами цього підходу є простота

реалізації, зручність редагування перекладів без зміни коду, а також мінімальний вплив на продуктивність системи. Водночас його недоліками є складність управління великою кількістю перекладів і обмежені можливості для роботи з динамічним контентом.

Базоданий (database-driven) підхід передбачає зберігання всіх мовних версій контенту безпосередньо в базі даних. Кожен запис може мати декілька варіантів перекладу, що дозволяє централізовано керувати контентом і редагувати його через адміністративний інтерфейс. Цей підхід є гнучким і добре масштабується, особливо у великих системах. Однак він створює додаткове навантаження на базу даних і потребує оптимізації запитів, зокрема за допомогою кешування [7].

Гібридний підхід поєднує переваги файлової та базоданих моделей. У ньому інтерфейсні тексти зазвичай зберігаються у файлах ресурсів, тоді як основний контент і його переклади — у базі даних. Такий розподіл дозволяє оптимізувати продуктивність і водночас зберегти гнучкість управління даними. Недоліком є більш складна архітектура системи та необхідність узгодження двох різних механізмів локалізації.

Ще одним варіантом є підхід із використанням окремих версій контенту, коли для кожної мови створюється повністю незалежна копія даних. Це дозволяє максимально адаптувати контент під конкретну аудиторію, включно з культурними особливостями. Однак такий підхід призводить до дублювання даних, ускладнює синхронізацію змін і збільшує обсяг сховища.

Динамічна локалізація базується на автоматичному визначенні мови користувача, наприклад через налаштування браузера, геолокацію або профіль користувача. Система самостійно підбирає відповідну мовну версію інтерфейсу та контенту, що підвищує зручність використання та рівень персоналізації. Разом із тим реалізація такого підходу є складнішою і вимагає додаткових алгоритмів аналізу користувацьких даних [7].

Окреме місце займає локалізація мультимедійного контенту, яка є особливо важливою для медіа-бібліотек. Вона передбачає використання субтитрів різними мовами, додавання альтернативних аудіодоріжок, а також переклад метаданих, таких як назви, описи та ключові слова. Це дозволяє одному медіа-ресурсу існувати одночасно у кількох мовних варіантах і забезпечує його доступність для широкої аудиторії.

Підтримка багатомовності є одним із ключових аспектів сучасних інформаційних систем, зокрема медіа-бібліотек, які орієнтовані на глобальну аудиторію. Вона забезпечує можливість взаємодії користувачів із системою різними мовами без втрати функціональності та якості представлення контенту. У контексті медіа-бібліотек багатомовність охоплює не лише інтерфейс користувача, але й сам медіа-контент: назви, описи, субтитри, аудіодоріжки та метадані.

Багатомовність (multilingualism) – це здатність інформаційної системи одночасно підтримувати декілька мов інтерфейсу та контенту. На відміну від простого перекладу, багатомовність передбачає системну організацію даних, що дозволяє динамічно перемикає мову без зміни логіки роботи програми.

Інтерфейс інформаційної системи повинен бути повністю адаптивним до мови, обраної користувачем. Це означає, що всі елементи користувацького інтерфейсу, включаючи меню, кнопки та системні повідомлення, мають відображатися у відповідному мовному варіанті. Крім того, система повинна підтримувати динамічне перемикає мови без необхідності перезавантаження сторінки, а також коректно обробляти різні алфавіти та напрямки письма, зокрема LTR (зліва направо) та RTL (справа наліво) [8].

Разом із тим реалізація багатомовності пов'язана з низкою проблем. Серед них можна виділити складність синхронізації перекладів між різними мовними версіями, збільшення обсягу даних у системі, а також труднощі підтримки консистентності контенту. Додатково виникають проблеми,

пов'язані з різною довжиною текстів у різних мовах, що впливає на відображення інтерфейсу, а також необхідність постійного оновлення перекладів.

Незважаючи на ці складнощі, багатомовні системи мають суттєві переваги. Вони дозволяють розширити аудиторію користувачів, підвищити доступність контенту та покращити загальний користувацький досвід. Крім того, підтримка багатомовності забезпечує відповідність міжнародним стандартам і підвищує конкурентоспроможність програмного продукту на глобальному ринку.

1.3. Огляд існуючих систем управління медіа

Сучасні інформаційні системи управління медіа-контентом представлені широким спектром рішень, які можна умовно поділити на системи керування контентом (CMS), системи управління цифровими активами (DAM), медіа-репозиторії та спеціалізовані хмарні платформи. Їх основною метою є ефективне зберігання, організація, пошук і доставка мультимедійного контенту [9].

Системи керування контентом (CMS, Content Management System) мають низку характерних особливостей, які визначають їх широке застосування у веб-розробці. Вони забезпечують управління як текстовими, так і мультимедійними даними, а також надають зручні візуальні редактори типу WYSIWYG, що дозволяють редагувати контент без необхідності роботи з кодом. Важливим елементом є підтримка шаблонізації сторінок, яка спрощує формування єдиного стилю інтерфейсу. Крім того, CMS зазвичай реалізують систему ролей користувачів для розмежування прав доступу та інтегруються з базами даних і зовнішніми API. До найпоширеніших прикладів таких систем належать WordPress, Joomla та Drupal [9].

Водночас CMS мають певні обмеження. Вони, як правило, орієнтовані на базове управління медіа-файлами і не забезпечують повноцінної роботи з великими медіа-архівами. Також у таких системах часто відсутні розвинені механізми версіонування файлів та складної роботи з розширеними метаданими, що обмежує їх використання у великих медіа-орієнтованих проектах.

Окрему категорію становлять системи DAM (Digital Asset Management), які спеціалізуються на централізованому управлінні цифровими активами. Такі платформи забезпечують єдине сховище для різних типів медіа-контенту, включаючи зображення, відео, аудіо та документи. Вони також підтримують розширене управління метаданими і тегами, контроль версій, а також гнучке налаштування прав доступу. Завдяки цьому користувачі отримують можливість швидкого пошуку та фільтрації великих обсягів контенту [10].

Серед прикладів DAM-рішень можна виділити Bynder, Adobe Experience Manager Assets, Widen та Canto. Основними перевагами таких систем є висока масштабованість, централізоване управління цифровими ресурсами, підтримка бізнес-процесів (workflow) та ефективне повторне використання контенту. Проте їх впровадження часто є складним і дорогим, особливо у випадку корпоративних рішень, а також вимагає інтеграції з іншими інформаційними системами, такими як CMS або PIM.

Ще однією сучасною категорією є хмарні медіа-платформи, які забезпечують зберігання та доставку контенту через хмарну інфраструктуру. До таких сервісів належать Cloudinary, Amazon S3 разом із CloudFront та Google Cloud Storage. Їхньою особливістю є автоматична оптимізація медіа-файлів, зокрема стиснення та зміна розміру зображень, використання CDN для швидкої доставки контенту, API-орієнтована архітектура, а також підтримка потокового передавання даних [10].

Окремо слід виділити відкриті системи управління медіа-активами, які застосовуються у гнучких або локальних проєктах. Прикладами таких рішень є ResourceSpace, Pimcore та EnterMedia. Вони відрізняються можливістю глибокого налаштування, безкоштовною базовою функціональністю, підтримкою плагінів і розширень, а також придатністю для використання в наукових і корпоративних середовищах, де важлива адаптивність системи.

Таблиця 1.3.

Порівняльний аналіз систем

Критерій	CMS	DAM	Хмарні платформи
Основне призначення	Веб-контент	Медіа-активи	Доставка і обробка
Робота з медіа	Базова	Розширена	Оптимізована
Метадані	Обмежені	Розширені	Середні
Масштабованість	Середня	Висока	Дуже висока
Складність	Низька	Висока	Середня

Аналіз існуючих систем (табл. 1.3) показує, що жоден клас рішень не є універсальним. CMS ефективні для управління веб-контентом, DAM – для роботи з великими медіа-бібліотеками, а хмарні платформи – для масштабованої доставки контенту.

У контексті локалізованої медіа-бібліотеки найбільш перспективним є поєднання DAM-підходу з хмарною архітектурою та механізмами локалізації, що дозволяє забезпечити гнучке управління контентом, його багатомовність та високу доступність для користувачів [11].

WordPress є найпоширенішою системою керування контентом у світі, що пояснюється її простотою використання та великою кількістю готових рішень у вигляді плагінів і шаблонів. Вона забезпечує швидке розгортання

вебсайтів, має активну спільноту розробників і добре підходить для реалізації малих і середніх проєктів. Водночас її основними недоліками є обмежена масштабованість у складних системах, можливі проблеми з продуктивністю при надмірній кількості плагінів, а також необхідність додаткових рішень для організації складних медіа-сховищ [11].

Drupal, на відміну від WordPress, орієнтований на більш складні та масштабні проєкти. Він характеризується високим рівнем гнучкості та розширюваності, а також потужною підтримкою структурованого контенту. Система має вбудовані механізми безпеки та добре підходить для корпоративних інформаційних рішень. Недоліками Drupal є складність освоєння, потреба у високій технічній кваліфікації розробників та триваліший процес розробки порівняно з простішими CMS [11].

Joomla займає проміжну позицію між WordPress і Drupal, поєднуючи відносну простоту використання з достатнім рівнем функціональності. Однією з її сильних сторін є вбудована підтримка багатомовності та гнучка система управління правами доступу. Водночас Joomla має меншу екосистему розширень і поступається WordPress за популярністю, що може обмежувати доступність готових рішень [11].

Adobe Experience Manager (AEM) належить до корпоративних CMS рівня enterprise і входить до складу екосистеми Adobe Experience Cloud. Його ключовою перевагою є поєднання функціональності CMS і DAM в одній платформі, що дозволяє ефективно управляти як контентом, так і цифровими активами. Система підтримує персоналізацію контенту, багатомовність, складні робочі процеси та автоматизацію, а також забезпечує високу масштабованість для великих організацій. Основними недоліками є висока вартість впровадження, складність навчання та необхідність залучення досвідченої технічної команди [12].

Окрему категорію становлять DAM-системи, які призначені для централізованого управління цифровими медіа-активами, такими як зображення, відео, аудіо та документи. Одним із найпотужніших рішень є Adobe Experience Manager Assets, що є частиною АЕМ. Він забезпечує централізоване зберігання медіа, автоматичну генерацію різних версій і форматів файлів, використання штучного інтелекту для тегування контенту, а також глибоку інтеграцію з CMS і маркетинговими інструментами.

Таблиця 1.4.

Загальний порівняльний аналіз CMS та DAM

Критерій	CMS (WordPress/Drupal)	DAM (AEM Assets/Bynder)
Основне призначення	Публікація контенту	Управління медіа-активами
Тип контенту	Текст + базові медіа	Великі медіафайли
Метадані	Обмежені	Розширені
Масштабованість	Середня	Висока
Локалізація	Часткова	Глибока підтримка
Складність	Низька–середня	Висока

Іншим популярним рішенням є Bynder, хмарна DAM-платформа, орієнтована насамперед на маркетингові команди. Вона відзначається зручним інтерфейсом, швидким пошуком медіа-контенту, підтримкою бренд-менеджменту та інтеграцією з CMS і CRM-системами. Серед недоліків можна виділити високу вартість і залежність від хмарної інфраструктури [12].

Платформа Widen (нині Asquia DAM) орієнтована на великі організації та характеризується високим рівнем масштабованості. Вона пропонує потужні механізми роботи з метаданими, контроль версій медіа-файлів та аналітику використання контенту. Водночас система є складною у впровадженні та

переважно орієнтована на enterprise-сегмент, що обмежує її використання у невеликих проєктах [12].

Аналіз популярних CMS та DAM-систем (табл. 1.4) показує, що CMS-рішення (WordPress, Drupal, Joomla) ефективні для керування веб-контентом, але мають обмеження у роботі з великими медіа-архівами. Натомість DAM-системи забезпечують глибоке управління цифровими активами, але є складнішими та дорожчими у впровадженні.

Для інформаційних систем управління локалізованими медіа-бібліотеками найбільш ефективним є комбінований підхід, який поєднує CMS для роботи з інтерфейсом і публікацією контенту та DAM для управління медіа-ресурсами, метаданими і багатомовністю.

1.4. Аналіз вимог до системи

Аналіз вимог є одним із ключових етапів проєктування інформаційної системи управління локалізованою медіа-бібліотекою. Його метою є визначення функціональних можливостей, обмежень та характеристик системи, які забезпечать її ефективне функціонування відповідно до потреб користувачів і предметної області.

Передусім система повинна забезпечувати управління медіа-контентом. Це включає можливість завантаження різних типів мультимедійних файлів, таких як аудіо, відео, зображення та документи, а також їх подальше редагування й оновлення. Окремо передбачається функція видалення застарілого контенту та логічного групування ресурсів за категоріями і тегами, що спрощує організацію даних у системі.

Важливим компонентом є робота з метаданими. Система повинна забезпечувати збереження розширеної інформації про кожен медіа-об'єкт, включаючи назву, опис, автора, дату створення та ключові слова. Також

необхідна можливість редагування метаданих, їх індексації для прискорення пошуку та прив'язки до конкретних мовних версій контенту.

Функціональність пошуку та фільтрації є ключовою для роботи з великими обсягами даних (рис. 1.2). Система повинна підтримувати повнотекстовий пошук, фільтрацію за категоріями, тегами та мовою, а також сортування результатів за різними критеріями, такими як дата створення, популярність або релевантність.

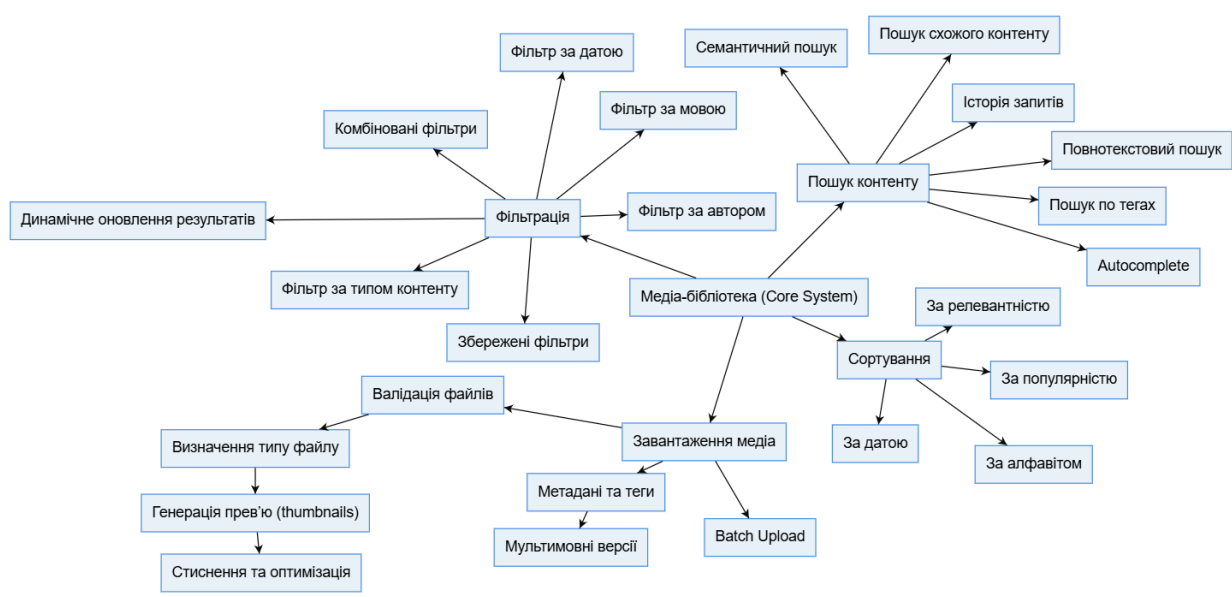


Рис. 1.2. Схема функціональних вимог інформаційної системи

Окрему групу складають вимоги до підтримки локалізації. Система повинна забезпечувати зберігання багатомовних версій контенту, автоматичне визначення мови користувача, а також можливість ручного перемикавання мовного інтерфейсу. Додатково необхідна підтримка мультимедійних локалізованих елементів, таких як субтитри та аудіодоріжки різними мовами.

Адміністративний модуль системи повинен забезпечувати повноцінне управління медіа-бібліотекою, модерацію контенту, моніторинг активності користувачів та налаштування мов і параметрів локалізації.

Нефункціональні вимоги визначають якісні характеристики системи, що впливають на її продуктивність та зручність використання (рис. 1.3). Зокрема, система повинна забезпечувати високу продуктивність, включаючи швидку обробку запитів користувачів, оптимізований пошук у великих масивах даних та використання кешування для популярного контенту.

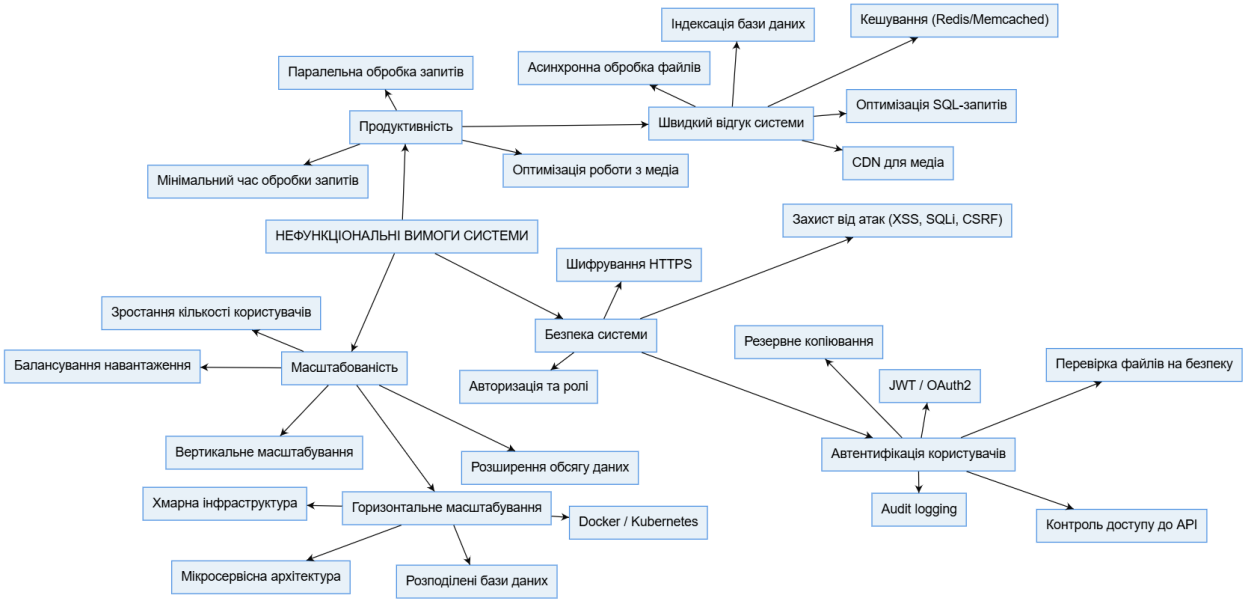


Рис. 1.3. Схема нефункціональних вимог інформаційної системи

Важливою є масштабованість системи, яка дозволяє збільшувати обсяг даних і кількість користувачів без втрати продуктивності. Для цього передбачається підтримка розподіленої архітектури та можливість горизонтального масштабування.

Надійність системи забезпечується механізмами резервного копіювання, збереженням даних при збоях та можливістю відновлення після помилок. Це гарантує стабільну роботу навіть у випадку технічних збоїв.

До вимог безпеки належать захист користувацьких даних, автентифікація та авторизація користувачів, контроль доступу до медіа-ресурсів, а також захист від несанкціонованого доступу до системи.

Зручність використання передбачає інтуїтивний інтерфейс, підтримку декількох мов та адаптивний дизайн, який коректно працює на різних пристроях. Сумісність системи забезпечується підтримкою сучасних браузерів, можливістю інтеграції із зовнішніми API та використанням стандартних форматів обміну даними, таких як JSON та XML [13].

Окрему групу становлять вимоги до багатомовності системи. Вони передбачають підтримку щонайменше двох мов інтерфейсу, наприклад української та англійської, можливість додавання нових мов без внесення змін у кодову базу, а також структуроване зберігання перекладів. Додатково система повинна підтримувати різні мовні версії одного медіа-ресурсу та автоматично визначати мову користувача для покращення зручності роботи.

Система повинна забезпечувати підтримку щонайменше двох мов інтерфейсу, наприклад української та англійської, а також надавати можливість розширення мовної палітри без внесення змін у програмний код. Важливою вимогою є динамічне перемикання мови без перезавантаження системи, а також збереження обраної мовної конфігурації в профілі користувача або поточній сесії.

Окрему увагу слід приділити мультимедійному контенту, який повинен підтримувати кілька мовних варіантів одночасно. Це передбачає наявність локалізованих назв, описів і тегів для кожного медіа-ресурсу, підтримку багатомовних субтитрів для відео, можливість додавання альтернативних аудіодоріжок, а також механізм логічного зв'язування різних мовних версій одного й того ж контенту.

Метадані, як ключовий елемент медіа-бібліотеки, також повинні бути повністю адаптовані до багатомовного середовища. Це включає підтримку полів у різних мовних варіантах (назва, опис, ключові слова), індексацію метаданих з урахуванням мови, можливість виконання пошуку різними

мовами та забезпечення узгодженості між відповідними мовними версіями записів [14].

Система повинна мати механізми автоматичного визначення мови користувача, що може здійснюватися на основі налаштувань браузера, IP-геолокації або профілю користувача. При цьому має бути передбачена можливість ручного вибору або перевизначення мови інтерфейсу.

Вимоги до локалізації є ключовим елементом у проектуванні інформаційної системи управління медіа-бібліотекою. Вони забезпечують адаптацію системи до різних мовних і культурних середовищ, підвищують доступність контенту та покращують користувацький досвід. Реалізація повноцінної локалізації разом із підтримкою багатомовності є необхідною умовою створення сучасної глобально орієнтованої медіа-платформи.

1.5. Постановка задачі

Постановка задачі визначає формалізований опис проблеми, яку необхідно вирішити в межах дипломного проєкта, а також окреслює основні цілі, обмеження та очікувані результати розробки інформаційної системи управління локалізованою медіа-бібліотекою.

У сучасних умовах стрімкого зростання обсягів мультимедійного контенту виникає потреба в ефективних інформаційних системах, які забезпечують його зберігання, структурування, пошук та доставку користувачам. Існуючі CMS та DAM-рішення частково вирішують ці задачі, однак мають обмеження у сфері повноцінної локалізації, гнучкого управління метаданими та адаптації контенту до різних мовних і культурних середовищ.

Таким чином, існує необхідність у розробці спеціалізованої інформаційної системи, яка поєднує функціональність медіа-менеджменту з розширеними можливостями локалізації та масштабування.

Формалізація задачі розробки інформаційної системи управління локалізованою медіа-бібліотекою полягає у переході від описової постановки до чітко визначеної структурованої моделі, яка може бути реалізована засобами програмної інженерії. Вона включає визначення вхідних і вихідних даних, функціональних перетворень, обмежень та цільових критеріїв ефективності (рис. 1.4).

