

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Фото-відео форум "PI-Archive"»

Виконав студент групи ПЗк-23-1
ІГНАТОВ Олександр Сергійович

Керівник роботи:
ГОДЛЕВСЬКИЙ Юрій Олександрович

Рецензент:
МОРОЗОВ Андрій Васильович

Житомир – 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«26» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ІГНАТОВ Олександр Сергійович**

Керівник роботи: **ГОДЛЕВСЬКИЙ Юрій Олександрович**

Тема роботи: **«Фото-відео форум "PI-Archive"»,**

затверджена Наказом закладу вищої освіти від **«23» лютого 2025 р., №74/с**

Вихідні дані для роботи: Об'єктом дослідження є використання інформаційних технологій для оптимального збереження та демонстрації медіаконтенту. Предметом дослідження є використання технологій розробки вебзастосунків для оптимального збереження та демонстрації медіаконтенту.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Годлевський Ю. О.	27.02.2025р.	27.02.2025р.
2	Годлевський Ю. О.	27.02.2025р.	27.02.2025р.
3	Годлевський Ю. О.	27.02.2025р.	27.02.2025р.

РЕФЕРАТ

Кваліфікаційна робота освітнього ступеня «бакалавр» складається з фото-відео форуму «Pi-Archive» та пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 58 сторінок, 41 ілюстрацію, та 13 таблиць. Загальний обсяг 60 сторінок.

Метою роботи є розробка фото-відео форуму. Система включає модулі створення та керування постами, автоматичної обробки відео файлів у універсальний формат, керування колекціями, керування тегами, перегляду постів.

У роботі поставлені основні завдання на розробку системи. Також було проведено порівняльний аналіз аналогічних систем. Було обрано клієнт-серверну архітектуру. Приведені основні сценарії використання системи, логіка взаємодії між модулями системи, діаграма класів та алгоритми роботи ключових модулів, спроектована база даних з використанням SQLITE. Клієнтська частина розроблена за допомогою чистих HTML, JS та компільованого SCSS, а серверна частина JS та SQL на базі Node.js.

КЛЮЧОВІ СЛОВА: ПОСТ, ТЕГ, ФОТО, ВІДЕО, КОЛЕКЦІЇ, РІДЕР, КОМЕНТАРІ, СЕЛФ-ХОСТЕД.

					ІПЗ.КР.Б – 121 – 25 – ЗПП						
Змн.	Арк.	№ докум.	Підпис	Дата	Фото-відео форум «Pi-Archive» Пояснювальная записка			Літ.	Арк.	Аркушів	
Розроб.		Ігнатов О.С.									
Керівник		Годлевський Ю.О.							3	57	
								Житомирська політехніка група ІПЗк-23-1			
Н. контр.											
Затвердив											

ABSTRACT

The qualification work for the bachelor's degree consists of a photo-video forum 'Pi-Archive' and an explanatory note. The explanatory note to the qualification work contains 58 pages, 41 illustrations and 13 tables. The total volume is 60 pages.

The aim of the work is to develop a photo-video forum. The system includes modules for creating and managing posts, automatically processing video files into a universal format, managing collections, managing tags, and viewing posts.

The paper sets out the main tasks for the development of the system. A comparative analysis of similar systems was also carried out. The client-server architecture was chosen. The main scenarios of using the system, the logic of interaction between the system modules, the class diagram and algorithms of the key modules are presented, and the database is designed using SQLITE. The client side is developed using pure HTML, JS and compiled SCSS, and the server side is developed using JS and SQL based on Node.js.

KEYWORDS: POST, TAG, PHOTO, VIDEO, COLLECTIONS, READER, COMMENTS, SELF-HOSTED.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – 3ПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РОЗДІЛ 1. АНАЛІЗ ВИМОГ КОРИСТУВАЧА	10
1.1. Постановка задачі	10
1.2. Аналіз аналогів програмного продукту	13
1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	18
Висновки до першого розділу	20
РОЗДІЛ 2. ПРОЄКТУВАННЯ ФОТО-ВІДЕО ФОРУМУ PI-ARCHIVE..	22
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	22
2.2. Розробка бази даних	28
2.3. Проєктування та реалізація алгоритмів роботи системи	31
2.4. Реалізація фото-відео форуму	35
Висновки до другого розділу	43
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ	44
3.1. Порядок встановлення та налаштування системи	44
3.2. Структура інтерфейсу та порядок взаємодії з фото-відео форумом	46
3.3. Тестування роботи програмного продукту	52
Висновки до третього розділу	55
ВИСНОВКИ	56
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	57
ДОДАТКИ	59