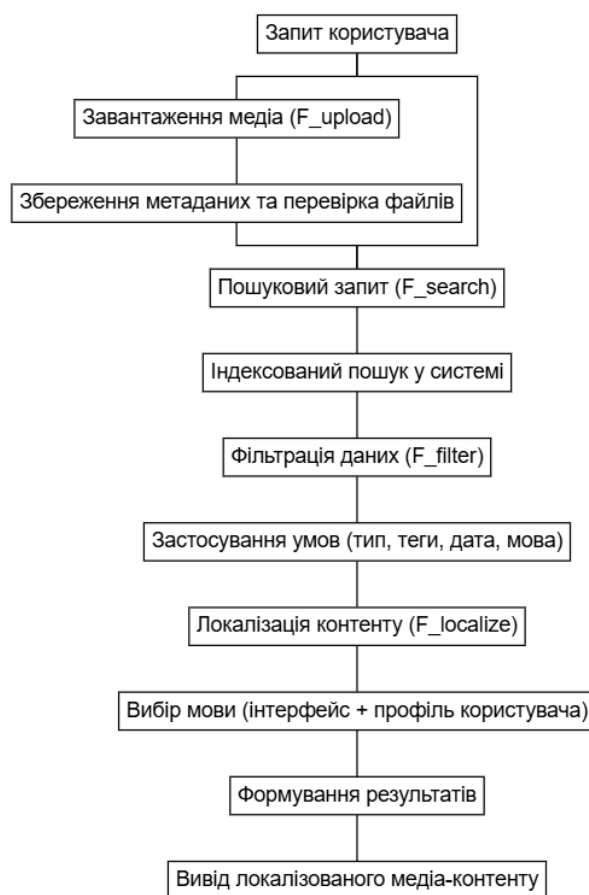


Рис. 1.4. UML-діаграма активностей інформаційної системи

Нехай існує множина медіа-ресурсів [15]:

$$M = \{m_1, m_2, \dots, m_n\}, \quad (1)$$

де кожен медіа-об'єкт m_i характеризується набором атрибутів:

$$m_1 = (type, metadata, content, languageVersions), \quad (2)$$

де: *type* – тип медіа (audio, video, image, document), *metadata* – набір описових параметрів, *content* – фізичний файл або посилання на нього, *languageVersions* – множина локалізованих представлень.

Нехай задано множини користувачів та мов: $U = \{u_1, u_2, \dots, u_k\}$ — множина користувачів; $L = \{l_1, l_2, \dots, l_p\}$ — множина підтримуваних мов.

Кожному користувачу ставиться у відповідність мовний профіль, який визначає його мовні налаштування:

$$lang(u_i) \in L. \quad (3)$$

Інформаційна система управління локалізованою медіа-бібліотекою описується набором основних функцій.

Функція завантаження контенту визначається як відображення вхідного набору медіа-об'єктів у внутрішнє сховище системи:

$$F_{upload}: M_{in} \rightarrow M, \quad (4)$$

де система приймає нові медіа-ресурси та додає їх до загального сховища M .

Пошук контенту визначається як функція, що повертає підмножину медіа-об'єктів відповідно до запиту користувача та його мовних налаштувань:

$$F_{search}(q, l) \rightarrow M', \quad (5)$$

де q — пошуковий запит; $l \in L$ — мова користувача; $M' \subseteq M$ — множина знайдених медіа-об'єктів.

Фільтрація контенту дозволяє звужувати множину даних за заданими умовами [15]:

$$F_{filter}(M, C) \rightarrow M', \quad (6)$$

де C — набір критеріїв (тип, дата, мова, теги тощо); $M' \subseteq M$ — відфільтрована множина результатів.

Функція локалізації контенту формує мовно-залежне представлення медіа-об'єкта:

$$F_{localize}(m_i, l) \rightarrow m_i^{(l)}, \quad (7)$$

де $m_i^{(l)}$ — локалізована версія контенту для мови l .

Функціонування системи повинно відповідати таким умовам: кожен медіа-об'єкт має унікальний ідентифікатор; для кожної мовної версії має існувати fallback-мова у разі відсутності перекладу; час виконання пошуку після індексації: $T_{search} \leq O(\log n)$; система повинна зберігати продуктивність при зростанні обсягу даних: $n \rightarrow \infty \Rightarrow performance \approx const$.

Ефективність системи можна представити як задачу оптимізації [16]:

$$Optimize(S) = \max(Q_{search}, Q_{avail}, Q_{local}) - \min(T_{response}, C_{resources}), \quad (8)$$

де Q_{search} — якість пошуку; Q_{avail} — доступність системи; Q_{local} — якість локалізації; $T_{response}$ — час відповіді; $C_{resources}$ — споживання ресурсів.

Інформаційна система визначається як кортеж:

$$IS = (M, U, L, F_{upload}, F_{search}, F_{filter}, F_{localize}), \quad (9)$$

де кожен елемент відповідає окремому компоненту системи.

Ефективність системи оцінюється через набір якісних та кількісних показників.

Основним критерієм є мінімізація часу відповіді як продуктивність системи:

$$T_{response} \rightarrow \min. \quad (10)$$

Додатково оцінюються швидкість пошуку, час завантаження та кількість запитів за секунду (RPS). Система вважається ефективною, якщо стабільно працює при зростанні навантаження [17].

Якість пошуку визначається через точність і повноту результатів:

$$Q_{search} = f(Precision, Recall) \rightarrow \max. \quad (11)$$

Оцінюється повнота перекладів, узгодженість мовних версій і швидкість перемикавання мов через якість локалізації:

$$Q_{local} \rightarrow \max. \quad (12)$$

Система повинна зберігати стабільність при збільшенні навантаження для своєї масштабованості:

$$S(n) \approx \text{const}, n \rightarrow \infty. \quad (13)$$

Надійність системи характеризується часом безвідмовної роботи та швидкістю відновлення [18]:

$$R \rightarrow \text{max}. \quad (14)$$

Безпека оцінюється рівнем захисту даних і контролем доступу:

$$\text{Sec} \rightarrow \text{max}. \quad (15)$$

Критерієм є мінімізація витрат обчислювальних ресурсів в якості ефективності використання ресурсів:

$$C_{\text{resources}} \rightarrow \text{min}. \quad (16)$$

Критерії ефективності дозволяють комплексно оцінити інформаційну систему управління локалізованою медіа-бібліотекою з точки зору продуктивності, якості пошуку, рівня локалізації, масштабованості, безпеки та ресурсної ефективності. Оптимальна система повинна забезпечувати баланс між високою продуктивністю, якістю обробки контенту та мінімальними витратами ресурсів, що є ключовою умовою її практичного застосування.

Висновки до розділу 1

У першому розділі дипломного проєкта було здійснено ґрунтовний аналіз предметної області, пов'язаної з розробкою інформаційної системи управління локалізованою медіа-бібліотекою. Розглянуті теоретичні та прикладні аспекти дозволили сформуванати цілісне уявлення про сучасний стан технологій управління мультимедійним контентом, а також визначити основні проблеми та обмеження існуючих рішень.

Окремо розглянуто підходи до організації медіа-бібліотек, зокрема централізовані та розподілені архітектури. Встановлено, що централізовані системи є простішими у реалізації, але мають обмеження щодо

масштабованості та відмовостійкості, тоді як розподілені системи забезпечують високу продуктивність і надійність, але потребують складнішої інфраструктури та механізмів синхронізації даних. Також проаналізовано особливості цифрових медіа-сховищ, які є основою сучасних мультимедійних платформ і забезпечують роботу з великими обсягами різномірних даних.

У розділі значну увагу приділено питанням інтернаціоналізації та локалізації. Показано, що інтернаціоналізація є архітектурною основою, тоді як локалізація – прикладним рівнем адаптації системи. Окремо підкреслено важливість підтримки багатомовності як невід’ємної складової сучасних медіа-платформ, що орієнтовані на глобальну аудиторію.

У рамках аналізу існуючих рішень було розглянуто сучасні CMS та DAM-системи, зокрема їх функціональні можливості, переваги та обмеження. Встановлено, що CMS-системи (наприклад, WordPress, Joomla, Drupal) забезпечують зручне управління веб-контентом, однак мають обмежені можливості роботи з великими медіа-архівами та складною структурою метаданих. У свою чергу, DAM-системи (Adobe Experience Manager, Bynder, Widen) надають розширені можливості управління цифровими активами, але характеризуються високою складністю впровадження та значними витратами ресурсів.

Також у розділі було виконано формалізацію задачі розробки системи, що дозволило перейти від описового рівня до математичної моделі. Визначено множини користувачів, медіа-об’єктів та мов, а також функції обробки даних (завантаження, пошук, фільтрація, локалізація). Це створює основу для подальшого алгоритмічного та архітектурного проектування системи.

Отримані теоретичні положення та сформульовані вимоги створюють необхідну основу для подальшого проектування архітектури системи та її програмної реалізації у наступних розділах дипломного проєкта.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Загальна архітектура системи

Загальна архітектура інформаційної системи управління локалізованою медіа-бібліотекою визначає структурну організацію компонентів системи, їх взаємодію та принципи обміну даними. Архітектура побудована з урахуванням вимог масштабованості, продуктивності, безпеки та підтримки багатомовності, що є критичними для системи роботи з мультимедійним контентом.

Для реалізації системи доцільно використовувати багаторівневу (layered) або клієнт-серверну архітектуру з елементами модульності. Такий підхід дозволяє [19]:

- розділити бізнес-логіку, інтерфейс та роботу з даними;
- спростити масштабування системи;
- підвищити гнучкість та підтримуваність;
- забезпечити незалежність компонентів.

У розширеному вигляді архітектура може бути представлена як трирівнева система (рис. 2.1):

- Presentation Layer (рівень представлення);
- Application/Business Layer (рівень бізнес-логіки);
- Data Layer (рівень даних).

Рівень представлення (Frontend) забезпечує безпосередню взаємодію користувача з інформаційною системою та відповідає за відображення контенту і реалізацію користувацького інтерфейсу. Саме на цьому рівні здійснюється перегляд медіа-ресурсів, виконання пошуку та фільтрації даних, а також управління основними діями користувача, такими як завантаження файлів, навігація сторінками чи перегляд локалізованого контенту. Важливою особливістю frontend-рівня є підтримка багатомовного інтерфейсу

користувача, що дозволяє динамічно змінювати мову системи відповідно до налаштувань користувача. Крім того, інтерфейс повинен бути адаптивним для коректної роботи як на персональних комп'ютерах, так і на мобільних пристроях. Для підвищення швидкодії та зручності взаємодії можуть застосовуватися технології SPA або AJAX, які забезпечують динамічне оновлення даних без повного перезавантаження сторінки. Frontend також інтегрується з API серверної частини для отримання та відправлення даних.

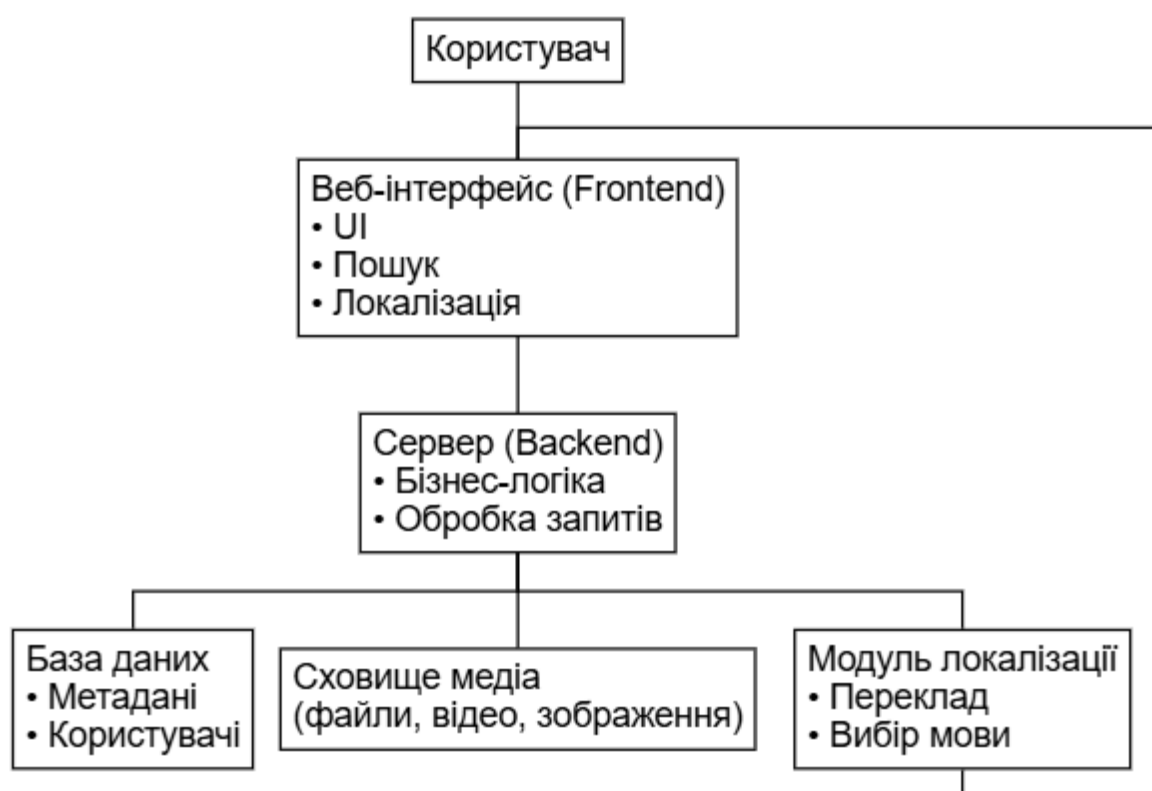


Рис. 2.1. Архітектурна схема інформаційної системи

Рівень бізнес-логіки (Backend) є центральним компонентом архітектури системи, оскільки саме він реалізує основні функціональні можливості платформи. На цьому рівні функціонують модулі управління медіа-контентом, пошуку та індексації, фільтрації і сортування даних, локалізації контенту, а

також управління користувачами та їх ролями. Backend обробляє запити, що надходять від клієнтської частини, застосовує бізнес-правила роботи з контентом, здійснює перевірку прав доступу та формує відповіді для клієнта. Важливим компонентом є API-шар, який забезпечує взаємодію між frontend-та backend-рівнями, а також дозволяє інтегрувати систему із зовнішніми сервісами.

Рівень даних (Data Layer) відповідає за зберігання, обробку та організацію інформаційних ресурсів системи. До його складу входить реляційна база даних, у якій зберігаються метадані, облікові записи користувачів та права доступу, а також файлове або об'єктне сховище для медіа-контенту. Для реалізації швидкого пошуку можуть використовуватися спеціалізовані системи індексації, наприклад Elasticsearch, а для оптимізації доступу до даних — кеш-системи. Основними функціями цього рівня є зберігання структурованих і неструктурованих даних, забезпечення швидкого доступу до медіа-файлів, підтримка версіонування контенту та індексація інформації для прискорення пошукових операцій [20].

Важливим елементом архітектури є модуль локалізації, який відповідає за підтримку багатомовності системи. Його функціональність включає зберігання різних мовних версій контенту, управління мовними ресурсами та автоматичне визначення мови користувача. У разі відсутності необхідного перекладу система повинна використовувати fallback-механізм, тобто резервну мовну версію. Модуль локалізації інтегрується як з інтерфейсом користувача, так і з API, забезпечуючи коректне відображення локалізованого контенту в усіх компонентах системи.

Взаємодія між компонентами системи відбувається у вигляді послідовного потоку обробки даних. Спочатку користувач взаємодіє з клієнтським інтерфейсом, після чого сформований запит передається через API до серверної частини. Backend-рівень виконує обробку запиту відповідно

до бізнес-логіки та звертається до рівня даних для отримання необхідної інформації. Після цього результати обробки повертаються користувачу у локалізованому вигляді відповідно до вибраної або автоматично визначеної мовної конфігурації системи.

Систему можна формалізувати як [21]:

$$IS = \{UI, BL, DL, L, API\}, \quad (17)$$

де **UI** – користувацький інтерфейс; **BL** – бізнес-логіка; **DL** – рівень даних; **L** – модуль локалізації; **API** – інтерфейс взаємодії між рівнями.

Загальна архітектура інформаційної системи управління локалізованою медіа-бібліотекою побудована на багаторівневому підході, що забезпечує чітке розділення відповідальності між компонентами системи. Така структура дозволяє досягти високої продуктивності, масштабованості та гнучкості, а також забезпечує ефективну підтримку мультимедійного контенту та багатомовності. Обрана архітектура створює основу для подальшого детального проєктування модулів системи та їх програмної реалізації.

2.2. Моделювання системи

Моделювання інформаційної системи управління локалізованою медіа-бібліотекою здійснюється із застосуванням UML (Unified Modeling Language), що дозволяє формалізувати структуру системи, поведінку її компонентів та взаємодію між ними. Використання UML забезпечує наочність проєктування, спрощує аналіз вимог та є основою для подальшої програмної реалізації.

Діаграма варіантів використання (Use Case Diagram) відображає функціональні можливості інформаційної системи з точки зору взаємодії користувачів із системою. Вона дозволяє формалізувати вимоги до системи, визначити ролі користувачів (акторів) та описати основні сценарії їх роботи з функціоналом медіа-бібліотеки. У контексті розроблюваної системи управління локалізованою медіа-бібліотекою дана діаграма демонструє, які

саме операції доступні різним типам користувачів та як вони взаємодіють із функціональними модулями системи [20].

У межах системи виділяються три ключові ролі користувачів, кожна з яких має власний рівень доступу та набір дозволених операцій (рис. 2.2).

Користувач (User) є базовим актором системи. Його основна роль полягає у перегляді та пошуку медіа-контенту. Такий користувач взаємодіє переважно з інтерфейсною частиною системи, отримуючи доступ до каталогу медіа, можливостей пошуку, фільтрації та перегляду локалізованого контенту відповідно до обраної мови.

Редактор (Editor) виконує розширені функції, пов'язані з управлінням контентом. Окрім перегляду медіа, редактор має можливість завантажувати нові медіа-файли, редагувати існуючі записи, оновлювати метадані та керувати структурою контенту. Його діяльність безпосередньо впливає на наповнення медіа-бібліотеки.

Адміністратор (Admin) є найвищим рівнем користувача в системі. Він відповідає за загальне управління системою, включаючи адміністрування користувачів, розподіл ролей і прав доступу, а також контроль безпеки та стабільності роботи системи. Адміністратор має повний доступ до всіх функцій платформи.

Функціональність системи реалізується через набір основних варіантів використання, які відображають ключові сценарії взаємодії користувача із системою.

Одним із базових сценаріїв є завантаження медіа-контенту, яке дозволяє редакторам та адміністраторам додавати нові файли до системи. Під час цього процесу відбувається не лише збереження файлів, але й автоматичне формування метаданих та перевірка відповідності форматів.

Важливим варіантом використання є перегляд медіа-бібліотеки, який забезпечує користувачам доступ до доступного контенту. У цьому режимі

система відображає медіа-ресурси з урахуванням прав доступу та мовних налаштувань користувача.

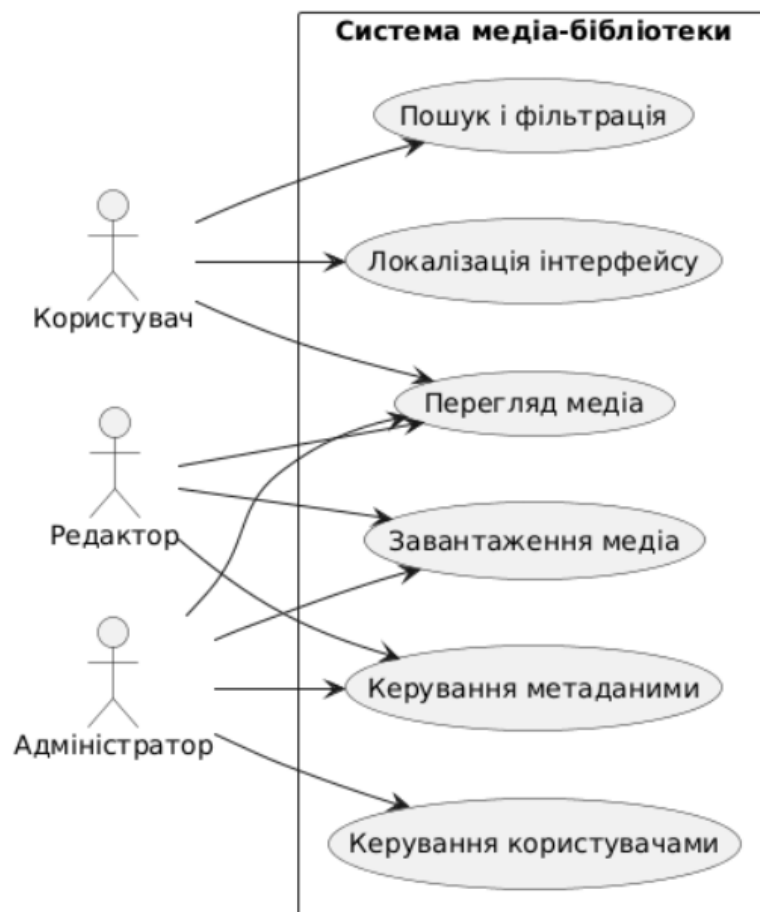


Рис. 2.2. UML-діаграма варіантів використання інформаційної системи

Функція пошуку та фільтрації контенту дозволяє ефективно працювати з великими обсягами даних. Користувач може здійснювати пошук за ключовими словами, метаданими, тегами, категоріями або мовною версією контенту, а також уточнювати результати за допомогою фільтрів.

Окремим важливим варіантом використання є керування метаданими, який передбачає можливість редагування описів, ключових слів, категорій та інших атрибутів медіа-ресурсів. Це забезпечує структурованість і зручність подальшого пошуку.

Функціонал локалізації інтерфейсу дозволяє користувачам змінювати мову системи та отримувати контент у зручному мовному представленні. Система підтримує динамічне перемикавання мов та автоматичне визначення мови користувача.

Окремим варіантом використання є управління користувачами, який доступний лише адміністратору. Він включає створення нових облікових записів, призначення ролей, редагування прав доступу та контроль активності користувачів у системі.

В нашому випадку, діаграма варіантів використання відображає повну структуру взаємодії між користувачами та системою, визначає межі функціональності та дозволяє чітко формалізувати вимоги до розроблюваної інформаційної системи управління локалізованою медіа-бібліотекою.

Опишемо динамічну взаємодію між об'єктами системи в часі під час виконання конкретного сценарію за допомогою діаграми послідовності (рис. 2.3). На відміну від структурних діаграм, вона показує не статичні зв'язки, а порядок викликів між компонентами системи, що дозволяє детально проаналізувати логіку виконання операцій.

У контексті інформаційної системи управління локалізованою медіа-бібліотекою розглядається сценарій пошуку медіа-контенту, який є одним із ключових функціональних процесів. Даний сценарій демонструє повний цикл обробки запиту — від введення користувачем до відображення локалізованих результатів у інтерфейсі.

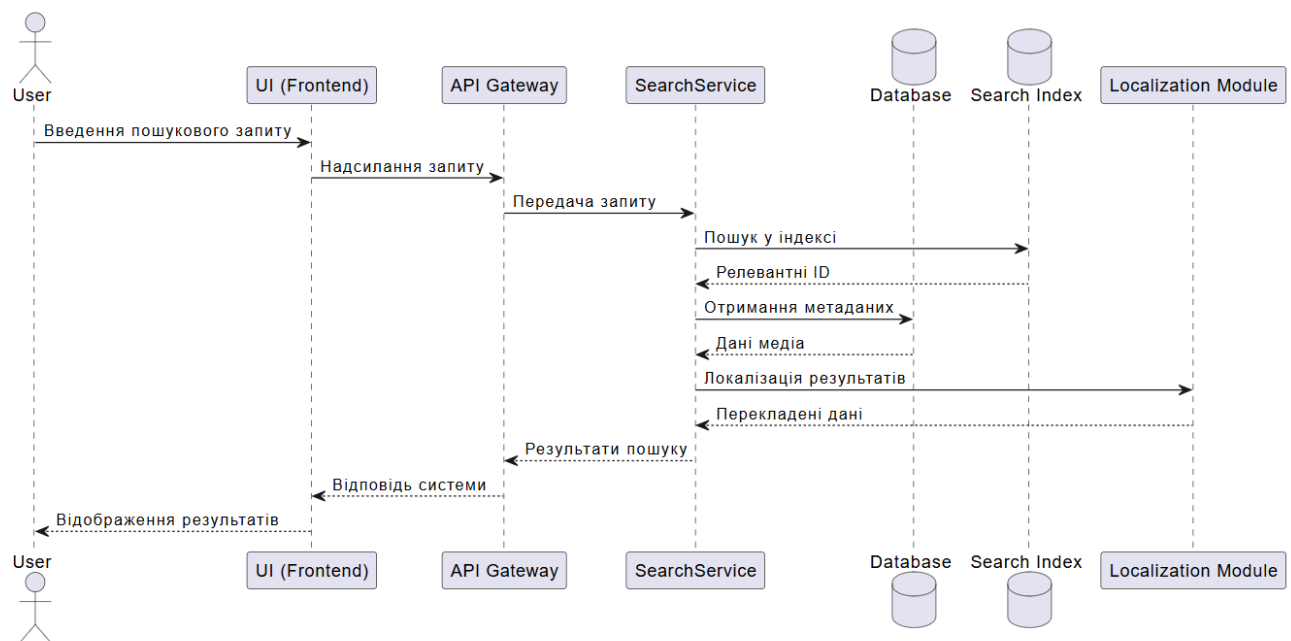


Рис. 2.3. Діаграма послідовності розробленої інформаційної системи

Процес починається з того, що користувач взаємодіє з інтерфейсом користувача (UI) та вводить пошуковий запит. Після цього запит передається через API до серверної частини системи. API виступає проміжним шаром, який забезпечує передачу даних між клієнтом і серверною логікою.

Далі запит надходить до модуля SearchService, який відповідає за обробку пошукових операцій. На цьому етапі система аналізує запит, визначає критерії пошуку та формує відповідні запити до бази даних і пошукового індексу.

Після цього SearchService виконує звернення до бази даних та індексної системи, де зберігаються метадані та проіндексований контент. Це дозволяє швидко отримати релевантні результати навіть при великому обсязі даних.

Отримані результати повертаються назад до сервісного шару, де додатково обробляються, зокрема з урахуванням мовних налаштувань користувача. На цьому етапі застосовується механізм локалізації, який забезпечує відображення контенту у відповідній мовній версії.

Фінальний результат передається назад через API до клієнтської частини, де UI відображає користувачу вже оброблені та локалізовані результати пошуку.

Відобразимо послідовність виконання бізнес-процесу роботи з медіа-контентом у межах інформаційної системи за допомогою діаграми діяльності (рис. 2.4). Вона демонструє логіку переходів між станами системи та дозволяє описати повний цикл обробки медіа-ресурсу — від моменту його завантаження до доступу кінцевого користувача.



Рис. 2.4. Діаграма діяльності розробленої інформаційної системи

У даній моделі основний акцент зроблено на процесі додавання нового медіа-контенту до системи та його подальшій обробці. Процес починається із

завантаження файлу користувачем або редактором через інтерфейс системи. На цьому етапі відбувається передача файлу до серверної частини для подальшої обробки.

Наступним кроком є перевірка формату файлу, під час якої система аналізує тип медіа-ресурсу, його розмір та відповідність дозволеним стандартам. У разі невідповідності файл може бути відхилений або конвертований у підтримуваний формат.

Після успішної перевірки система переходить до етапу генерації метаданих. На цьому кроці автоматично формуються описові дані про медіа-об'єкт, включаючи назву, тип, дату завантаження, ключові слова та інші атрибути, необхідні для подальшого пошуку та фільтрації.

Далі виконується збереження медіа-файлу у сховище даних. Файл розміщується у файловому або об'єктному сховищі, тоді як його метадані записуються у базу даних. Це забезпечує розділення зберігання контенту та його описової інформації.

Після цього система виконує індексацію контенту для пошуку. На цьому етапі медіа-об'єкт додається до пошукового індексу, що дозволяє забезпечити швидкий доступ до нього через механізми пошуку та фільтрації.

Завершальним етапом є надання доступу користувачу. Після завершення всіх попередніх кроків медіа-контент стає доступним для перегляду, пошуку та використання відповідно до прав доступу та мовних налаштувань користувача.

Відобразимо логічну та модульну структуру інформаційної системи управління локалізованою медіа-бібліотекою за допомогою діаграми компонентів (рис. 2.5). Вона показує основні програмні компоненти системи, їх призначення та взаємозв'язки, що дозволяє зрозуміти архітектурну організацію програмного забезпечення на рівні модулів.

У центрі архітектури знаходиться компонент Backend API, який виконує роль посередника між клієнтською частиною та внутрішніми сервісами системи. Він приймає запити від Frontend, маршрутизує їх до відповідних модулів і формує уніфіковані відповіді для користувача.

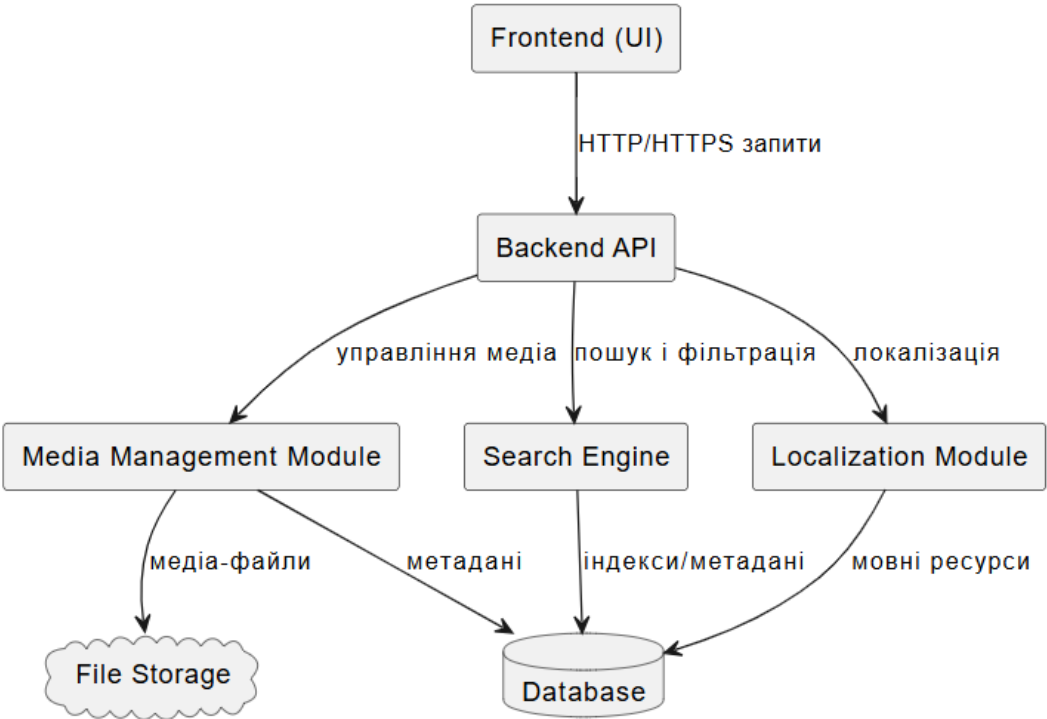


Рис. 2.5. Діаграма компонентів розробленої інформаційної системи

Клієнтський компонент Frontend (UI) відповідає за взаємодію користувача із системою. Він забезпечує відображення медіа-контенту, інтерфейси пошуку та фільтрації, а також підтримує багатомовний користувацький інтерфейс. Frontend взаємодіє виключно через API, не маючи прямого доступу до бази даних або сховищ.

Компонент Media Management Module відповідає за управління медіа-ресурсами. Він реалізує функціональність завантаження, редагування, видалення та організації медіа-контенту, а також взаємодіє з файловим сховищем для збереження фізичних файлів.

Search Engine є окремим компонентом, який забезпечує індексацію та пошук медіа-контенту. Він працює з метаданими та індексами, що дозволяє швидко виконувати повнотекстовий пошук, фільтрацію та ранжування результатів.

Localization Module відповідає за підтримку багатомовності системи. Він керує мовними ресурсами, забезпечує вибір мови користувача, обробляє переклади інтерфейсу та контенту, а також реалізує fallback-механізми для відсутніх перекладів.

Database використовується для зберігання структурованих даних, таких як метадані медіа-об'єктів, інформація про користувачів, ролі та права доступу. Вона є основним джерелом даних для бізнес-логіки системи.

File Storage призначений для зберігання великих медіа-файлів (відео, аудіо, зображення, документи). Він відокремлений від бази даних для підвищення продуктивності та масштабованості системи.

Таким чином, компонентна архітектура системи забезпечує чітке розділення відповідальностей, високу модульність, масштабованість і можливість незалежного розвитку окремих частин системи.

Архітектура системи дозволяє представити систему як сукупність взаємопов'язаних компонентів:

- користувачі взаємодіють через UI;
- бізнес-логіка обробляє запити;
- дані зберігаються в сховищах;
- локалізація забезпечує багатомовність;
- пошуковий модуль забезпечує швидкий доступ до контенту.

Моделювання системи дозволило формалізувати структуру та поведінку інформаційної системи управління локалізованою медіа-бібліотекою. Отримані діаграми забезпечують чітке розуміння взаємодії компонентів, ролей користувачів та основних процесів у системі. Це створює основу для

подальшого проєктування програмної архітектури та реалізації функціональних модулів системи.

2.3. Проєктування бази даних

Проєктування бази даних інформаційної системи управління локалізованою медіа-бібліотекою є одним із ключових етапів, оскільки саме база даних забезпечує зберігання, організацію та швидкий доступ до медіа-контенту, метаданих і інформації про користувачів. Основною метою проєктування є створення структурованої, нормалізованої та масштабованої моделі даних, яка підтримує багатомовність і ефективний пошук.

При розробці бази даних враховуються такі принципи:

- нормалізація даних для уникнення дублювання;
- цілісність даних (referential integrity);
- масштабованість структури;
- підтримка багатомовності (локалізація даних);
- оптимізація для пошуку та фільтрації;
- розділення медіа-файлів і метаданих.

Основною сутністю системи є клас **User**, який представляє користувача платформи. Користувач взаємодіє із системою залежно від своєї ролі, що визначається через зв'язок із класом **Role**. Ролі дозволяють реалізувати розмежування доступу, наприклад між звичайним користувачем, редактором та адміністратором.

Центральним елементом предметної області (рис. 2.6) є клас **MediaItem**, який описує окремий медіа-об'єкт. Це може бути зображення, відео, аудіо або документ. Кожен медіа-об'єкт містить набір метаданих, які представлені класом **Metadata**. Такий зв'язок реалізує агрегацію, оскільки метадані є невід'ємною, але логічно окремою частиною медіа-ресурсу.

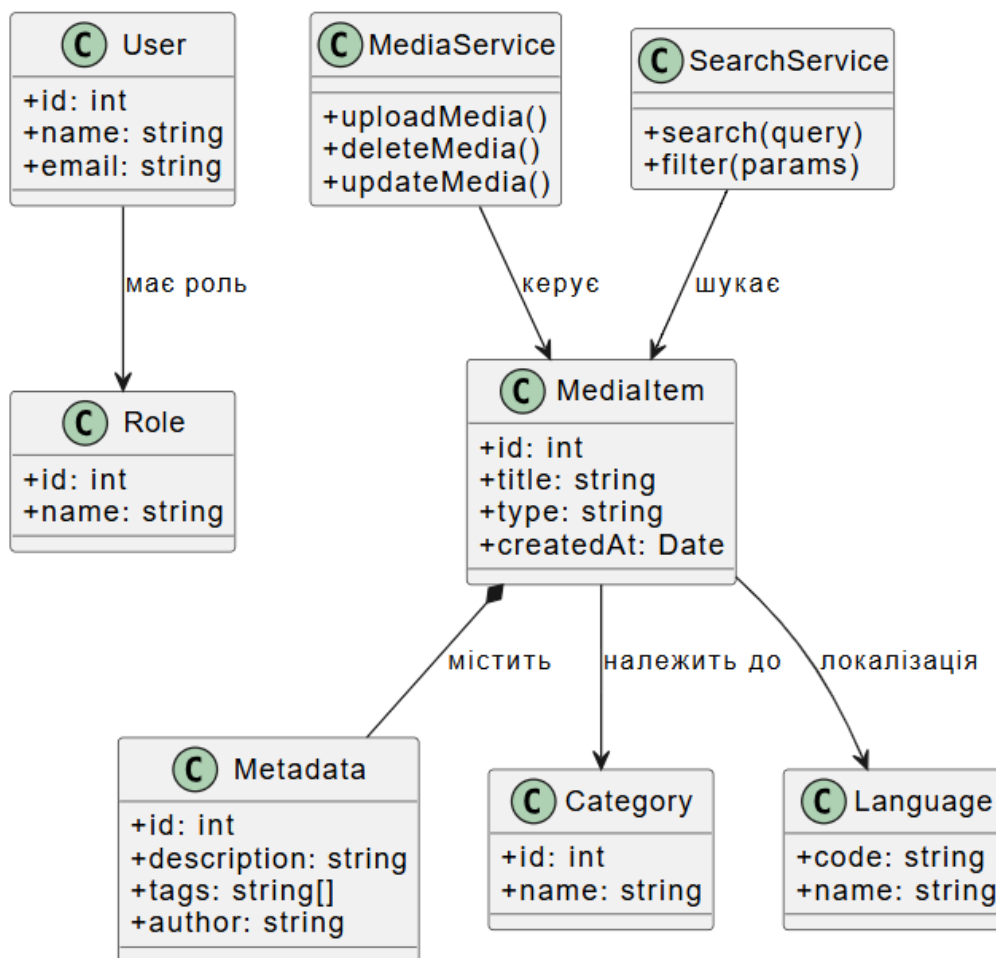


Рис. 2.6. Діаграма класів розробленої інформаційної системи

Для підтримки багатомовності система використовує клас **Language**, який визначає доступні мовні версії контенту. Один медіа-об'єкт може мати кілька мовних варіантів, що забезпечує реалізацію механізму локалізації та адаптації контенту під користувача.

Логіка роботи з медіа інкапсульована у класі **MediaService**, який відповідає за завантаження, обробку, оновлення та видалення медіа-об'єктів. Він взаємодіє безпосередньо з класом **MediaItem** та координує роботу з метаданими і мовними версіями.

Пошукова функціональність реалізується через окремий клас **SearchService**, який забезпечує індексацію, виконання пошукових запитів,

фільтрацію та ранжування результатів. Це дозволяє відокремити бізнес-логіку пошуку від логіки управління контентом.

Таким чином, запропонована модель відображає чітке розділення відповідальностей між сутностями системи, що підвищує її масштабованість, підтримуваність та розширюваність.

Логічна модель бази даних включає основні сутності системи та зв'язки між ними.

У реляційній базі даних інформаційної системи управління медіа-бібліотекою реалізовано набір основних таблиць, які забезпечують зберігання користувачів, медіа-контенту, метаданих, а також підтримку багатомовності та класифікації даних.

Таблиця **User** використовується для зберігання інформації про користувачів системи. Вона містить унікальний ідентифікатор користувача, ім'я, електронну пошту, хеш пароля та зовнішній ключ, який визначає роль користувача в системі.

Таблиця **Role** визначає ролі користувачів та рівні доступу. Вона включає ідентифікатор ролі, її назву (наприклад, адміністратор, редактор або звичайний користувач) та набір дозволів, які визначають права доступу до функціоналу системи.

Таблиця **MediaItem** відповідає за зберігання інформації про медіа-об'єкти. Вона містить шлях до файлу, тип медіа (зображення, відео, аудіо або документ), дату завантаження, розмір файлу та посилання на користувача, який його додав.

Таблиця **Metadata** використовується для опису медіа-контенту. Вона містить назву, опис та посилання на мовну версію, що дозволяє реалізувати багатомовність на рівні метаданих.

Таблиця **Category** призначена для класифікації медіа-ресурсів за категоріями, забезпечуючи структуровану організацію контенту.

Таблиця **Tag** використовується для гнучкого маркування медіа-об'єктів ключовими словами, що покращує пошук та фільтрацію.

Зв'язувальна таблиця **MediaTag** реалізує зв'язок типу many-to-many між медіа-об'єктами та тегами, дозволяючи одному файлу мати декілька тегів і навпаки.

Таблиця **Language** зберігає інформацію про доступні мови системи, включаючи мовний код (наприклад, ua, en) та повну назву мови.

Таблиця **MediaLocalization** відповідає за зберігання локалізованих версій медіа-контенту. Вона містить перекладені назви, описи та шлях до субтитрів для відео, а також зв'язок із відповідною мовою.

Модель даних базується на чітко визначених зв'язках між сутностями, що забезпечує цілісність і узгодженість інформації.

Кожен користувач системи пов'язаний із певною роллю через зв'язок типу many-to-one, що дозволяє реалізувати рольову модель доступу. Також кожен медіа-об'єкт належить конкретному користувачу, який його створив або завантажив.

Медіа-об'єкти пов'язані з метаданими у відношенні один до багатьох, що дозволяє зберігати кілька описових записів для різних мовних версій або контекстів використання.

Зв'язок між медіа-об'єктами та тегами реалізовано як many-to-many, що забезпечує гнучку систему класифікації контенту.

Кожен медіа-об'єкт може належати певній категорії, що реалізується зв'язком many-to-one, тоді як локалізація забезпечується через зв'язок один до багатьох між медіа-об'єктами та мовними версіями.

Таблиця локалізації додатково пов'язана з таблицею Language, яка визначає конкретну мову для кожної версії контенту.

Багатомовність у системі реалізується через окрему структуру зберігання локалізованих даних. Основним елементом цієї структури є

таблиця MediaLocalization, яка дозволяє зберігати різні мовні версії одного медіа-об'єкта.

Кожен медіа-ресурс може мати декілька перекладених варіантів, що забезпечує гнучку підтримку міжнародних користувачів. Мова визначається через унікальний код, який використовується як ключ для доступу до відповідної версії контенту.

У системі також реалізовано механізм fallback, який дозволяє автоматично відображати контент мовою за замовчуванням у випадку відсутності перекладу.

Для забезпечення високої продуктивності системи застосовуються різні методи оптимізації роботи з даними. Зокрема, передбачено індексацію ключових полів, таких як назва, теги та ідентифікатор мови, що значно прискорює пошук.

Також використовується повнотекстовий пошук, який дозволяє ефективно знаходити контент за змістом. Для зменшення навантаження на базу даних впроваджується кешування часто використовуваних запитів.

Архітектура системи передбачає розділення зберігання медіафайлів і метаданих, що дозволяє масштабувати систему незалежно від типу даних. Додатково може використовуватися зовнішній пошуковий індекс, наприклад Elasticsearch, для покращення продуктивності пошуку.

Загальна ER-модель бази даних відображає взаємозв'язок між основними сутностями системи та їх ролями в предметній області.

Користувачі системи взаємодіють із медіа-контентом, створюючи, редагуючи або переглядаючи його. Кожен медіа-об'єкт має набір метаданих, які описують його характеристики та забезпечують можливість пошуку.

Медіа-контент може існувати у кількох мовних версіях, що реалізується через механізм локалізації. Додатково контент класифікується за категоріями та тегами, що забезпечує зручну навігацію.

Система також реалізує рольову модель доступу, яка визначає рівень прав користувачів і контролює доступ до функціональних можливостей платформи.

Проектування бази даних інформаційної системи управління локалізованою медіа-бібліотекою забезпечує структуроване зберігання мультимедійного контенту та пов'язаних з ним даних. Запропонована модель підтримує багатомовність, ефективний пошук і масштабованість системи. Розділення медіа-файлів, метаданих і локалізації дозволяє підвищити продуктивність і гнучкість системи, що є критично важливим для роботи з великими обсягами цифрового контенту.

2.4. Модель локалізації контенту

Модель локалізації контенту є ключовим елементом інформаційної системи управління локалізованою медіа-бібліотекою, оскільки забезпечує підтримку багатомовності та адаптацію медіа-ресурсів до різних мовних і культурних середовищ. Її основна мета полягає у відокремленні базового медіа-об'єкта від його мовних представлень та забезпеченні гнучкого механізму доступу до локалізованих версій.

Модель локалізації будується на принципі розділення:

- ядра контенту (MediaItem) – базовий медіа-ресурс;
- локалізованих представлень (LocalizedContent) – мовні версії контенту;
- мовного середовища (Language context) – вибрана мова користувача.

Таким чином, один медіа-об'єкт може мати множину локалізованих варіантів без дублювання основного файлу.

Локалізаційна модель базується на ключовій сутності MediaItem, яка представляє базовий медіа-контент системи. Вона містить унікальний ідентифікатор, шлях до файлу, тип медіа (зображення, відео, аудіо або

документ), а також набір загальних метаданих, які не залежать від мови. Такий підхід дозволяє відокремити сам контент від його мовних представлень і забезпечити повторне використання одного й того ж ресурсу в різних локалізаціях.

Сутність Language визначає підтримувані мовні варіанти в системі. Вона включає код мови (наприклад, uk, en, de), її повну назву та статус доступності в межах платформи. Ця таблиця виступає довідником, який використовується для керування мовними версіями контенту та налаштуваннями користувача.

Основною зв'язуючою ланкою є MediaLocalization, яка описує локалізовані варіанти одного медіа-об'єкта. Вона містить посилання на MediaItem та Language, а також локалізовані поля, такі як назва, опис, ключові слова, субтитри для відео та альтернативні аудіодоріжки. Саме ця сутність забезпечує зберігання мовних версій контенту в структурованому вигляді.

Формально модель локалізації можна подати як відображення $L: \text{MediaItem} \times \text{Language} \rightarrow \text{LocalizedContent}$, де кожна пара медіа-об'єкта та мови повертає відповідну локалізовану версію. Якщо така версія відсутня, система застосовує fallback-механізм і повертає контент мовою за замовчуванням.

Процес вибору мови в системі реалізується через механізм language resolution. Спочатку визначається мова користувача на основі профілю, налаштувань браузера або геолокації. Далі система перевіряє наявність відповідного запису в MediaLocalization. Якщо він існує — повертається локалізований контент, якщо ні — використовується резервна мова. Після цього дані передаються на відображення у користувацький інтерфейс.

Локалізація в системі охоплює кілька рівнів. На рівні метаданих виконується переклад назв, описів і ключових слів. На рівні мультимедіа підтримуються субтитри, альтернативні аудіодоріжки та локалізовані елементи графіки. На рівні інтерфейсу здійснюється переклад UI-елементів і

адаптація форматів дат, чисел та напрямків тексту відповідно до мовних стандартів.

Для реалізації моделі можуть застосовуватися різні архітектурні підходи. Таблична модель передбачає зберігання перекладів у реляційній базі даних і забезпечує чітку структуру. JSON-підхід дозволяє зберігати переклади у вигляді гнучких структур, що зручно для API-орієнтованих систем. Гібридна модель поєднує обидва підходи, забезпечуючи баланс між продуктивністю та масштабованістю.

Попри переваги, модель має певні обмеження. Основними проблемами є складність синхронізації перекладів між мовами, збільшення обсягу даних при масштабуванні, а також необхідність підтримки fallback-механізмів. Додатково можуть виникати труднощі через різну довжину текстів, що впливає на інтерфейс користувача.