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – база даних

API – Application Program Interface

SQL - Structured Query Language

SCSS – Super Cascading Style Sheets

JSON – JavaScript Object Notation

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ВСТУП

Актуальність теми. Інтенсивний розвиток цифрових технологій та повсюдне поширення візуального контенту зумовлюють переосмислення класичних підходів до зберігання та обміну медіаданими. Активні учасники та творці цього візуального простору дедалі частіше констатують, що провідні платформи, незважаючи на їхню функціональну потужність, не завжди задовольняють глибинні потреби у формуванні згуртованих локальних спільнот. Алгоритми, орієнтовані на глобальне охоплення, нерідко залишають поза увагою специфіку камерної взаємодії, можливості персоналізованої організації контенту та, що суттєво, відчуття повного контролю користувача над власними цифровими активами.

Саме на цьому перетині технологічних можливостей та людської потреби у значущому спілкуванні й викристалізувалася актуальність теми даної кваліфікаційної роботи. Це зумовило нагальну потребу у створенні такого цифрового простору, який би не просто слугував архівом візуальних матеріалів, а став би своєрідним осередком, довкола якого могли б об'єднуватися індивіди зі спільними інтересами, формуючи тісні та довірливі спільноти. Йдеться про платформу, де кожен функціональний елемент – від завантаження файлу до його обговорення – був би інтуїтивно зрозумілим, безпечним та гнучким, адаптуючись до унікальних потреб кожної окремої групи.

Проблематика полягає не стільки у відсутності інструментарію як такого, скільки в дефіциті рішень, що гармонійно поєднують функціональність медіа-архіву з можливостями повноцінної соціальної мережі, орієнтованої на нішеві інтереси. Аспекти приватності, можливості локального розгортання для закритих спільнот, розширені інструменти для тематичної категоризації та обговорень – усе це часто залишається на периферії уваги масових сервісів. Розробка фото-відео форуму "PI-Archive" є спробою відповісти на ці виклики, запропонувавши альтернативу, що ставить у центр саме користувача та його спільноту.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

Метою кваліфікаційної роботи є проєктування та розробка інноваційного фото-відео форуму "PI-Archive" – програмного продукту, покликаного стати гнучкою, безпечною та інтуїтивно зрозумілою платформою для обміну, зберігання та обговорення візуального медіаконтенту всередині локальних спільнот. Ключовим прагненням було створення не просто сховища файлів, а повноцінного інтерактивного середовища, що сприяє формуванню та розвитку тематичних об'єднань користувачів.

Досягнення поставленої мети передбачало вирішення комплексу завдань. Початковим етапом став глибокий аналіз предметної області та існуючих аналогічних рішень, що дозволило ідентифікувати їхні сильні сторони та функціональні прогалини, які потребують заповнення. Наступним кроком було сформульовано вичерпний перелік функціональних та нефункціональних вимог до майбутньої системи, враховуючи потреби потенційних користувачів та сучасні тенденції у веб-розробці. Важливим етапом стало обґрунтування вибору технологічного стеку та архітектурних рішень, що забезпечили б оптимальний баланс між продуктивністю, масштабованістю, безпекою та швидкістю розробки. В рамках проєктування була реалізована модульна архітектура платформи, що включає центральне ядро системи ("Core"), багатофункціональний веб-інтерфейс ("Web") та інтегрований модуль для взаємодії через Telegram-бот ("TGBot"). Також була розроблена реляційна модель даних та відповідна структура бази даних (з використанням SQLite), здатна ефективно зберігати та обробляти інформацію про користувачів, їхній контент, теги, коментарі та взаємозв'язки. Ключовий функціонал платформи, що включає механізми реєстрації та автентифікації, інструменти завантаження, перегляду, детального тегування, коментування та групування фото та відео матеріалів, був імплементований з акцентом на високий рівень юзабіліті та інтуїтивно зрозумілий, адаптивний користувацький інтерфейс. Завершальним етапом стало проведення комплексного тестування розробленого програмного продукту для виявлення

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

та усунення можливих дефектів, а також для підтвердження його відповідності заявленим вимогам.