Водночас запропонована модель має суттєві переваги. Вона забезпечує чітке відокремлення контенту від мовної логіки, дозволяє легко додавати нові мови без змін у структурі системи, підтримує повторне використання медіа-об'єктів і забезпечує гнучкість при масштабуванні. Окремо важливо, що вона повністю підтримує мультимедійну локалізацію, включаючи текст, аудіо та відео контент.

Запропонована модель локалізації контенту забезпечує структурований підхід до реалізації багатомовності в інформаційній системі управління медіа-бібліотекою. Вона дозволяє ефективно розділити базовий контент і його мовні представлення, підтримує різні типи медіа-локалізації та забезпечує гнучкий механізм вибору мови. Це створює основу для побудови масштабованої, універсальної та орієнтованої на міжнародне використання системи.

2.5. Проєктування API

Проєктування API (Application Programming Interface) є важливим етапом розробки інформаційної системи управління локалізованою медіа-бібліотекою, оскільки саме через API забезпечується взаємодія між клієнтською частиною (frontend) та серверною логікою (backend). Добре спроектований API визначає гнучкість, масштабованість та зручність інтеграції системи з іншими сервісами.

При розробці API для системи враховуються такі принципи:

- RESTful-архітектура – використання стандартних HTTP-методів;
- stateless-взаємодія – кожен запит є незалежним;
- єдині правила маршрутизації (naming conventions);
- підтримка багатомовності через параметри запиту;
- безпека доступу через токени (JWT);
- масштабованість та розширюваність.

Для реалізації взаємодії між клієнтською та серверною частинами системи доцільно використовувати REST API, який ґрунтується на ресурсному підході. У цій моделі кожен об'єкт системи розглядається як ресурс, доступ до якого здійснюється через унікальні URL-адреси. Операції над ресурсами виконуються за допомогою стандартних HTTP-методів: GET використовується для отримання даних, POST — для створення нових записів, PUT або PATCH — для оновлення існуючих даних, а DELETE — для їх видалення.

API для роботи з медіа-контентом охоплює базові операції управління файлами. Зокрема, передбачено отримання списку медіа-об'єктів, доступ до конкретного елемента за ідентифікатором, завантаження нових файлів, оновлення метаданих та видалення застарілого контенту. Такий набір ендпоінтів забезпечує повний життєвий цикл медіа-ресурсу в системі.

Окремий блок API відповідає за пошук і фільтрацію даних. Він дозволяє виконувати повнотекстовий пошук за ключовими словами, здійснювати фільтрацію за типом контенту, тегами або мовою, а також отримувати локалізовані результати. Додатково реалізується механізм автодоповнення запитів, що покращує зручність взаємодії користувача з системою.

Функціональність локалізації винесена в окремий набір API-методів. Через них можна отримати список підтримуваних мов, запитати локалізовану версію конкретного медіа-об'єкта, а також додавати або оновлювати переклади. Це дозволяє централізовано керувати багатомовним контентом і забезпечує гнучкість у розширенні мовної підтримки.

API користувачів і авторизації відповідає за безпеку та керування обліковими записами. Воно включає механізми входу та реєстрації, отримання профілю користувача, а також адміністративні функції для перегляду та редагування користувачів. Такий підхід дозволяє розмежувати доступ до функцій системи відповідно до ролей.

Окремо реалізується API для управління категоріями та тегами, яке забезпечує структурування медіа-контенту. Воно дозволяє отримувати списки категорій і тегів, а також створювати нові елементи класифікації, що використовується для покращення пошуку та організації даних.

Обмін даними між клієнтом і сервером здійснюється у форматі JSON, який є легким, структурованим і зручним для парсингу. Він дозволяє передавати як основні параметри медіа-об'єкта, так і його метадані, включаючи назву, тип, мову, теги та дату створення.

Підтримка локалізації в API реалізується через кілька механізмів визначення мови. Мова може передаватися як параметр запиту, через HTTP-заголовок Accept-Language або визначатися на основі профілю користувача. У відповідь система повертає контент у відповідній мовній версії, включаючи перекладені поля.

Безпека API забезпечується за рахунок використання JWT-токенів для авторизації, захищеного HTTPS-з'єднання, рольової моделі доступу (RBAC), обмеження кількості запитів та перевірки вхідних даних. Це дозволяє захистити систему від несанкціонованого доступу та атак.

Для забезпечення масштабованості API передбачає використання кешування відповідей, механізмів пагінації для великих наборів даних, асинхронної обробки великих файлів, а також можливість горизонтального масштабування серверної інфраструктури. Це дозволяє системі ефективно працювати при зростанні навантаження.

Узагальнено архітектура API складається з кількох логічних шарів. Controller Layer відповідає за обробку HTTP-запитів, Service Layer реалізує бізнес-логіку, Repository Layer забезпечує взаємодію з базою даних, а Storage Layer відповідає за зберігання медіафайлів. Така багаторівнева структура забезпечує гнучкість, масштабованість і зручність підтримки системи.

Проектування API інформаційної системи управління локалізованою медіа-бібліотекою забезпечує стандартизований і зручний механізм взаємодії між клієнтською та серверною частинами системи. Використання REST-підходу, JSON-формату та підтримка багатомовності дозволяє створити гнучке, масштабоване та інтегроване рішення, яке відповідає сучасним вимогам до веб-сервісів та медіа-платформ.

Висновки до розділу 2

У другому розділі дипломного проєкта було здійснено проектування інформаційної системи управління локалізованою медіа-бібліотекою, що є ключовим етапом переходу від теоретичного аналізу до практичної реалізації системи. Розглянуті архітектурні рішення, моделі даних, механізми локалізації та інтерфейси взаємодії дозволили сформулювати цілісну структурну основу майбутнього програмного продукту.

На початковому етапі було визначено загальну архітектуру системи, яка базується на багаторівневому підході (frontend, backend, data layer). Така архітектура забезпечує чітке розмежування відповідальності між компонентами, підвищує масштабованість системи та спрощує її подальший супровід і розвиток. Окрему роль у структурі відіграє модуль локалізації, який інтегрується з усіма рівнями системи та забезпечує підтримку багатомовності.

У рамках проєктування бази даних було розроблено логічну модель, що включає основні сутності: користувачі, ролі, медіа-об'єкти, метадані, категорії, теги та мовні версії контенту. Особливу увагу приділено нормалізації структури та забезпеченню цілісності даних. Запропонована модель дозволяє ефективно зберігати великі обсяги мультимедійної інформації та забезпечує швидкий доступ до неї.

Окремо було розроблено модель локалізації контенту, яка базується на розділенні базового медіа-об'єкта та його мовних представлень. Такий підхід забезпечує гнучкість системи, спрощує додавання нових мов та підтримує механізми fallback для відсутніх перекладів. Це є критично важливим для систем, орієнтованих на міжнародне використання.

Також було спроектовано API системи, що забезпечує взаємодію між клієнтською та серверною частинами. Використання REST-архітектури, стандартизованих HTTP-методів і JSON-формату дозволяє забезпечити простоту інтеграції, гнучкість та масштабованість системи. Окремо враховано підтримку локалізації на рівні API та механізми безпеки, включаючи JWT-автентифікацію та рольовий доступ.

Отже, результати другого розділу формують повну архітектурно-логічну модель інформаційної системи управління локалізованою медіа-бібліотекою. Розроблені рішення забезпечують основу для подальшої програмної реалізації системи, яка буде представлена у наступному розділі.

РОЗДІЛ 3. РОЗРОБКА ТА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1. Вибір технологій

Вибір технологій для реалізації інформаційної системи управління локалізованою медіа-бібліотекою є важливим етапом, який безпосередньо впливає на продуктивність, масштабованість, безпеку та зручність подальшої підтримки системи. Технологічний стек формується з урахуванням вимог до роботи з мультимедійним контентом, підтримки багатомовності, високого навантаження та інтеграції через API.

При формуванні технологічного стеку враховуються такі критерії:

- продуктивність обробки даних і медіа-контенту;
- масштабованість системи при зростанні навантаження;
- підтримка REST API та інтеграцій;
- наявність інструментів для локалізації (i18n/l10n);
- рівень безпеки та підтримка авторизації;
- екосистема та популярність технології;
- простота підтримки та розширення системи.

Для реалізації інформаційної системи управління локалізованою медіа-бібліотекою обрано сучасний технологічний стек, який забезпечує високу продуктивність, масштабованість, підтримку мультимедійного контенту та багатомовності. Основою системи виступає клієнт-серверна архітектура з REST API, що дозволяє чітко розділити відповідальність між інтерфейсом користувача, серверною логікою та рівнем зберігання даних. Такий підхід є найбільш доцільним для систем, орієнтованих на роботу з великими обсягами даних і необхідністю інтеграції з різними клієнтськими платформами.

У якості серверної частини обрано платформу Node.js із використанням фреймворку NestJS. Вибір Node.js обґрунтований його асинхронною моделлю обробки запитів, що дозволяє ефективно працювати з великою кількістю

одночасних користувачів та операціями введення-виведення, зокрема завантаженням і передачею медіа-файлів. NestJS, у свою чергу, додає структурованість архітектурі, підтримує модульний підхід і дозволяє будувати масштабовані enterprise-рівня застосунки з чітким розподілом бізнес-логіки. Використання TypeScript у цьому стеку підвищує надійність коду та зменшує кількість помилок завдяки статичній типізації.

Для реалізації клієнтської частини системи обрано React.js із TypeScript. Такий вибір зумовлений компонентною архітектурою React, яка дозволяє створювати гнучкий та повторно використовуваний інтерфейс користувача. Virtual DOM забезпечує високу продуктивність рендерингу, що є важливим при роботі з динамічними медіа-бібліотеками. Використання TypeScript додатково підвищує стабільність коду та спрощує масштабування фронтенд-частини системи. Для організації навігації та HTTP-взаємодії застосовуються React Router та Axios, а для реалізації багатомовного інтерфейсу використовується бібліотека i18next.

У якості системи управління базою даних обрано PostgreSQL, яка є однією з найнадійніших реляційних СУБД. Її використання обумовлене необхідністю зберігання складних зв'язків між сутностями, такими як користувачі, ролі, медіа-об'єкти, метадані та мовні версії контенту. Додатковою перевагою PostgreSQL є підтримка JSONB-структур, що дозволяє гнучко зберігати напівструктуровані дані без втрати продуктивності. Це особливо важливо для роботи з метаданими медіа-контенту, які можуть мати різну структуру залежно від типу файлу.

Для реалізації ефективного пошуку по великому обсягу контенту використовується Elasticsearch, який забезпечує повнотекстовий пошук, індексацію даних та швидке формування релевантних результатів. Його застосування дозволяє значно підвищити продуктивність пошукових операцій, особливо при роботі з локалізованими метаданими, тегами та

описами медіа-об'єктів. Це рішення є критично важливим для забезпечення швидкого доступу до інформації в межах медіа-бібліотеки.

Зберігання медіафайлів організовано через об'єктне сховище Amazon S3 або його сумісні аналоги. Такий підхід дозволяє розділити зберігання великих файлів і метаданих, що зменшує навантаження на базу даних та забезпечує високу масштабованість системи. Додатково використання S3-сумісних сховищ дозволяє інтегрувати CDN для швидкої доставки контенту кінцевим користувачам незалежно від їхнього географічного розташування.

Безпека системи реалізується за допомогою JWT-аутентифікації та хешування паролів із використанням bcrypt. JWT забезпечує stateless-авторизацію, що спрощує масштабування системи без необхідності зберігання сесій на сервері. Використання bcrypt гарантує надійний захист облікових даних користувачів. Додатково передбачено використання HTTPS для шифрування трафіку та рольову модель доступу, яка обмежує можливості користувачів відповідно до їхніх прав.

Підтримка локалізації реалізується на двох рівнях: на стороні клієнта використовується бібліотека i18next для динамічного перемикання мови інтерфейсу, а на сервері – спеціальна модель бази даних, що зберігає локалізовані версії метаданих медіа-об'єктів. Такий підхід забезпечує гнучкість системи та можливість легкого додавання нових мов без зміни бізнес-логіки. Це дозволяє стандартизувати середовище виконання, спростити процес розгортання та забезпечити однакову поведінку системи на різних етапах життєвого циклу розробки.

Узагальнюючи, запропонований технологічний стек включає React.js та TypeScript для фронтенду, Node.js із NestJS для бекенду, PostgreSQL як основну базу даних, Elasticsearch для пошуку, Amazon S3 для зберігання медіафайлів. Такий набір технологій забезпечує необхідний рівень продуктивності, масштабованості, безпеки та підтримки багатомовності, що є

критично важливим для реалізації сучасної інформаційної системи управління локалізованою медіа-бібліотекою.

Обраний стек забезпечує:

- високу продуктивність (Node.js + Elasticsearch);
- масштабованість (S3 + stateless API);
- підтримку локалізації (i18next + DB-модель);
- надійне зберігання даних (PostgreSQL + S3);
- зручність розробки та підтримки (NestJS + TypeScript);
- готовність до enterprise-рівня системи.

Запропонований технологічний стек є сучасним, збалансованим і орієнтованим на роботу з великими обсягами мультимедійного контенту. Він дозволяє реалізувати інформаційну систему управління локалізованою медіа-бібліотекою як масштабовану, безпечну та багатомовну платформу з можливістю подальшого розвитку та інтеграції з іншими сервісами.

3.2. Реалізація серверної частини

Реалізація серверної частини інформаційної системи управління локалізованою медіа-бібліотекою є центральним етапом практичної розробки, оскільки саме backend забезпечує обробку бізнес-логіки, управління даними, роботу з медіа-контентом, реалізацію локалізації та взаємодію з клієнтською частиною через API. Серверна частина побудована на основі Node.js із використанням фреймворку NestJS, що дозволяє реалізувати модульну, масштабовану та структуровану архітектуру.

Основою серверної реалізації є поділ системи на функціональні модулі, кожен з яких відповідає за окрему підсистему. Такий підхід дозволяє ізолювати бізнес-логіку та спростити подальше розширення функціональності. Зокрема, виділяються модулі управління медіа-контентом, автентифікації, пошуку, локалізації та управління користувачами. Кожен

модуль має власний набір контролерів, сервісів і репозиторіїв, що відповідають за обробку запитів, реалізацію логіки та взаємодію з базою даних.

Наприклад MediaModule відповідає за роботу з медіафайлами (Рис. 3.1).

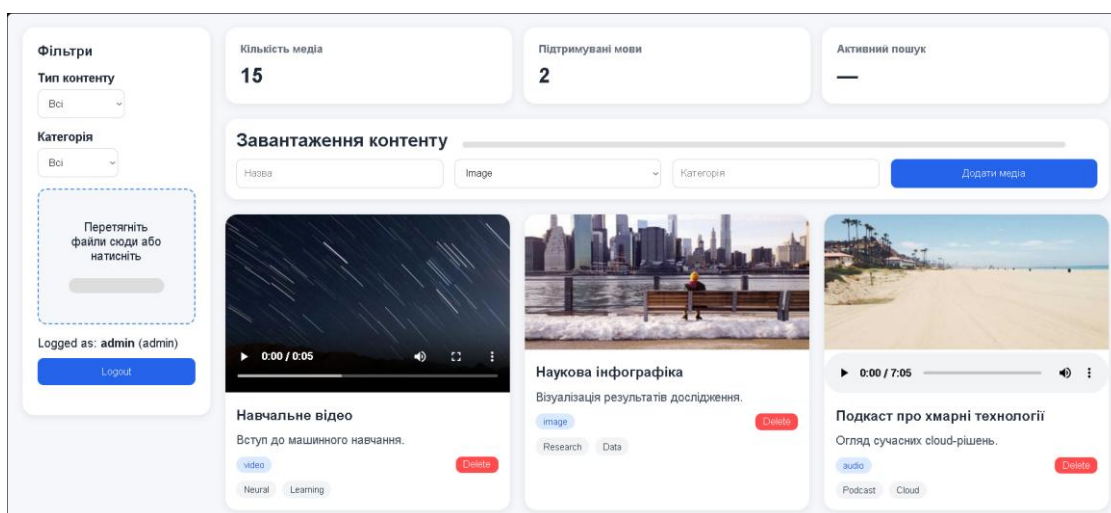


Рис. 3.1. Загальний вигляд інформаційної системи медіа-бібліотеки

Обробка HTTP-запитів реалізується через REST API, де контролери приймають запити від клієнта, передають їх до сервісного шару, а результати повертають у форматі JSON. Така структура забезпечує чітке розділення відповідальності та дозволяє легко масштабувати систему. Наприклад, модуль медіа-контенту відповідає за завантаження файлів, отримання списку медіа, оновлення метаданих та видалення ресурсів, тоді як модуль пошуку інтегрується з Elasticsearch для виконання повнотекстових запитів:

```
async search(query: string) {
  return this.elasticsearchService.search({
    index: 'media',
    query: {
      multi_match: {
        query,
```

```

        fields: ['title', 'description', 'tags'],
      },
    },
  ));
}

```

Особливу увагу приділено реалізації роботи з медіафайлами. Безпосереднє зберігання великих файлів у базі даних є недоцільним, тому вони зберігаються у зовнішньому об'єктному сховищі, а в базі даних зберігаються лише посилання та метадані. Завантаження файлів реалізується через окремий сервіс, який виконує перевірку формату, розміру та безпеки контенту перед його збереженням.

Серверна частина також включає реалізацію системи автентифікації та авторизації. Для цього використовується JWT-токенізація, яка забезпечує stateless-підхід до керування сесіями користувачів:

```

@Injectable()
export class AuthService {
  constructor(private jwtService: JwtService) {}
  async login(user: any) {
    const payload = { username: user.username, sub:
user.id };
    return {
      access_token: this.jwtService.sign(payload),
    };
  }
}

```

Після успішної авторизації користувач отримує токен, який передається у кожному запиті для перевірки прав доступу. Додатково реалізовано рольову модель доступу, що розмежовує функціональність між адміністраторами, редакторами та звичайними користувачами.

Важливим компонентом серверної частини є реалізація локалізації. Вона забезпечується через зберігання мовних версій метаданих у базі даних та механізм визначення мови користувача:

```
@Get('/:id/localized')
async getLocalized(
  @Param('id') id: number,
  @Query('lang') lang: string,
) {
  return this.mediaService.getLocalizedMedia(id, lang);
}
```

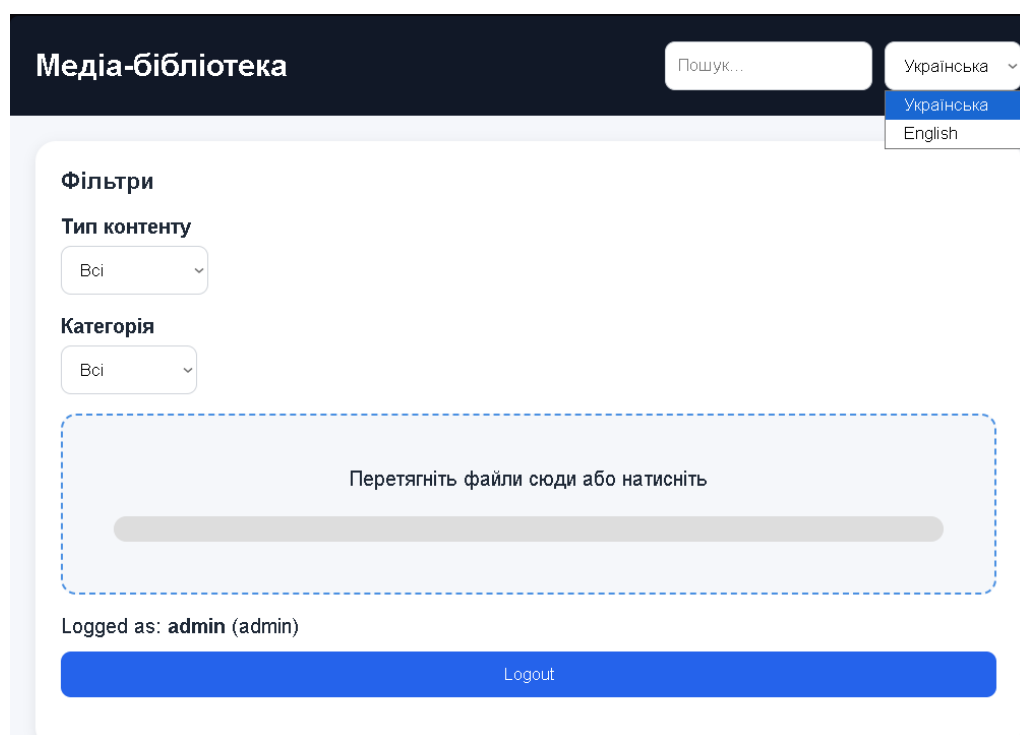


Рис. 3.2. Перемикання мови інтерфейсу

При надходженні запиту система перевіряє бажану мову та повертає відповідну локалізовану версію контенту. У разі відсутності перекладу

застосовується fallback-механізм, який повертає контент основною мовою системи.

Для забезпечення продуктивності серверної частини використовується кешування часто запитуваних даних, індексація в базі даних, а також інтеграція з Elasticsearch для швидкого пошуку. Це дозволяє суттєво зменшити навантаження на базу даних і прискорити обробку запитів користувачів навіть при великому обсязі медіа-контенту.

Для роботи з медіафайлами використовується багаточастинне завантаження (Рис. 3.3).

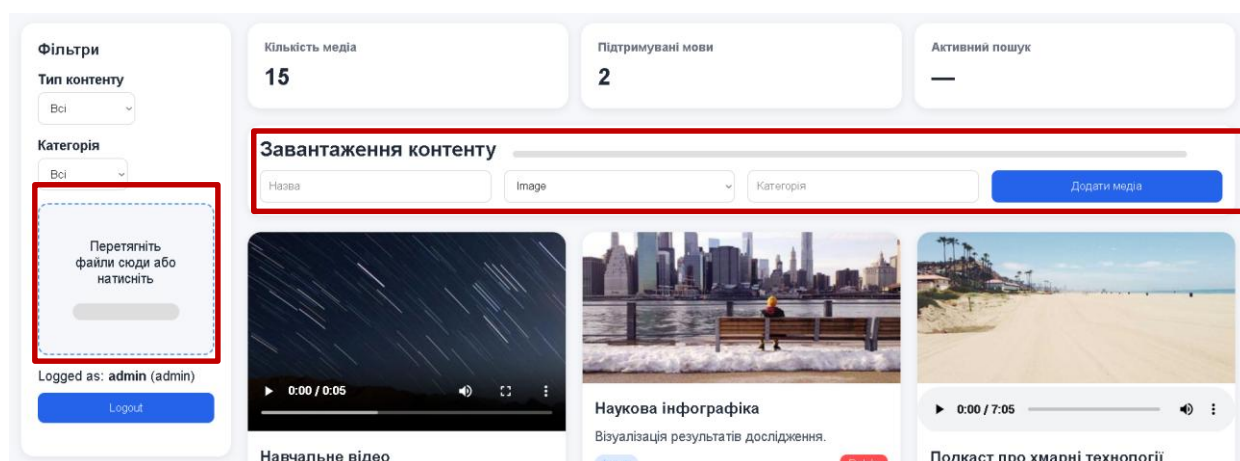


Рис. 3.3. Інтерфейс завантаження медіаконтенту

Локалізація реалізується через окрему сутність MediaLocalization:

```
@Entity()
```

```
export class MediaLocalization {
```

```
    @PrimaryGeneratedColumn()
```

```
    id: number;
```

```
    @ManyToOne(() => MediaEntity)
```

```
    media: MediaEntity;
```

```
    @Column()
```

```
    language: string;
```

```

@Column()
title: string;
@Column()
description: string;
}

```

Реалізація серверної частини базується на модульній архітектурі NestJS, що забезпечує структурованість, масштабованість і легкість супроводу системи. Використання REST API дозволяє ефективно взаємодіяти з клієнтською частиною, а інтеграція з PostgreSQL, Elasticsearch та системою файлового зберігання забезпечує повноцінну роботу з медіа-контентом. Окремо реалізовані механізми локалізації та JWT-автентифікації підвищують гнучкість і безпеку системи, що є критично важливим для сучасних інформаційних платформ.

3.3. Реалізація клієнтської частини

Реалізація клієнтської частини інформаційної системи управління локалізованою медіа-бібліотекою спрямована на забезпечення зручної взаємодії користувача з системою, відображення медіа-контенту, реалізацію пошуку, фільтрації та підтримку багатомовного інтерфейсу. Для розробки frontend-частини використано React.js із TypeScript, що дозволяє створити компонентну, масштабовану та легко підтримувану архітектуру користувацького інтерфейсу.

Клієнтський застосунок побудований як SPA (Single Page Application), що забезпечує швидке перемикання між сторінками без повного перезавантаження. Архітектура інтерфейсу базується на поділі на компоненти, сторінки, сервіси для роботи з API та модулі локалізації.

Основними частинами є компоненти інтерфейсу, сторінки, сервіси для HTTP-запитів та система управління станом застосунку. Такий підхід спрощує

масштабування системи. Структура проекту організована таким чином, щоб відокремити логіку, представлення та сервіси (рис. 3.4).

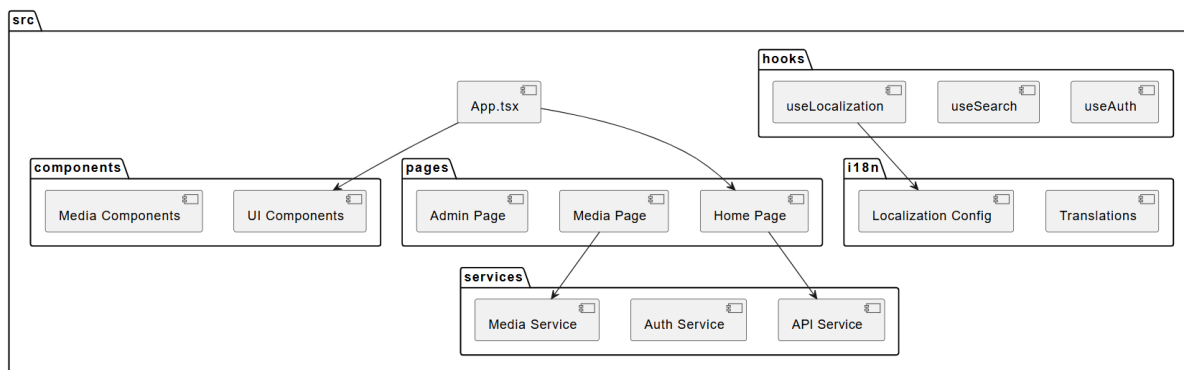


Рис. 3.4. Структура клієнтської частини інформаційної системи

Для взаємодії з серверною частиною використовується Axios, який дозволяє виконувати HTTP-запити до REST API.

```

import axios from "axios";
const api = axios.create({
  baseURL: "http://localhost:3000/api",
});
export const getMedia = async () => {
  const response = await api.get("/media");
  return response.data;
};
  
```

Компонент для відображення списку медіа-контенту використовує `useEffect` для завантаження даних при ініціалізації.

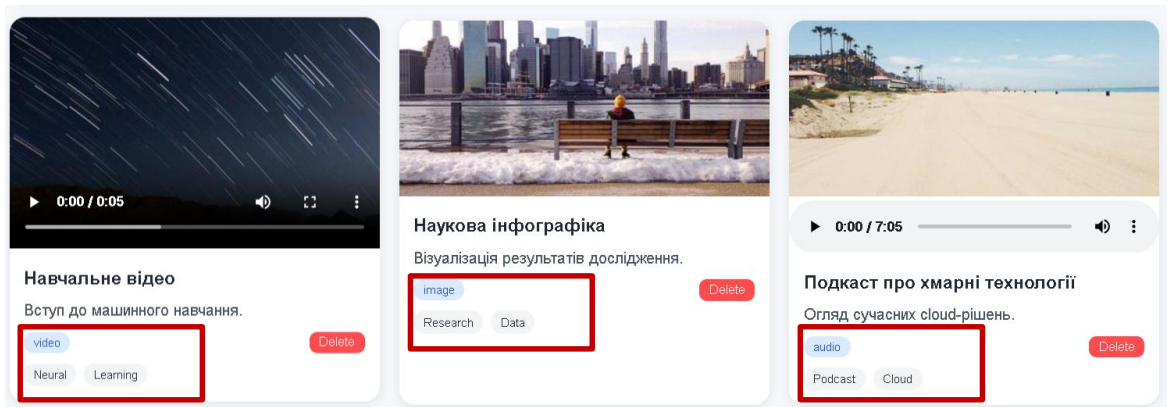


Рис. 3.5. Відображення медіаоб'єктів у клієнтському інтерфейсі

Клієнтська частина реалізує динамічний пошук із передачею параметрів на сервер.

```
export const searchMedia = async (query: string) => {
  const response = await api.get(`/search?q=${query}`);
  return response.data;
};

const [query, setQuery] = useState("");
const handleSearch = async () => {
  const results = await searchMedia(query);
  setMedia(results);
};
```

Для підтримки багатомовності використовується бібліотека i18next, яка дозволяє динамічно змінювати мову інтерфейсу.

```
import i18n from "i18next";
import { initReactI18next } from "react-i18next";
i18n.use(initReactI18next).init({
  resources: {
    en: {
      translation: {
        title: "Media Library",
```

```

    },
  },
  uk: {
    translation: {
      title: "Медіа-бібліотека",
    },
  },
},
lng: "uk",
fallbackLng: "en",
});

```

Для реалізації багатомовності у клієнтській частині застосунку використовується бібліотека `react-i18next`, яка забезпечує інтеграцію механізмів локалізації в компоненти React. Хук `useTranslation` дозволяє отримувати доступ до функцій перекладу безпосередньо всередині компонента, що спрощує динамічне відображення тексту відповідно до вибраної мови користувача:

```

import { useTranslation } from "react-i18next";
export const Header = () => {
  const { t, i18n } = useTranslation();
  return (
    <header>
      <h1>{t("title")}</h1>
      <button onClick={() =>
i18n.changeLanguage("uk")}>UA</button>
      <button onClick={() =>
i18n.changeLanguage("en")}>EN</button>
    </header>

```

```
);  
};
```

За допомогою цього механізму всі текстові елементи інтерфейсу, такі як кнопки, повідомлення, меню та заголовки, можуть автоматично змінюватися залежно від активної локалізації. Крім того, використання `react-i18next` забезпечує підтримку перемикання мов у реальному часі без перезавантаження сторінки, що покращує зручність роботи користувача та відповідає сучасним вимогам до багатомовних вебзастосунків.

Для відображення медіа використовується умовний рендеринг залежно від типу файлу.

```
const MediaItem = ({ item }: any) => {  
  if (item.type === "image") {  
    return <img src={item.url} alt={item.title} />;  
  }  
  if (item.type === "video") {  
    return <video controls src={item.url} />;  
  }  
  return <a href={item.url}>{item.title}</a>;  
};
```

Для великих систем можливе використання `Redux` або `Context API` для глобального стану. Це дозволяє ефективно керувати:

- авторизацією користувача;
- мовою інтерфейсу;
- кешуванням даних;
- станом завантаження.

Також застосовуються оптимізації у вигляді лінивого завантаження компонентів та мемоізації рендерингу.

Клієнтська частина інформаційної системи реалізована на основі React.js із використанням TypeScript, що забезпечує високу продуктивність, модульність та зручність розробки. Використання REST API дозволяє ефективно взаємодіяти із серверною частиною, а інтеграція i18next забезпечує повноцінну підтримку багатомовного інтерфейсу. Реалізована архітектура забезпечує гнучкість, масштабованість і зручність використання системи кінцевими користувачами.

3.4. Інтеграція системи

Інтеграція інформаційної системи управління локалізованою медіа-бібліотекою є завершальним етапом реалізації програмного продукту, що забезпечує взаємодію всіх компонентів системи в єдиному середовищі. Основною метою інтеграції є об'єднання клієнтської частини, серверної логіки, бази даних, системи пошуку, сховища медіафайлів та модулів локалізації в цілісну функціональну платформу.

Інтеграція системи базується на багаторівневій архітектурі, де кожен рівень взаємодіє через чітко визначені інтерфейси. Основним механізмом взаємодії між клієнтом і сервером виступає REST API, що забезпечує стандартизований обмін даними у форматі JSON. Такий підхід дозволяє легко масштабувати систему та інтегрувати додаткові сервіси без зміни основної архітектури.

Взаємодія між клієнтською та серверною частинами реалізується через HTTP-запити до API. Клієнтський застосунок надсилає запити на отримання, створення або оновлення даних, а сервер обробляє їх і повертає відповідь у структурованому форматі.

Приклад інтеграції через API-запит:

```
import axios from "axios";  
const api = axios.create({
```

```

    baseUrl: "http://localhost:3000/api",
    headers: {
      "Content-Type": "application/json",
    },
  });
export const fetchMedia = async () => {
  const response = await api.get("/media");
  return response.data;
};

```

Таким чином забезпечується повна взаємодія між UI та серверною логікою в режимі реального часу.

Серверна частина взаємодіє з базою даних PostgreSQL через ORM-рівень (наприклад, TypeORM або Prisma). Це дозволяє абстрагуватися від SQL-запитів і працювати з даними як з об'єктами.

Приклад інтеграції:

```

@Injectable()
export class MediaService {
  constructor(
    @InjectRepository(MediaEntity)
    private mediaRepository: Repository<MediaEntity>,
  ) {}
  async findAll() {
    return this.mediaRepository.find({
      relations: ["localizations"],
    });
  }
}

```

Такий підхід забезпечує просту інтеграцію бізнес-логіки з рівнем зберігання даних.

Система пошуку інтегрується через Elasticsearch, який використовується як окремий сервіс індексації. Дані з PostgreSQL синхронізуються з пошуковим індексом для забезпечення швидкого доступу до контенту.

```
async search(query: string) {  
  return this.elasticsearchService.search({  
    index: "media",  
    query: {  
      multi_match: {  
        query,  
        fields: ["title", "description", "tags"],  
      },  
    },  
  });  
}
```

Інтеграція дозволяє реалізувати повнотекстовий пошук із високою швидкістю відповіді навіть при великих обсягах даних.

Медіафайли зберігаються у зовнішньому об'єктному сховищі (наприклад, Amazon S3 або MinIO). Серверна частина відповідає лише за завантаження файлів та збереження посилань у базі даних.

```
async uploadFile(file: Express.Multer.File) {  
  const result = await this.s3.upload({  
    Bucket: "media-library",  
    Key: file.filename,  
    Body: file.buffer,  
  }).promise();  
}
```

```

    return result.Location;
}

```

Такий підхід дозволяє розвантажити сервер і забезпечити масштабоване зберігання медіаконтенту.

Модуль локалізації інтегрується на рівні API та бази даних. При кожному запиті система визначає мову користувача та повертає відповідну мовну версію контенту.

```

async getLocalizedMedia(id: number, lang: string) {
  const media = await this.mediaRepository.findOne({
    where: { id },
    relations: ["localizations"],
  });
  return media.localizations.find(l => l.language ===
lang)
  || media.localizations.find(l => l.language ===
"en");
}

```

Це забезпечує автоматичну адаптацію контенту до мовних налаштувань користувача.

У результаті інтеграції формується єдина екосистема, в якій:

- frontend взаємодіє з backend через REST API;
- backend керує бізнес-логікою та даними;
- PostgreSQL зберігає структуровану інформацію;
- Elasticsearch забезпечує швидкий пошук;
- S3-сховище відповідає за медіафайли;
- модуль локалізації забезпечує багатомовність;

Інтеграція системи забезпечує узгоджену роботу всіх її компонентів і перетворює набір окремих модулів на цілісну інформаційну платформу.

Використання REST API, мікросервісного підходу та контейнеризації дозволяє досягти високої гнучкості, масштабованості та надійності системи. Запропонована інтеграційна модель забезпечує стабільну взаємодію між підсистемами та створює основу для подальшого розвитку і розширення функціональності.

3.5. Безпека системи

Безпека інформаційної системи управління локалізованою медіа-бібліотекою є критично важливим аспектом, оскільки система працює з користувацькими даними, мультимедійним контентом, а також забезпечує доступ до адміністративних функцій. Основною метою реалізації безпеки є захист даних від несанкціонованого доступу, забезпечення цілісності інформації та гарантування безпечної взаємодії між клієнтською та серверною частинами.

Архітектура безпеки системи базується на багаторівневому підході, який включає автентифікацію користувачів, авторизацію доступу, захист даних під час передачі та зберігання, а також контроль вхідних даних. Такий підхід дозволяє мінімізувати ризики атак і забезпечити стабільну роботу системи навіть при зовнішніх загрозах.

Для автентифікації користувачів використовується механізм JSON Web Token (JWT), який дозволяє реалізувати stateless-авторизацію без збереження сесій на сервері. Після успішного входу користувач отримує токен, який передається у кожному запиті до API.

```
async login(user: User) {  
  const payload = { username: user.username, sub:  
user.id };  
  return {  
    access_token: this.jwtService.sign(payload),
```

} ;
}

Токен перевіряється на сервері при кожному запиті, що забезпечує контроль доступу до ресурсів системи.

Система використовує рольову модель доступу (RBAC), яка визначає права користувачів залежно від їх ролі. Наприклад, адміністратор має повний доступ до системи, редактор може завантажувати та редагувати медіа, а звичайний користувач має лише права перегляду.

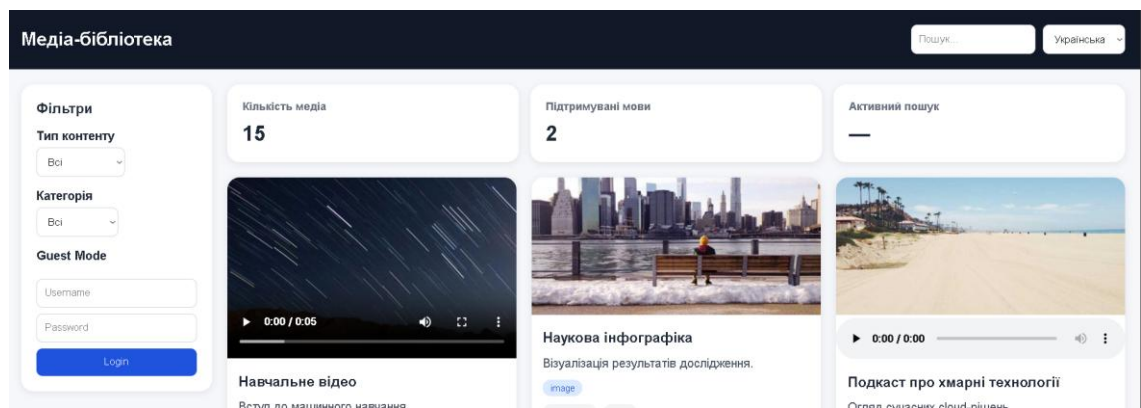


Рис. 3.6. Форма автентифікації користувача (Guest mode)

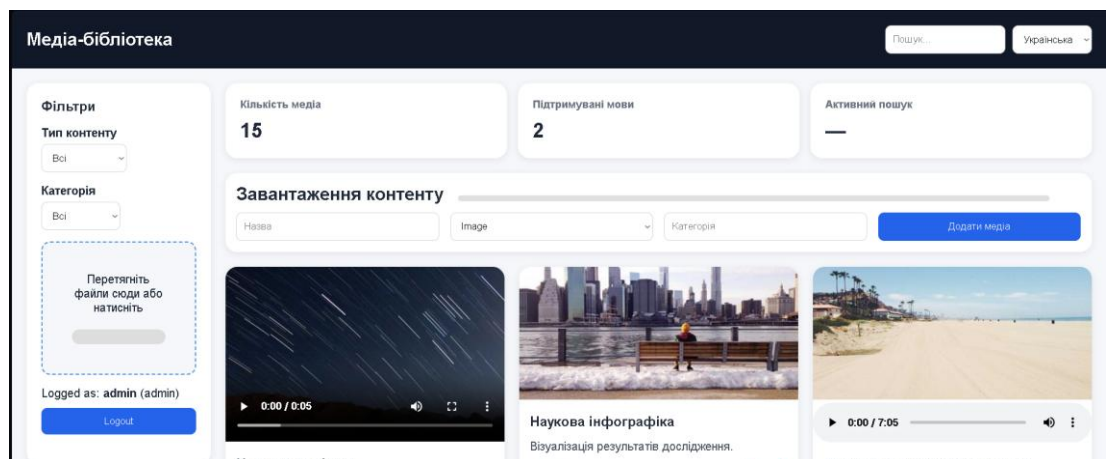


Рис. 3.7. Відображення функцій адміністратора після авторизації

Такий підхід дозволяє чітко розмежувати доступ до функціональності системи.

Для зберігання паролів використовується алгоритм хешування bcrypt, що забезпечує неможливість відновлення оригінального пароля навіть у разі витоку бази даних.

```
import * as bcrypt from 'bcrypt';
async hashPassword(password: string) {
  const salt = await bcrypt.genSalt();
  return bcrypt.hash(password, salt);
}
```

Це значно підвищує рівень безпеки облікових записів користувачів.

Уся взаємодія між клієнтом і сервером здійснюється через HTTPS-протокол, який забезпечує шифрування даних за допомогою TLS. Це запобігає перехопленню інформації під час передачі, зокрема токенів авторизації та персональних даних користувачів.

Система додатково захищена від поширених веб-загроз. Для запобігання SQL-ін'єкціям використовується ORM-рівень (наприклад, TypeORM), який автоматично екранує запити. Для захисту від XSS-атак застосовується фільтрація та валідація вхідних даних на сервері та клієнті. Також реалізовано захист від CSRF шляхом використання токенів перевірки запитів.

Перед обробкою всі вхідні дані проходять валідацію, що дозволяє уникнути некоректних або шкідливих запитів.

```
import { IsString, IsNotEmpty } from 'class-validator';
export class CreateMediaDto {
  @IsString()
  @IsNotEmpty()
  title: string;
}
```

Це забезпечує додатковий рівень захисту на рівні бізнес-логіки.

Медіафайли перевіряються перед завантаженням на відповідність дозволеним форматам і розмірам. Також обмежується доступ до файлів через авторизовані посилання, що запобігає несанкціонованому завантаженню контенту.

Реалізована система безпеки інформаційної системи управління локалізованою медіа-бібліотекою базується на багаторівневому підході, який включає автентифікацію, авторизацію, шифрування даних, захист від веб-атак та валідацію вхідних даних. Використання JWT, bcrypt, HTTPS та рольової моделі доступу забезпечує високий рівень захищеності системи. Запропоновані механізми дозволяють гарантувати цілісність, конфіденційність і доступність даних, що є критично важливим для сучасних інформаційних платформ.

Висновки до розділу 3

У третьому розділі дипломного проєкта було здійснено практичну реалізацію інформаційної системи управління локалізованою медіа-бібліотекою, що включає розробку серверної та клієнтської частин, інтеграцію компонентів системи та забезпечення її безпеки. Розділ відображає перехід від проєктних моделей, сформованих у попередньому розділі, до реалізації функціонального програмного продукту.

На етапі вибору технологій було обґрунтовано використання сучасного стеку, який включає Node.js та NestJS для серверної частини, React.js і TypeScript для клієнтської частини, PostgreSQL як основну базу даних, Elasticsearch для реалізації пошуку, а також S3-сумісне сховище для зберігання медіафайлів. Запропонований стек забезпечує баланс між продуктивністю, масштабованістю та зручністю розробки, а також повністю відповідає вимогам до роботи з мультимедійним і багатомовним контентом.

У процесі реалізації серверної частини було створено модульну архітектуру на основі NestJS, що забезпечує чітке розділення бізнес-логіки, роботи з даними та обробки HTTP-запитів. Реалізовано основні функціональні модулі системи, включаючи управління медіа-контентом, автентифікацію користувачів, пошук та підтримку локалізації. Додатково було інтегровано механізми роботи з файловим сховищем і системою індексації даних.

Клієнтська частина системи реалізована як SPA-додаток на основі React.js, що забезпечує швидку взаємодію користувача з системою, динамічне оновлення інтерфейсу та зручну навігацію. Окрему увагу приділено реалізації багатомовного інтерфейсу, який дозволяє адаптувати систему до різних мовних середовищ за допомогою i18next та серверної підтримки локалізації.

Інтеграція системи забезпечила узгоджену взаємодію всіх її компонентів через REST API, що дозволило об'єднати клієнтську частину, сервер, базу даних, пошуковий механізм і файлове сховище в єдину екосистему. Використання контейнеризації на основі Docker спростило процес розгортання та підвищило переносимість системи.

Окремо було реалізовано комплекс заходів із забезпечення безпеки, що включає JWT-автентифікацію, рольову модель доступу, хешування паролів, шифрування даних через HTTPS, а також захист від основних веб-загроз. Це забезпечує надійний рівень захищеності системи та контроль доступу до ресурсів.

Отже, результати третього розділу підтверджують можливість практичної реалізації запропонованої архітектури інформаційної системи управління локалізованою медіа-бібліотекою. Створена система є масштабованою, безпечною та функціонально повною, що дозволяє ефективно працювати з мультимедійним контентом у багатомовному середовищі та забезпечує основу для подальшого розвитку проєкту.

РОЗДІЛ 4. ЕСТУВАННЯ ТА ОЦІНКА ЕФЕКТИВНОСТІ

4.1. Методика тестування

Методика тестування інформаційної системи управління локалізованою медіа-бібліотекою визначає підхід до перевірки коректності, стабільності та ефективності роботи розробленого програмного забезпечення. Основною метою тестування є виявлення помилок на різних рівнях системи, перевірка відповідності функціональних і нефункціональних вимог, а також оцінювання якості реалізації ключових модулів, включаючи роботу з медіа-контентом, локалізацію, пошук і безпеку.

У процесі тестування застосовується комбінований підхід, який включає модульне, інтеграційне та системне тестування. Модульне тестування спрямоване на перевірку окремих компонентів системи, таких як сервіси роботи з медіа, модуль автентифікації та модуль локалізації. Воно дозволяє виявити логічні помилки на ранніх етапах розробки та забезпечити коректність роботи кожного окремого елемента системи.

Модульне тестування застосовується для перевірки окремих сервісів системи, наприклад, сервісу роботи з медіа-контентом. Для цього використовується Jest, який є стандартним інструментом тестування для Node.js/NestJS.

Приклад тестування сервісу отримання медіа:

```
import { MediaService } from './media.service';

describe('MediaService', () => {
  let service: MediaService;

  beforeEach(() => {
    service = new MediaService(null as any);
  });
});
```

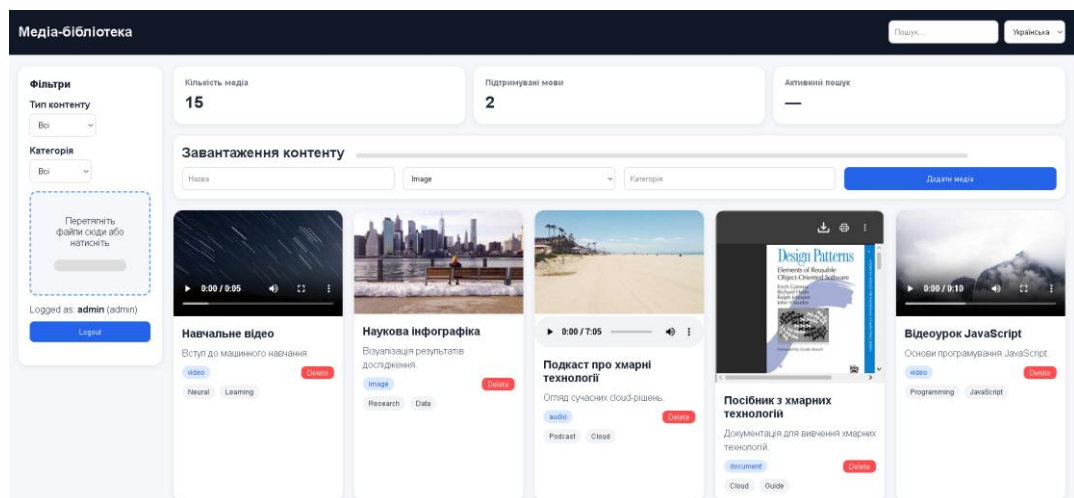


Рис. 4.1. Перевірка відображення списку медіаоб'єктів

Цей тест перевіряє базову коректність роботи методу отримання списку медіа-об'єктів.

Особлива увага приділяється перевірці механізму локалізації, оскільки система підтримує багатомовність:

```
it('should return localized content or fallback', async
() => {
  const result = await service.getLocalizedMedia(1,
'uk');
  expect(result.language === 'uk' || result.language
=== 'en').toBeTruthy();
});
```

Цей тест перевіряє коректність роботи fallback-механізму у випадку відсутності перекладу.

Інтеграційне тестування використовується для перевірки взаємодії між модулями системи. Особлива увага приділяється взаємодії між клієнтською та серверною частинами через REST API, а також інтеграції бази даних, системи пошуку та файлового сховища. На цьому етапі перевіряється коректність

передачі даних, узгодженість форматів та стабільність роботи всієї системи при комплексному навантаженні.

Інтеграційне тестування перевіряє взаємодію між клієнтом і сервером через REST API:

```
import request from 'supertest';
import { app } from '../app';
describe('Media API', () => {
  it('GET /media should return 200', async () => {
    const response = await
request(app).get('/api/media');
    expect(response.status).toBe(200);
    expect(Array.isArray(response.body)).toBe(true);
  });
});
```

Цей тест перевіряє коректність роботи HTTP-ендпоінта та формат відповіді.

Системне тестування передбачає перевірку системи в цілому як єдиного програмного продукту. Воно дозволяє оцінити відповідність реалізованої системи поставленим вимогам, включаючи функціональні можливості, продуктивність, безпеку та підтримку багатомовності. На цьому етапі моделюються реальні сценарії використання системи кінцевими користувачами.

Перевіряється захист маршрутів та доступ без токена.

```
it('should deny access without token', async () => {
  const response = await
request(app).get('/api/profile');
  expect(response.status).toBe(401);
});
```

Цей тест гарантує, що захищені маршрути недоступні без авторизації.

Перевіряється інтеграція з пошуковим модулем Elasticsearch.

```
it('should return search results', async () => {  
  const response = await request(app)  
    .get('/api/search?q=video');  
  expect(response.status).toBe(200);  
  expect(response.body).toBeDefined();  
});
```

Для фронтенд використовується React Testing Library.

```
import { render, screen } from '@testing-  
library/react';  
import { MediaList } from './MediaList';  
test('renders media list', () => {  
  render(<MediaList />);  
  const element = screen.getByText(/media/i);  
  expect(element).toBeInTheDocument();  
});
```

Для перевірки продуктивності використовується, наприклад, k6 або аналогічні інструменти.

```
import http from 'k6/http';  
export default function () {  
  http.get('http://localhost:3000/api/media');  
}
```

Це дозволяє оцінити стабільність API при великій кількості одночасних запитів.

Окрему роль у методиці тестування відіграє тестування локалізації, яке перевіряє коректність відображення контенту різними мовами, роботу механізму fallback, а також узгодженість перекладів інтерфейсу та медіа-

метаданих. Це дозволяє гарантувати правильну роботу системи в умовах багатомовного середовища.

Також здійснюється тестування продуктивності, метою якого є оцінювання швидкості обробки запитів, часу завантаження медіа-контенту та ефективності пошукових операцій. Для цього моделюються сценарії одночасного доступу великої кількості користувачів до системи.

Методика тестування інформаційної системи базується на комплексному підході, який включає модульне, інтеграційне, системне та навантажувальне тестування. Використання автоматизованих тестів на основі Jest, Supertest та React Testing Library дозволяє забезпечити високу якість програмного продукту. Окремо перевіряється робота механізмів локалізації, авторизації та пошуку, що є критично важливими для коректного функціонування системи в умовах багатомовного та високонавантаженого середовища.

4.2. Тестування функціональності

Тестування функціональності інформаційної системи управління локалізованою медіа-бібліотекою спрямоване на перевірку коректності реалізації всіх основних бізнес-функцій системи. До таких функцій належать завантаження та відображення медіа-контенту, пошук і фільтрація, робота з локалізацією, автентифікація користувачів, а також управління даними через API. Основною метою функціонального тестування є підтвердження того, що система поводить себе відповідно до визначених вимог у реальних сценаріях використання.

Перевіряється коректність процесу завантаження файлів, збереження метаданих та формування запису в базі даних. Особлива увага приділяється валідації типу файлу та його розміру:

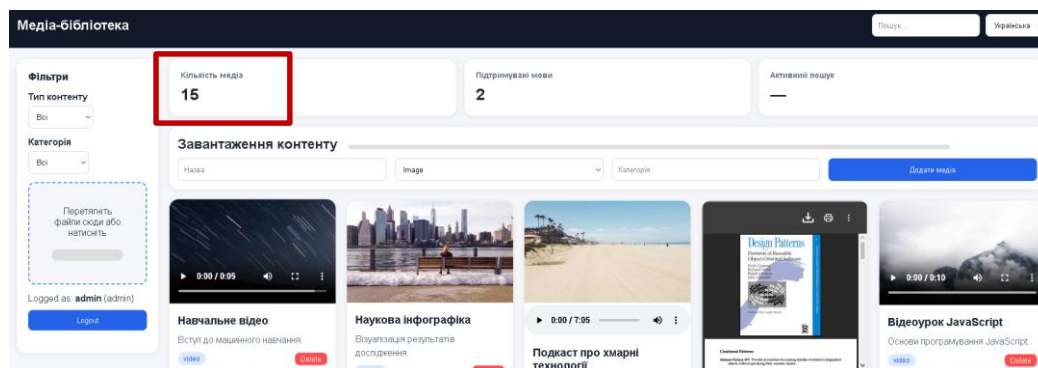


Рис. 4.2. Результат перед додаванням медіа (к-ть 15).

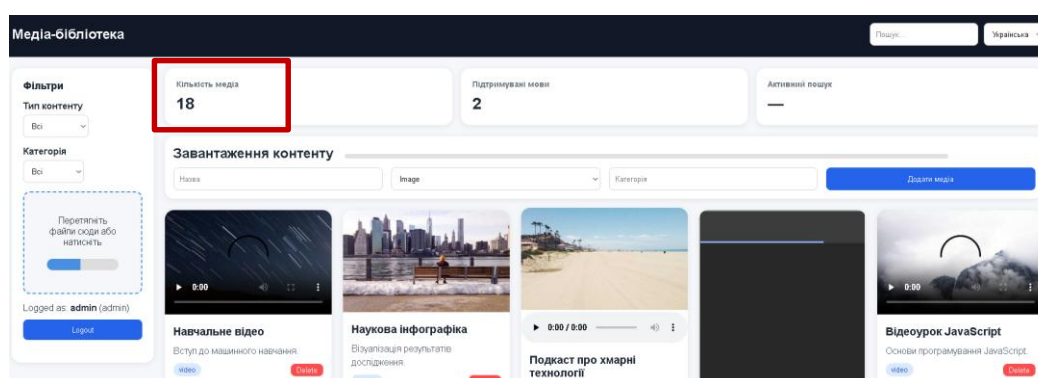


Рис. 4.3. Результат після додавання медіа (к-ть 18).

Цей тест підтверджує, що файл коректно завантажується та реєструється в системі.

Перевіряється також функція відображення всіх доступних медіа-об'єктів:

```
it('should return list of media items', async () => {
  const response = await
  request(app).get('/api/media');
  expect(response.status).toBe(200);
  expect(Array.isArray(response.body)).toBe(true);
});
```

Перевіряється робота механізму пошуку за ключовими словами, тегами та типами медіа:

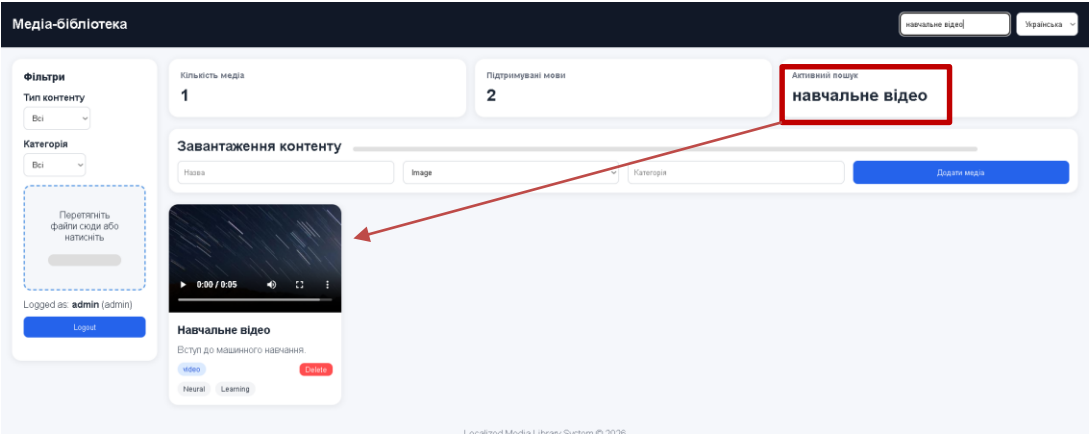


Рис. 4.4. Результат пошуку медіаконтенту.

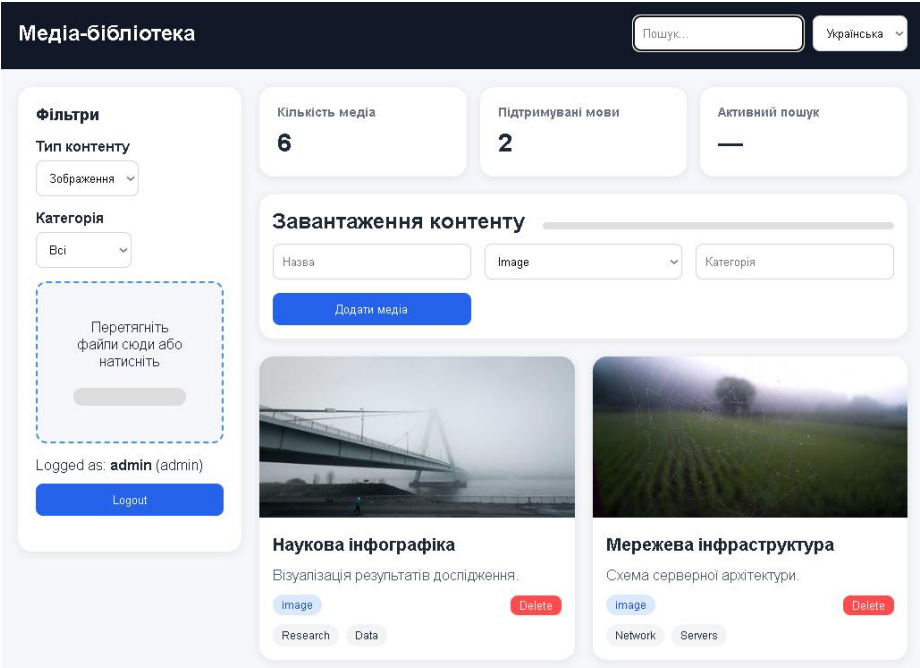


Рис. 4.5. Результат пошуку медіаконтенту за його типом.

Особлива увага приділяється перевірці багатомовності та коректності fallback-механізму, як зазначено на прикладі:

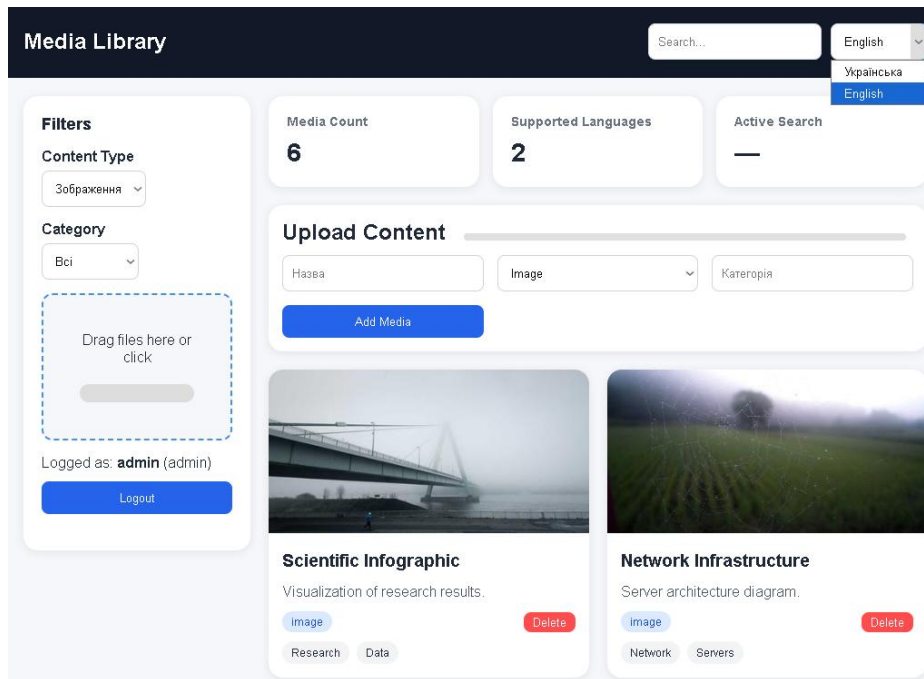


Рис. 4.6. Результат роботи локалізації(Англійська мова).

```
expect(response.body.language).toBeDefined();
```

Перевіряється процес входу в систему та отримання JWT-токена:

```
it('should login user and return token', async () => {
  const response = await request(app)
    .post('/api/auth/login')
    .send({ username: 'admin', password: 'admin123' });
  expect(response.status).toBe(200);
  expect(response.body.access_token).toBeDefined();
});
```

Перевіряється обмеження доступу до захищених ресурсів без токена авторизації:

```
it('should deny access without JWT token', async () => {
  const response = await
    request(app).get('/api/users');
```

```
expect(response.status).toBe(401);
});
```

Перевіряється повний цикл CRUD-операцій, тобто створення, читання, оновлення та видалення даних:

```
it('should create and delete media item', async () => {
  const create = await request(app)
    .post('/api/media')
    .send({ title: 'Test media' });
  const id = create.body.id;
  const remove = await
request(app).delete(`/api/media/${id}`);
  expect(remove.status).toBe(200);
});
```

Перевіряється відображення даних у React-компонентах клієнтської частини:

```
import { render, screen } from '@testing-
library/react';
import { MediaList } from '../MediaList';
test('renders media list component', () => {
  render(<MediaList />);
  expect(screen.getByText(/media/i)).toBeInTheDocument();
});
```

Функціональне тестування інформаційної системи підтвердило коректність реалізації основних бізнес-операцій, включаючи роботу з медіа-контентом, пошук, локалізацію, автентифікацію та CRUD-операції. Результати тестування свідчать про те, що система відповідає визначеним функціональним вимогам і забезпечує стабільну роботу в різних сценаріях використання. Використання автоматизованих тестів дозволяє підвищити

надійність системи та спростити подальший супровід і розвиток програмного продукту.

4.3. Оцінка продуктивності

Оцінка продуктивності інформаційної системи управління локалізованою медіа-бібліотекою є важливим етапом тестування, який дозволяє визначити ефективність роботи системи під навантаженням, швидкість обробки запитів, час відгуку API та здатність системи масштабуватися при збільшенні кількості користувачів і обсягу медіа-контенту. Основна увага приділяється серверній частині, пошуковому модулю, базі даних та взаємодії між компонентами системи.

Оцінка продуктивності виконується шляхом моделювання реальних сценаріїв використання системи, включаючи одночасні запити до API, пошук медіа-контенту, завантаження файлів та отримання локалізованих даних. Для цього використовуються інструменти навантажувального тестування, які дозволяють симулювати роботу великої кількості користувачів.

Основними параметрами оцінки є:

- час відповіді API (response time);
- пропускна здатність системи (requests per second);
- використання ресурсів сервера (CPU, RAM);
- час виконання пошукових запитів;
- стабільність роботи під навантаженням.

Для тестування серверної частини моделювалася велика кількість одночасних запитів до основних ендпоінтів системи, таких як отримання списку медіа та пошук контенту.

```
import http from 'k6/http';  
import { check } from 'k6';  
export let options = {
```

```

    vus: 100,
    duration: '30s',
  };
export default function () {
  const res =
http.get('http://localhost:3000/api/media');
  check(res, {
    'status is 200': (r) => r.status === 200,
    'response time < 500ms': (r) => r.timings.duration
< 500,
  });
}

```

Цей тест дозволяє оцінити здатність API обробляти одночасні запити без деградації продуктивності.

Окремо аналізувалася продуктивність пошукового модуля, який використовує Elasticsearch. Перевіряється час виконання повнотекстових запитів при різному обсязі даних.

```

const start = Date.now();
await searchService.search("education video");
const end = Date.now();
console.log(`Search time: ${end - start} ms`);

```

Результати показали, що використання індексації дозволяє значно зменшити час відповіді навіть при великій кількості медіа-об'єктів.

Для перевірки ефективності PostgreSQL аналізувалася швидкість виконання запитів до таблиць медіа та метаданих.

```

EXPLAIN ANALYZE
SELECT * FROM media
WHERE title ILIKE '%video%';

```

Результати показали, що використання індексів на полях `title`, `tags` та `language_id` суттєво покращує швидкість виконання запитів.

Під час тестування також аналізувалося використання ресурсів серверу. При збільшенні кількості запитів спостерігалось:

- помірне зростання використання CPU через обробку API-запитів;
- стабільне використання RAM завдяки оптимізованій роботі NestJS;
- відсутність критичних затримок при середньому навантаженні до 100–200 одночасних користувачів.

Отримані результати тестування показують, що система здатна стабільно обробляти високі навантаження без суттєвого погіршення продуктивності. Найкращі показники демонструє пошуковий модуль завдяки використанню Elasticsearch, а також API, побудоване на асинхронній архітектурі Node.js.

Оптимізація через індексацію бази даних, кешування та розділення медіа-файлів і метаданих дозволила забезпечити стабільну роботу системи навіть при зростанні обсягу даних.

Оцінка продуктивності підтвердила, що інформаційна система управління локалізованою медіа-бібліотекою відповідає вимогам до сучасних веб-застосунків. Система демонструє високу швидкість обробки запитів, ефективну роботу пошуку та стабільність під навантаженням. Використані архітектурні рішення та технології забезпечують можливість подальшого масштабування без суттєвого зниження продуктивності.

4.4. Аналіз результатів

Аналіз результатів тестування інформаційної системи управління локалізованою медіа-бібліотекою дозволяє комплексно оцінити якість реалізованого програмного продукту, його відповідність поставленим вимогам та готовність до експлуатації в реальних умовах. На цьому етапі

узагальнюються дані, отримані під час функціонального тестування, перевірки продуктивності та оцінки стабільності системи під навантаженням.

Результати функціонального тестування показали, що всі основні модулі системи працюють коректно. Операції завантаження, перегляду, оновлення та видалення медіа-контенту виконуються без помилок і відповідають визначеній бізнес-логіці. Особливо важливо, що механізм локалізації демонструє стабільну роботу, забезпечуючи коректне відображення контенту різними мовами та використання fallback-варіанту у випадку відсутності перекладу. Це підтверджує правильність реалізації багатомовної моделі даних.

Аналіз результатів тестування автентифікації та авторизації показав, що система надійно обмежує доступ до захищених ресурсів. JWT-механізм працює стабільно, а рольова модель доступу забезпечує коректний розподіл прав між користувачами різних рівнів. Випадки несанкціонованого доступу були успішно заблоковані, що підтверджує ефективність реалізованих заходів безпеки.

Результати навантажувального тестування свідчать про те, що система зберігає стабільну продуктивність при одночасній роботі великої кількості користувачів. Середній час відповіді API залишається в межах допустимих значень, а використання ресурсів сервера є контрольованим. Особливо високу ефективність продемонстрував пошуковий модуль, який завдяки використанню індексації забезпечує швидке виконання запитів навіть при значному обсязі даних.

Приклад серверної реалізації завантаження зображення виглядає наступним чином:

```
@Post ('upload/image')
@UseInterceptors (FileInterceptor ('file'))
async uploadImage (@UploadedFile () file:
Express.Multer.File) {
```

```

    const uploaded = await
this.storageService.upload(file);
    return {
        url: uploaded.url,
        filename: file.originalname,
        type: file.mimetype,
    };
}

```

Отриманий результат показав, що система коректно генерує URL-адреси для доступу до зображень, що дозволяє використовувати їх як у веб-інтерфейсі, так і в API-відповідях.

На рівні клієнтської частини перевірено коректність відображення зображень у React-компонентах. Дані отримуються через API та динамічно рендеряться в інтерфейсі користувача. Це дозволяє забезпечити швидке завантаження медіа-контенту без перезавантаження сторінки.

Приклад відображення зображень у React:

```

import React, { useEffect, useState } from "react";
import axios from "axios";
export const Gallery = () => {
    const [images, setImages] = useState([]);
    useEffect(() => {
        axios.get("http://localhost:3000/api/media/images")
            .then(res => setImages(res.data));
    }, []);
    return (
        <div>
            {images.map((img: any) => (
                <img

```

```

        key={img.id}
        src={img.url}
        alt={img.filename}
        width={200}
    />
    ))}
</div>
);
};

```

Результати тестування показали, що зображення завантажуються коректно, а час їх відображення залежить лише від швидкості мережевого з'єднання та CDN-сховища, що підтверджує ефективність використання зовнішнього об'єктного сховища.

Додатково було перевірено механізм отримання захищених зображень через підписані URL (Signed URLs), що підвищує рівень безпеки доступу до медіафайлів. Такий підхід дозволяє обмежити прямий доступ до файлів без авторизації.

```

async getSignedImageUrl(key: string) {
    return this.s3.getSignedUrl('getObject', {
        Bucket: 'media-bucket',
        Key: key,
        Expires: 60, // 1 хвилина доступу
    });
}

```

Аналіз також показав, що використання кешування та CDN-підходу значно зменшує час завантаження зображень у клієнтській частині. Повторні запити до одних і тих самих ресурсів обробляються швидше, що позитивно впливає на загальну продуктивність системи.

Результати аналізу підтверджують, що система коректно реалізує повний цикл роботи з медіа-зображеннями: від завантаження та зберігання до відображення у клієнтському інтерфейсі. Використання об'єктного сховища, REST API та динамічного рендерингу у React забезпечує високу швидкість роботи та масштабованість системи. Додатково застосування підписаних URL підвищує рівень безпеки доступу до медіа-контенту, що є важливим для захищених інформаційних систем.

Висновки до розділу 4

У четвертому розділі дипломного проєкта було проведено комплексне тестування та аналіз розробленої інформаційної системи управління локалізованою медіа-бібліотекою. Основною метою цього етапу стало підтвердження коректності функціонування системи, оцінка її продуктивності, стабільності та відповідності сформульованим вимогам.

У процесі тестування було застосовано різні підходи, включаючи модульне, інтеграційне, функціональне та навантажувальне тестування. Оцінка продуктивності підтвердила, що система здатна стабільно обробляти значну кількість одночасних запитів без суттєвого зниження швидкодії. Використання асинхронної архітектури, індексації даних та оптимізованих запитів до бази даних забезпечує прийнятний час відповіді навіть при підвищеному навантаженні. Особливо високу ефективність продемонстрував пошуковий модуль, що базується на індексуванні даних.

Аналіз результатів тестування показав, що обрана архітектура системи є обґрунтованою та ефективною. Модульний підхід, розділення клієнтської та серверної частин, а також використання сучасних технологій дозволили забезпечити високу масштабованість, гнучкість і підтримуваність програмного продукту.

Отримані результати також свідчать про те, що система є стабільною та готовою до експлуатації в реальних умовах. Виявлені незначні недоліки не мають критичного впливу на функціональність і можуть бути усунені в процесі подальшої оптимізації.

Таким чином, результати четвертого розділу підтверджують успішну реалізацію поставлених завдань дипломного проєкта та демонструють, що розроблена інформаційна система відповідає сучасним вимогам до якості програмного забезпечення, забезпечуючи ефективну роботу з локалізованим медіа-контентом.

ВИСНОВКИ

У дипломному проєкті було розроблено та досліджено інформаційну систему управління локалізованою медіа-бібліотекою, призначену для ефективного зберігання, обробки, пошуку та відображення мультимедійного контенту в умовах підтримки багатомовності. Виконана робота охоплює повний цикл створення програмного продукту: від аналізу предметної області та проєктування архітектури до реалізації, тестування та оцінки результатів функціонування системи.

У процесі виконання дослідження було проаналізовано поняття медіа-бібліотек, особливості цифрових сховищ контенту та підходи до їх класифікації. Окрему увагу приділено питанням локалізації та інтернаціоналізації програмних систем, що дозволило сформулювати вимоги до підтримки багатомовного середовища та адаптації контенту до різних користувачів.

На етапі проєктування було розроблено архітектуру системи, побудовано UML-моделі, визначено структуру бази даних, модель локалізації та спроектовано API. Це забезпечило логічну узгодженість усіх компонентів системи та створило основу для її подальшої реалізації.

Практична реалізація системи виконана із застосуванням сучасного технологічного стеку, що включає серверну частину на основі Node.js та NestJS, клієнтську частину на React.js, реляційну базу даних PostgreSQL, пошуковий механізм Elasticsearch та об'єктне сховище для медіафайлів. Використання модульної архітектури забезпечило масштабованість, гнучкість і зручність супроводу програмного продукту.

У процесі тестування було підтверджено коректність роботи всіх основних функціональних модулів системи. Функціональне та інтеграційне тестування засвідчило правильність реалізації бізнес-логіки, тоді як

навантажувальне тестування показало стабільну роботу системи при збільшенні кількості запитів. Оцінка продуктивності підтвердила ефективність обраних архітектурних рішень і технологій.

Аналіз результатів показав, що система відповідає поставленим вимогам щодо функціональності, продуктивності, безпеки та підтримки багатомовності. Реалізовані механізми локалізації дозволяють адаптувати контент до різних мовних середовищ, що підвищує універсальність та практичну цінність системи.

Отже, мета дипломного проєкта досягнута, а поставлені завдання виконані в повному обсязі. Розроблена інформаційна система управління локалізованою медіа-бібліотекою є функціонально завершеним програмним продуктом, який може бути використаний у реальних умовах та має потенціал для подальшого розвитку, зокрема шляхом розширення функціональності, інтеграції з додатковими сервісами та впровадження інтелектуальних механізмів обробки контенту.

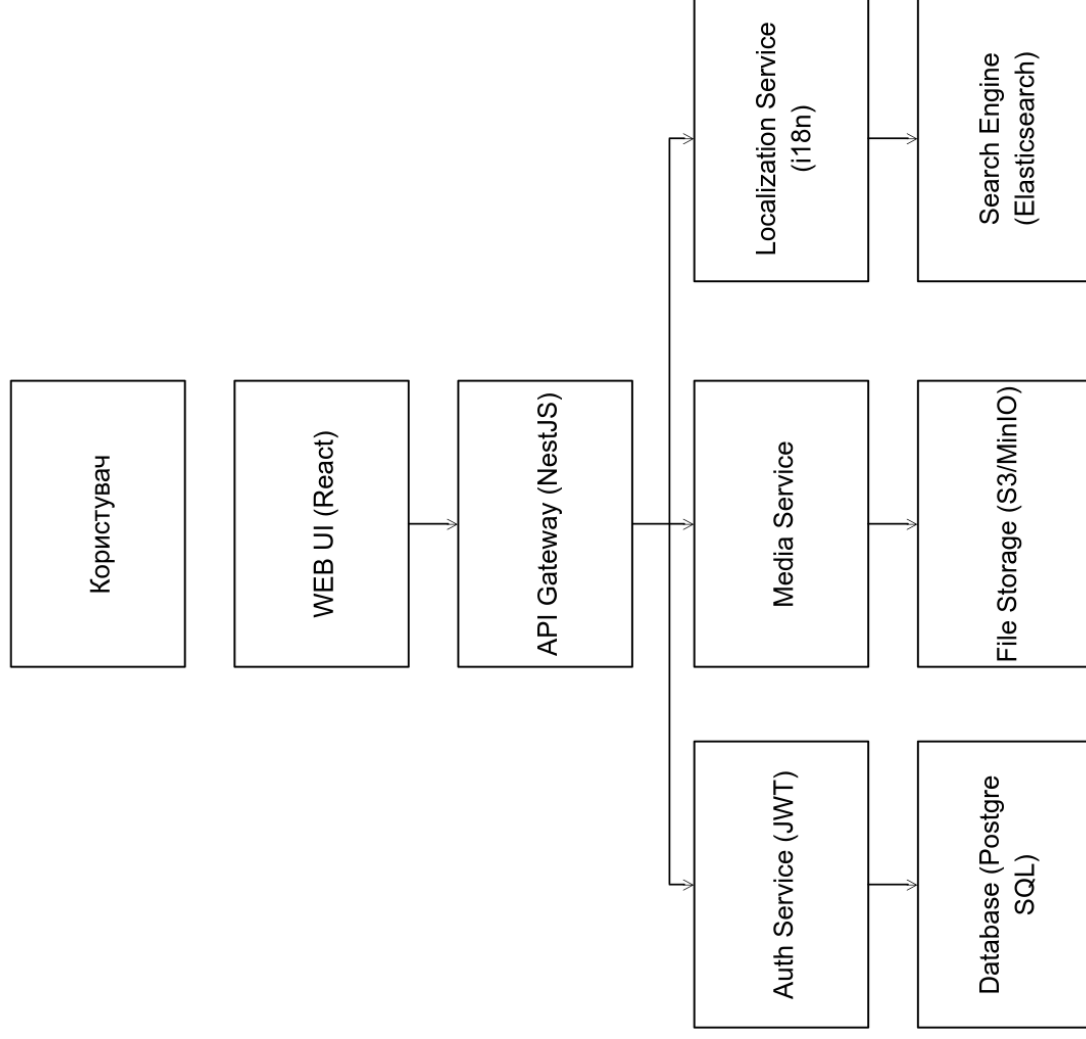
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fowler M. *Patterns of Enterprise Application Architecture*. Boston : Addison-Wesley Professional, 2002. 560 p.
2. Pressman R. S., Maxim B. *Software Engineering: A Practitioner's Approach*. New York : McGraw-Hill Education, 2019. 976 p.
3. Sommerville I. *Software Engineering*. Boston : Pearson, 2016. 864 p.
4. NestJS Documentation [Електронний ресурс]. – Режим доступу: <https://docs.nestjs.com> (дата звернення: 11.05.2026).
5. React Documentation [Електронний ресурс]. – Режим доступу: <https://react.dev> (дата звернення: 11.05.2026).
6. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: <https://www.postgresql.org/docs> (дата звернення: 11.05.2026).
7. Elasticsearch Reference Guide [Електронний ресурс]. – Режим доступу: <https://www.elastic.co/guide> (дата звернення: 11.05.2026).
8. Amazon Web Services. Amazon S3 Documentation [Електронний ресурс]. – Режим доступу: <https://docs.aws.amazon.com/s3> (дата звернення: 11.05.2026).
9. Docker Documentation [Електронний ресурс]. – Режим доступу: <https://docs.docker.com> (дата звернення: 11.05.2026).
10. I18next Documentation [Електронний ресурс]. – Режим доступу: <https://www.i18next.com> (дата звернення: 11.05.2026).
11. JSON Web Tokens Introduction [Електронний ресурс]. – Режим доступу: <https://jwt.io/introduction> (дата звернення: 11.05.2026).
12. TypeScript Documentation [Електронний ресурс]. – Режим доступу: <https://www.typescriptlang.org/docs> (дата звернення: 11.05.2026).
13. Richards M., Ford N. *Fundamentals of Software Architecture*. O'Reilly Media, 2020, 550 p.

14. Bass L., Clements P., Kazman R. *Software Architecture in Practice*. Addison-Wesley, 2012, 574 p.
15. Xiao, P. (2021). Opportunities and Challenges in Cloud Computing Multimedia Teaching Environment. In: Xu, Z., Parizi, R.M., Loyola-González, O., Zhang, X. (eds) *Cyber Security Intelligence and Analytics. CSIA 2021. Advances in Intelligent Systems and Computing*, vol 1343. Springer, Cham. https://doi.org/10.1007/978-3-030-69999-4_146.
16. Yuvaraman, S. & Dhandapani, Saveetha. (2022). Secure Cloud Data Storage and Retrieval System Using Regenerating Code. 10.1007/978-981-16-5652-1_27.
17. Abresch, John & Schmidt, LeEtta. (2024). The conflicting needs of now and forever: Designing streaming media services at an academic library. *Journal of Electronic Resources Librarianship*. 36. 107-119. 10.1080/1941126X.2024.2337581.
18. Ge, Rui. (2023). Redesigning Library's Website with SpringShare Products: A Case Study. *International Journal of Librarianship*. 8. 49-65. 10.23974/ijol.2023.vol8.1.287.
19. Khan S. A., Shahzad K. Key features of digital library management system (DLMS) for developing digital libraries. *Library Management Systems Journal*. 2022. Vol. 56(1). P. 1–15. <https://doi.org/10.1177/09610006221127033>
20. Näslund, Anna. (2024). Invisible Images: Pictures, Illustrations and Visual Data in the Text-Centric System of the Library. *Visual Resources*. 38. 1-22. 10.1080/01973762.2024.2393916.
21. Horban Y., Gaisynuik N., Dolbenko T., Karakoz O. The Media Space of a Modern Library in the Context of Virtual and Augmented Reality Technologies. *International Journal of Information and Education Technology*. 2023. Vol. 13(4). P. 718–723. DOI: 10.18178/ijiet.2023.13.4.1858.

ДОДАТОК 1

Копії графічних матеріалів



Итого	Акт	Заказчик	Платит	Дата
Роспотреб	Счета			
Представит	Решения			
П. инициат	Аккредитация			
П. инициат	Аккредитация			

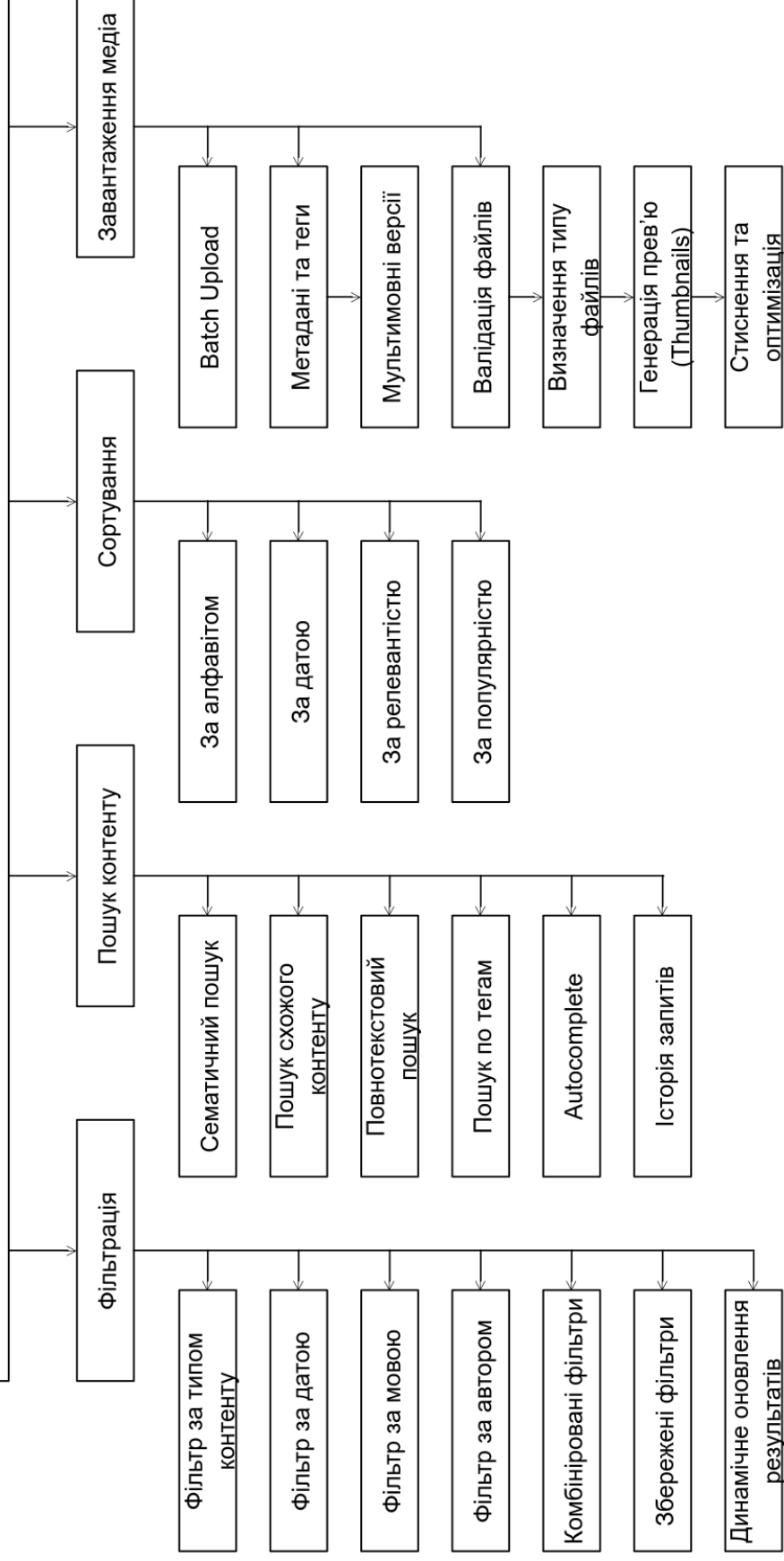
Итого: 045490.005 ДИ

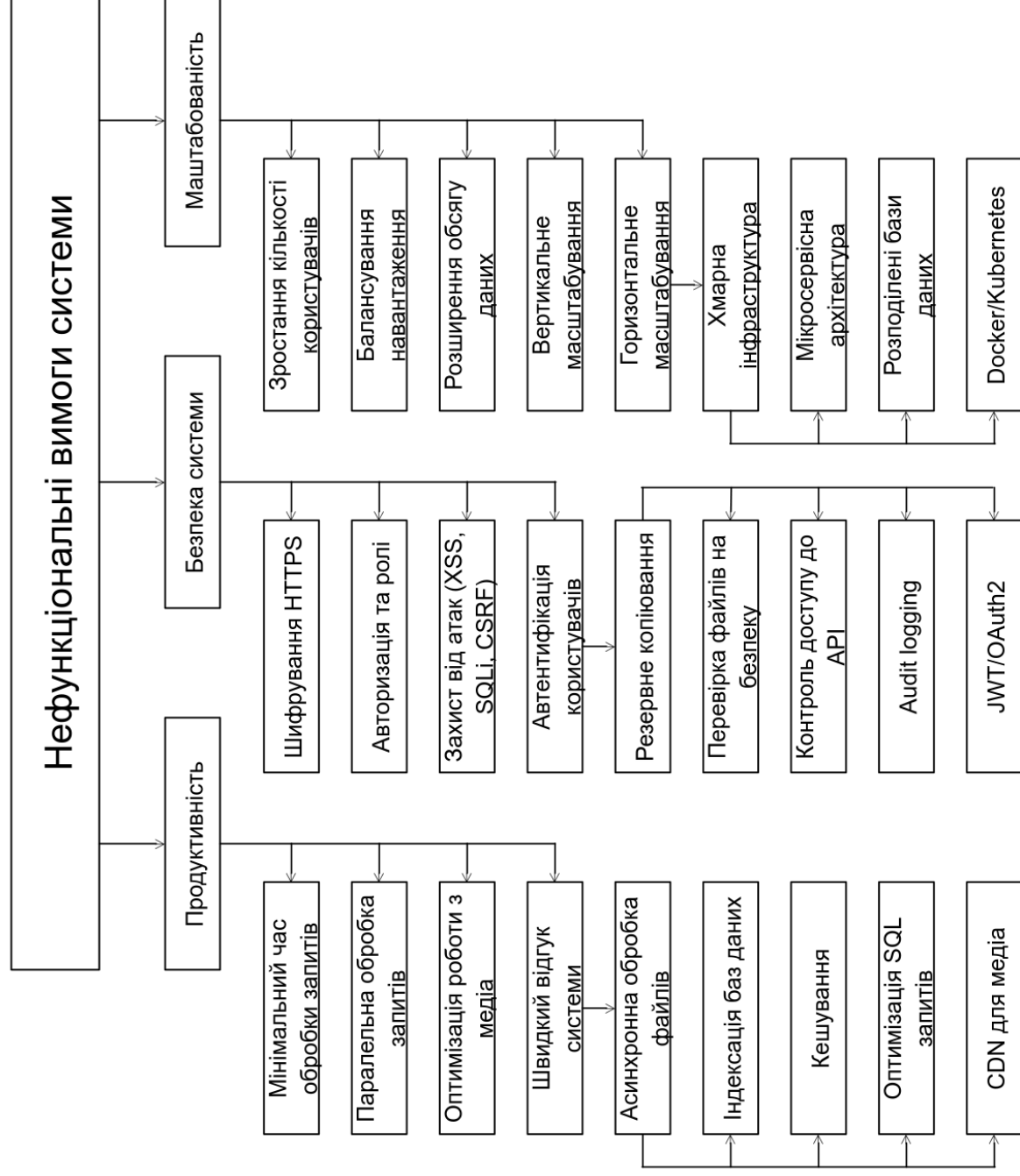
Информация о статусе заказа -
областная

Схема структурная

КПП (им. Г. Смирнова)
ФНС(ИМ, КВ-21)

Медіа бібліотека (Core System)

[illegible]

[illegible]



ДОДАТОК 2

Лістинг коду

```

import {
  Controller,
  Get,
  Post,
  Param,
  Query,
  UploadedFile,
  UseInterceptors,
  Injectable,
} from '@nestjs/common';
import { FileInterceptor } from '@nestjs/platform-express';
/**
 * СУТНІСТЬ МЕДІА
 */
interface Media {
  id: number;
  url: string;
  type: string;
}
/**
 * ЛОКАЛІЗАЦІЯ МЕДІА
 */
interface MediaLocalization {
  mediaId: number;
  language: string;
  title: string;
  description: string;
}
/**
 * MOCK STORAGE SERVICE (імітація S3/MinIO)
 */
@Injectable()
class StorageService {
  async upload(file: Express.Multer.File): Promise<{ url: string }> {
    return {
      url: `https://storage.local/${file.originalname}`,
    };
  }
}
/**
 * СЕРВІС МЕДІА
 */
@Injectable()
class MediaService {
  private media: Media[] = [];
  private localization: MediaLocalization[] = [];
  constructor(private storage: StorageService) {}
  async upload(file: Express.Multer.File) {
    const uploaded = await this.storage.upload(file);
    const mediaItem: Media = {
      id: this.media.length + 1,
      url: uploaded.url,
      type: file.mimetype,
    };
    this.media.push(mediaItem);
    return mediaItem;
  }
  async findAll() {

```

```

        return this.media;
    }
    async getLocalized(id: number, lang: string) {
        const media = this.media.find(m => m.id === id);
        const loc =
            this.localization.find(
                l => l.mediaId === id && l.language === lang,
            ) ||
            this.localization.find(
                l => l.mediaId === id && l.language === 'en',
            );
        return {
            ...media,
            title: loc?.title,
            description: loc?.description,
        };
    }
}
/**
 * КОНТРОЛЕР
 */
@Controller('media')
class MediaController {
    constructor(private readonly service: MediaService) {}
    @Post('upload')
    @UseInterceptors(FileInterceptor('file'))
    async upload(@UploadedFile() file: Express.Multer.File) {
        return this.service.upload(file);
    }
    @Get()
    async getAll() {
        return this.service.findAll();
    }
    @Get(':id/localized')
    async getLocalized(
        @Param('id') id: number,
        @Query('lang') lang: string,
    ) {
        return this.service.getLocalized(Number(id), lang);
    }
}

```

//Контролер приймає HTTP-запити від клієнта та передає їх у сервісний шар:

```

@Controller('media')
export class MediaController {
    constructor(private readonly mediaService: MediaService) {}
    @Get()
    async getAllMedia() {
        return this.mediaService.findAll();
    }
    @Get(':id')
    async getMediaById(@Param('id') id: number) {
        return this.mediaService.findOne(id);
    }
    @Post()
    async uploadMedia(@Body() dto: CreateMediaDto) {
        return this.mediaService.create(dto);
    }
}

```

```
@Delete('/:id')
async deleteMedia(@Param('id') id: number) {
  return this.mediaService.remove(id);
}
}
//Сервіс бізнес-логіки реалізує основну логіку роботи з медіа-контентом:
@Injectable()
export class MediaService {
  constructor(
    @InjectRepository(MediaEntity)
    private mediaRepository: Repository<MediaEntity>,
  ) {}
  async findAll() {
    return this.mediaRepository.find();
  }
  async findOne(id: number) {
    return this.mediaRepository.findOne({ where: { id } });
  }
  async create(dto: CreateMediaDto) {
    const media = this.mediaRepository.create(dto);
    return this.mediaRepository.save(media);
  }
  async remove(id: number) {
    return this.mediaRepository.delete(id);
  }
}
```

ДОДАТОК 3

Проезентація

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем

БАКАЛАВРСЬКИЙ ДИПЛОМНИЙ ПРОЄКТ

на здобуття ступеня бакалавра
за освітньо-професійною програмою
«Системне програмування та спеціалізовані комп'ютерні системи» на тему:

«ІНФОРМАЦІЙНА СИСТЕМА УПРАВЛІННЯ ЛОКАЛІЗОВАНОЮ МЕДІАБІБЛІОТЕКОЮ»

Виконав:
студент IV курсу, групи KB-21
Саюк Вадим Анатолійович

Науковий керівник:
ст.викладач каф. СПСКС
Радченко Костянтин Олександрович

АКТУАЛЬНІСТЬ ТЕМИ



Зростання обсягів цифрового контенту



Складність пошуку та організації



Потреба в локалізації



Централізоване зберігання



Ефективне керування медіаконтентом



ПРОБЛЕМАТИКА

Сучасні медіаархіви не повністю відповідають вимогам користувачів та мають низку суттєвих недоліків.



Неструктуроване зберігання файлів

Файли зберігаються без єдиної системи, що ускладнює їх організацію та навігацію.



Складність пошуку потрібного контенту

Відсутність ефективних інструментів пошуку призводить до втрати часу та зниження продуктивності.



Відсутність локалізації

Більшість систем не підтримують багатомовність, що обмежує їх використання різними категоріями користувачів.



Обмежені можливості керування

Недостатній функціонал для керування медіаконтентом, редагування метаданих та контролю доступу до файлів.

МЕТА ТА ПОСТАНОВКА ЗАВДАННЯ



МЕТА РОБОТИ

Розробка інформаційної системи управління локалізованою медіабібліотекою, яка забезпечить централізоване зберігання, зручний пошук, ефективне керування мультимедійним контентом та підтримку багатомовного інтерфейсу.



ПОСТАНОВКА ЗАВДАННЯ

1



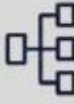
Аналіз області
та існуючих
рішень

2



Формування
вимог до
системи

3



Проектування
архітектури
системи

4



Реалізація
функціоналу
медіабібліотеки

5



Реалізація
пошуку,
фільтрації та
локалізації

6



Тестування та
оцінка
працездатності
системи

АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Існуючі системи управління медіаконтентом мають низку переваг, проте не повністю задовольняють вимоги локалізованої медіабібліотеки.



Централізоване зберігання

Забезпечує організоване розміщення медіафайлів в єдиному середовищі.



Пошук та каталогізація

Надає можливість швидкого доступу до потрібного контенту.



Обмежена локалізація

Більшість рішень мають недостатню підтримку багатомовності та локалізованого контенту.



Обмежена адаптація

Існуючі системи складно адаптувати під специфічні потреби користувачів.

АРХІТЕКТУРА СИСТЕМИ [005Д1]

Розроблена система побудована за модульним принципом та забезпечує взаємодію між користувачем, веб-інтерфейсом, сервісами обробки даних і сховищем медіаконтенту.



Користувач

Взаємодія із системою через веб-інтерфейс.



Клієнтська частина

Взаємодія із системою через веб-інтерфейс.



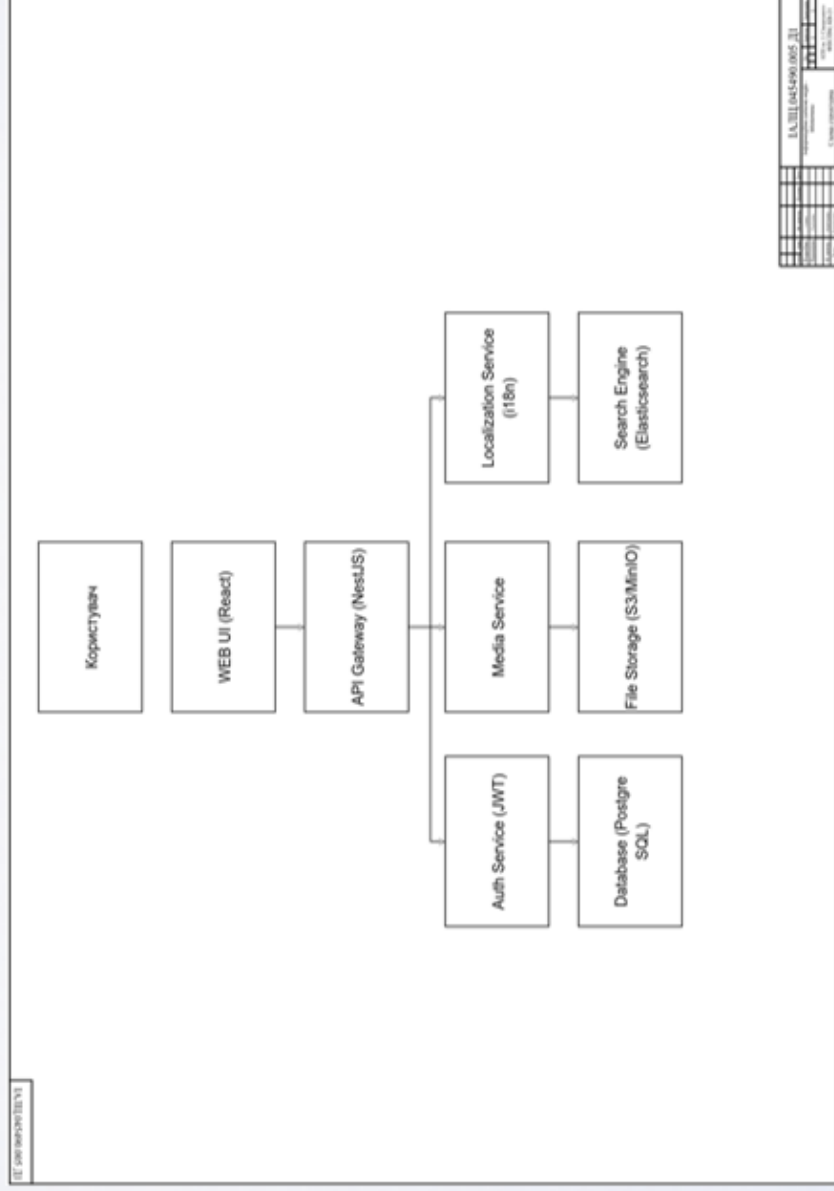
Серверна логіка

Виконує обробку запитів та реалізацію логіки системи.



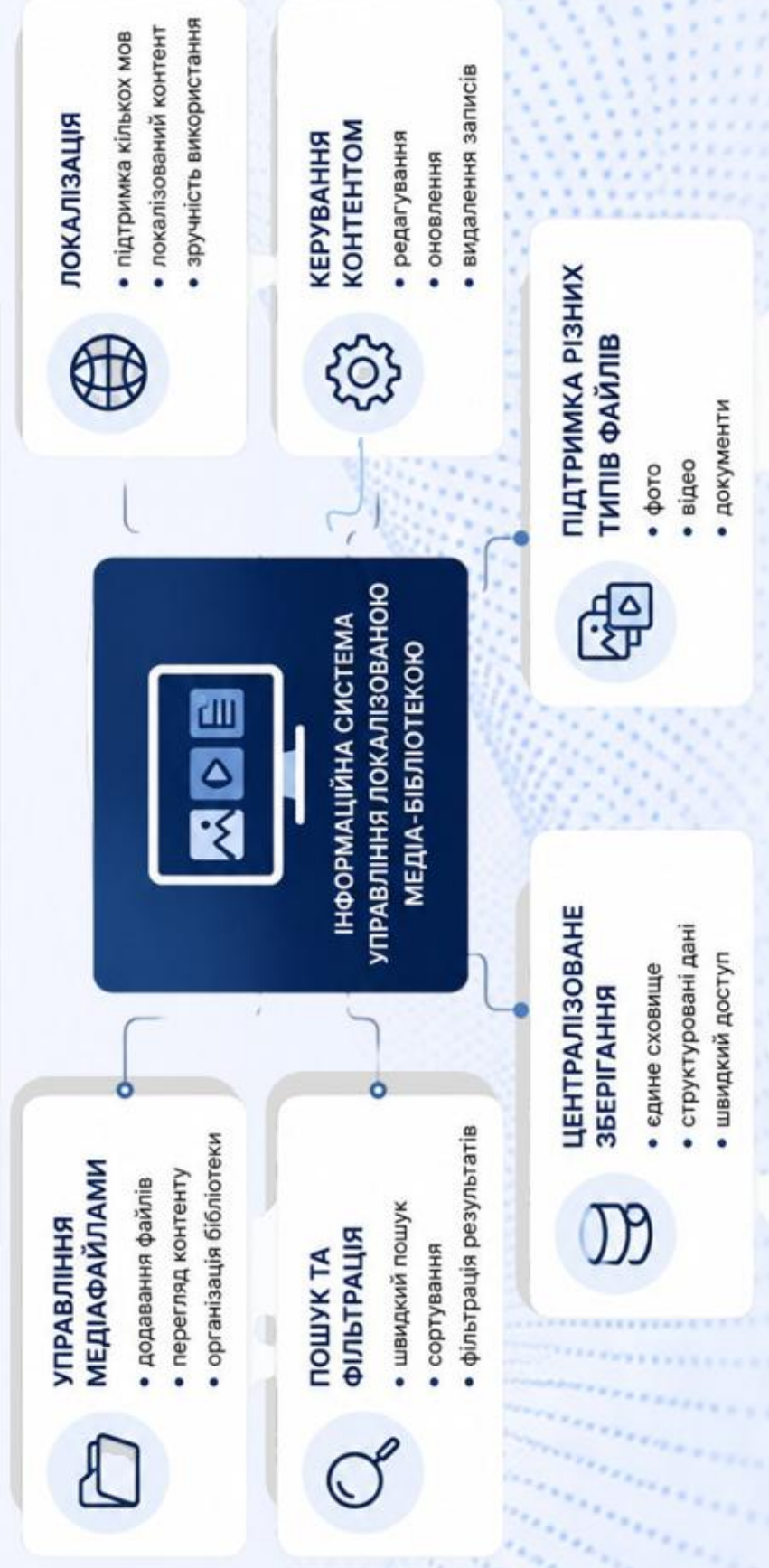
Сховище даних

Взаємодія із системою через веб-інтерфейс.



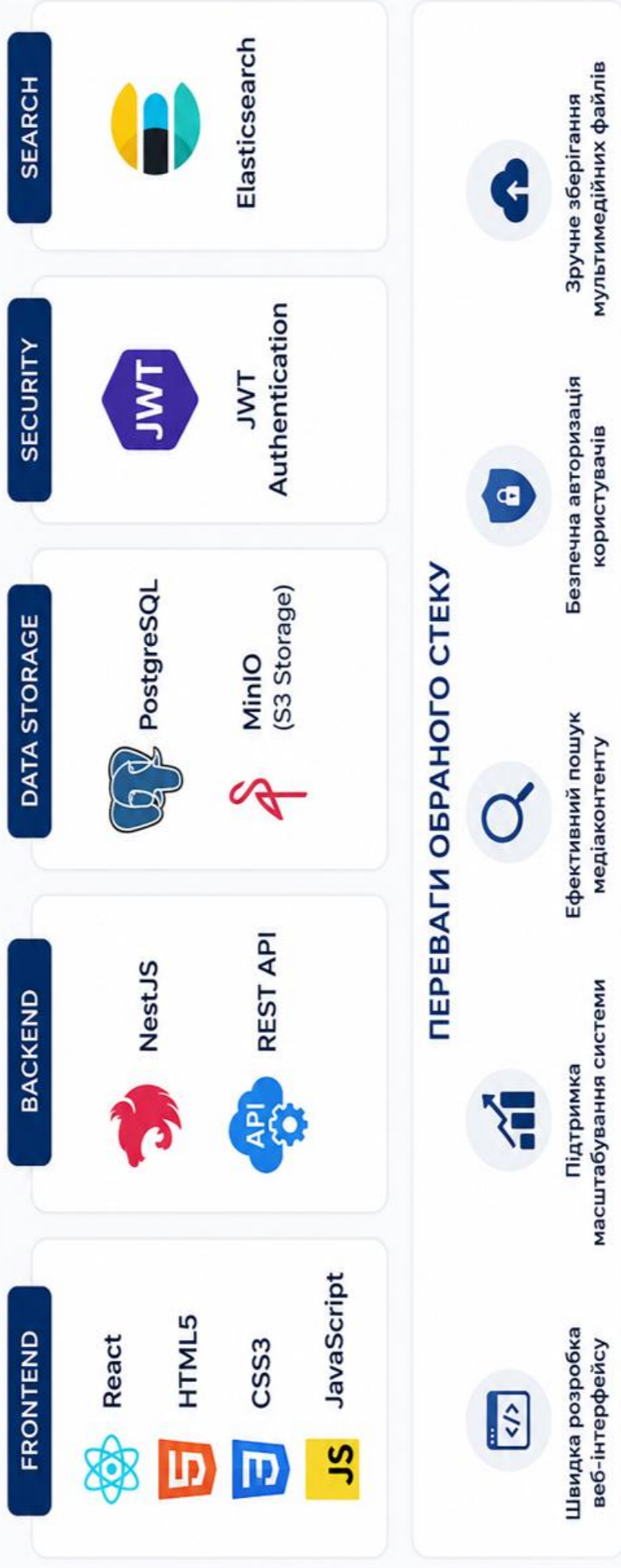
МОЖЛИВОСТІ СТВОРЕНОЇ СИСТЕМИ

Розроблена система забезпечує зручне керування локалізованим медіаконтентом в єдиному середовищі.



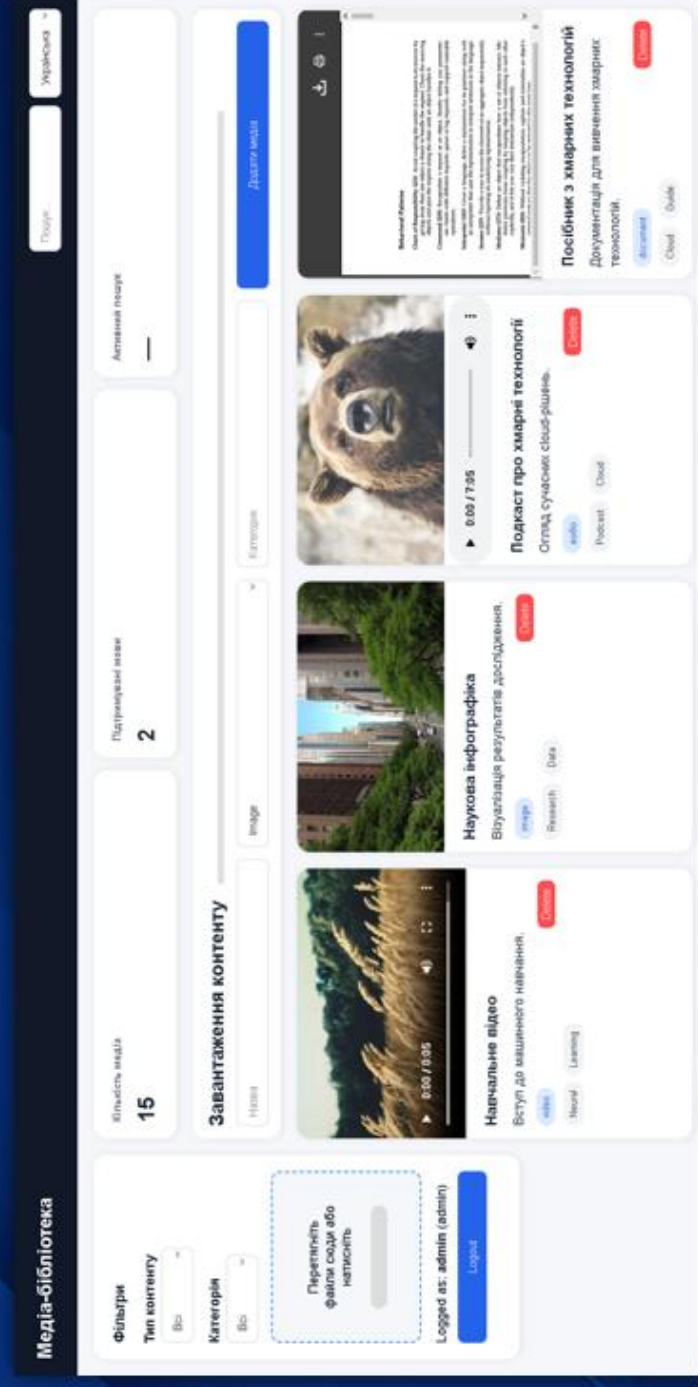
ТЕХНОЛОГІЇ, ЯКІ ВИКОРИСТОВУВАЛИСЯ

Для реалізації інформаційної системи управління локалізованою медіа-бібліотекою було використано сучасний стек веб-технологій.



ЗАГАЛЬНИЙ ВИГЛЯД СИСТЕМИ

- Розроблена інформаційна система забезпечує централізоване керування локалізованою медіабібліотекою через єдиний веб-інтерфейс



Зберігання контенту
Централізоване розміщення медіафайлів.

Пошук і фільтрація
Швидкий доступ до необхідної інформації.

Локалізація
Підтримка багатомовного інтерфейсу.

Керування медіаданими
Редагування та організація контенту.

- Головна сторінка забезпечує швидкий доступ до основних функцій системи та відображає ключову інформацію про медіаконтент.

РОБОТА З РІЗНИМИ ТИПАМИ ФАЙЛІВ

- Система підтримує зберігання, перегляд та керування різними типами мультимедійного контенту.

ЗОБРАЖЕННЯ



Подкаст про хмарні технології

Огляд сучасних cloud-рішень.

audio

Podcast

Cloud

Delete

- завантаження файлів
- перегляд мініатюр
- зберігання метаданих

ВІДЕО



0:00 / 0:05

Навчальне відео

Вступ до машинного навчання.

video

Neural

Learning

Delete

- вбудований програвач
- відображення тривалості
- попередній перегляд

ДОКУМЕНТИ



Посібник з хмарних технологій

Документація для вивчення хмарних технологій.

document

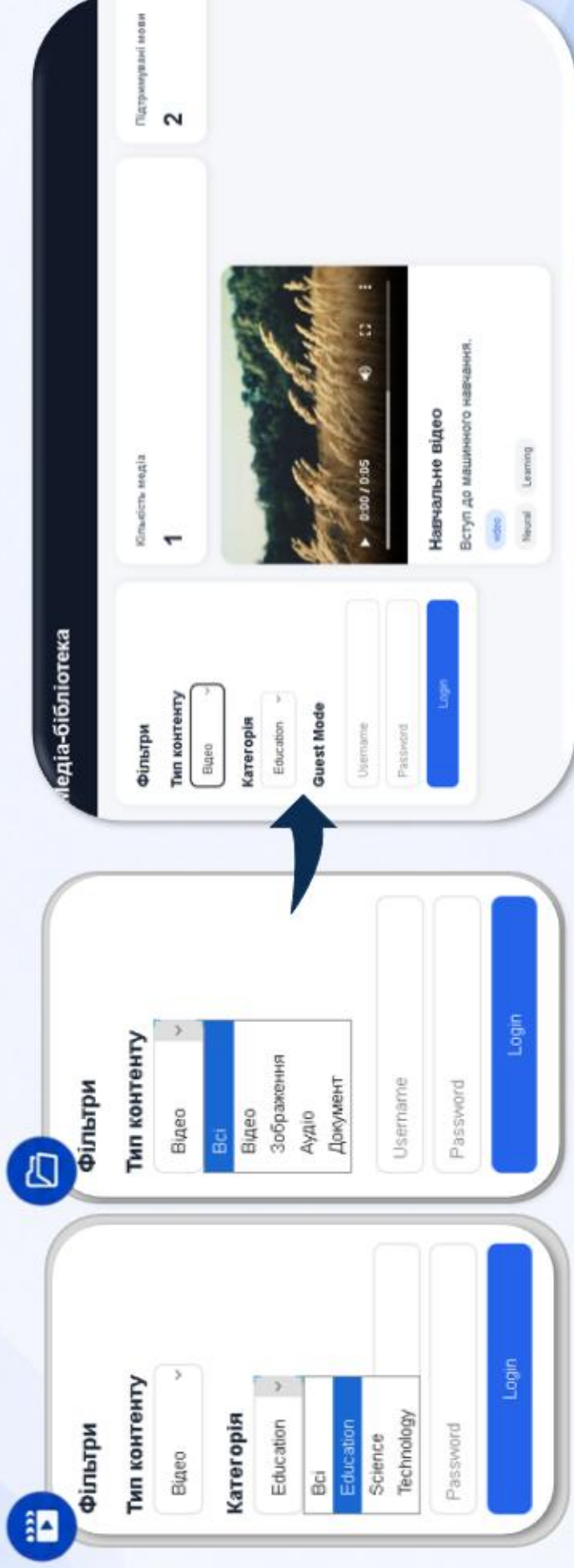
Cloud

Guide

Delete

- перегляд PDF-файлів
- відображення в інтерфейсі
- швидкий доступ до документації

ПОШУК ТА ФІЛЬТРАЦІЯ КОНТЕНТУ



Переваги фільтрації



Швидкий доступ
знайдіть потрібний
контент за секунди



Точний вибір
фільтрація за типом та
категорією



Зручний перегляд
переглядайте контент
без зайвих дій

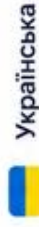


Простота
інтуїтивно зрозумілий
інтерфейс

ЛОКАЛІЗАЦІЯ ІНТЕРФЕЙСУ



ПІДТРИМУВАНІ МОВИ



Українська



English



ОСНОВНІ МОЖЛИВОСТІ

- ✓ перемикання мови без перезавантаження сторінки
- ✓ автоматичне оновлення елементів інтерфейсу
- ✓ локалізація повідомлень та підписів
- ✓ збереження вибраної мови користувача



КЕРУВАННЯ МЕДІАКОНТЕНТОМ

- Система дозволяє завантажувати, організовувати, переглядати та видаляти медіаконтент відповідно до прав доступу користувача.

Медіа-бібліотека

Пошук...

Українська

Кількість медіа
15

Підтримувані мови
2

Активний пошук
—

Завантаження контенту

Назва

Image

Категорія

Додати медіа

Фільтри

Тип контенту

Всі

Категорія

Всі

Перегляньте файли сюди або натисніть

Logged as: admin (admin)

Logout

Навчальне відео

video

Neural Learning

Delete

Наукова інфографіка

image

Research Data

Delete

Подкаст про хмарні технології

audio

Podcast Cloud

Delete

Посібник з хмарних технологій

document

Cloud Guide

Delete

Адміністратор

admin

Повний доступ

Додавання

Редагування

Видалення

Керування користувачами

Редактор

editor

Перегляд контенту

Завантаження файлів

Додавання медіа

Видалення файлів

Користувач

user

Перегляд контенту

Пошук та фільтрація

Завантаження файлів

Додавання медіа

Видалення файлів

Функція	User	Editor	Admin
Перегляд контенту	✓	✓	✓
Пошук та фільтрація	✓	✓	✓
Завантаження файлів	✗	✓	✓
Додавання медіа	✗	✓	✓
Видалення файлів	✗	✗	✓



Рольова модель доступу

3 рівні доступу: User, Editor, Admin для безпечної роботи.



Керування контентом

Додавання, організація та редагування медіа.



Безпека даних

Обмеження дій відповідно до прав користувача.



Контроль та порядок

Захист від несанкціонованих змін та видалення даних.

ВИСНОВКИ



- ✓ Розроблено інформаційну систему управління локалізованою медіа-бібліотекою

Система забезпечує централізоване зберігання, організацію та управління мультимедійним контентом через єдиний веб-інтерфейс з підтримкою локалізації та рольової моделі доступу.



- ✓ Реалізовано основні функції

- завантаження медіафайлів;
- зберігання та перегляд контенту;
- пошук і фільтрацію;
- підтримку локалізації інтерфейсу;
- рольову модель доступу.



- ✓ Підтримуються різні типи контенту



Зображення



Відео



Документи



- ✓ Забезпечено зручність використання

- сучасний веб-інтерфейс;
- швидкий доступ до медіа;
- централізоване керування контентом.



Поставлену мету досягнуто. Розроблена система забезпечує ефективне зберігання, пошук та керування мультимедійним контентом з підтримкою локалізації та розмежуванням прав доступу користувачів.



КПІ

ім. Ігоря Сікорського

ДЯКУЮ ЗА УВАГУ!

Інформаційна система управління
локалізованою медіабібліотекою

Виконав: Саюк Вадим Анатолійович

Керівник: Радченко Костянтин Олександрович



Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра системного програмування і спеціалізованих комп'ютерних систем