Об'єктом дослідження у даній кваліфікаційній роботі виступає комплексний процес проєктування, розробки та впровадження спеціалізованої веб-платформи, призначеної для організації спільного доступу, управління та інтерактивної взаємодії навколо фото- та відеоконтенту. Це дослідження охоплює весь життєвий цикл створення програмного продукту: від концептуалізації ідеї та аналізу потреб користувачів до архітектурного дизайну, програмної реалізації функціональних модулів, розробки користувацького інтерфейсу та фінального тестування готової системи.

Предметом дослідження є сукупність моделей, методів, алгоритмів, архітектурних підходів та інструментальних програмних засобів, застосованих для створення та забезпечення ефективного функціонування фото-відео форуму "PI-Archive". Предмет дослідження охоплює архітектурні патерни, зокрема клієнт-серверну модель та принципи модульного дизайну, що лягли в основу побудови гнучкої системи з компонентами "Core", "Web" та "TGBot". Технологічний стек представлений середовищем виконання Node.js та фреймворком Express.js для серверної логіки, тоді як клієнтська частина реалізована з використанням нативних веб-технологій (HTML5, CSS3, SCSS, JavaScript ES6+). Важливим аспектом є системи управління базами даних, зокрема вибір та інтеграція легкої файлової СУБД SQLite для зберігання структурованих даних програми, включаючи розробку схеми бази даних та SQL-запитів.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ ВИМОГ КОРИСТУВАЧА

1.1. Постановка задачі

Метою кваліфікаційної роботи бакалавра є проектування та реалізація інноваційного фото-відео форуму “PI-Archive”, що слугує платформою для поширення та обміну візуальним медіа-контентом серед локальних спільнот. У межах наукового підходу передбачається забезпечити простий, інтуїтивно зрозумілий та захищений доступ до накопичених матеріалів всередині сервісу, а також гарантувати комфортний перегляд будь-яких форматів медіафайлів незалежно від їх розміру чи жанрової приналежності. При цьому особлива увага приділяється архітектурній гнучкості системи та можливості її подальшого масштабування відповідно до росту обсягу даних і кількості користувачів.

Архітектура платформи складається з двох основних модулів та одного додаткового компонента, кожен із яких виконує спеціалізовані функції. Першим базовим компонентом є модуль “Core” – ядро системи, яке відповідає за взаємодію з базою даних за допомогою уніфікованих API-інтерфейсів та прямого виконання SQL-запитів. Йому відводиться роль центрального координаційного блока, що опрацьовує запити, забезпечує цілісність даних та оптимізує продуктивність зберігання й вибірки інформації.

Другим ключовим елементом є модуль “Web” – веб-інтерфейс із повним спектром функціональних можливостей платформи. Він реалізує фронтенд-логіку, що включає навігацію по контенту, механізми пошуку та фільтрації, відображення галерей, а також розширені інструменти управління обліковим записом і ролями користувачів. У його межах передбачено адаптивний дизайн із підтримкою мультимовності та налаштувань тем оформлення, що сприяє підвищенню зручності та доступності ресурсу.

Опціональним, проте структурно інтегрованим компонентом виступає модуль “TGBot” – телеграм-бот для взаємодії з архівом через месенджер Telegram. Його завданням є спрощення процесів завантаження нових постів,

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

отримання матеріалів і редагування базових метаданих (теги, описи) безпосередньо з мобільного пристрою. Реалізація цього модуля передбачає модульний обробник команд авторизації та безпечне керування потоками даних між ботом та ядром системи.

У результаті дослідження і розробки має бути створено високодоступний веб-сервіс, який надає користувачам можливість завантажувати, переглядати та обговорювати різноманітні фото та відео-файли. Передбачається впровадження системи коментування, приватних повідомлень для ведення дискусій у закритому режимі, а також відображення публікацій у вигляді зручних галерей із можливістю швидкого переходу до детального перегляду окремих елементів контенту. Таке комплексне рішення створить комфортне середовище для спільнот, які прагнуть ділитися візуальним матеріалом у безпечному та організованому форматі.

У процесі створення платформи перший етап передбачає розробку мінімального «Core» модуля, який закладає фундамент всіх подальших компонентів системи.

Насамперед, реалізується ініціуючий скрипт, що відповідає за перевірку наявності оновлень, а також автоматичне створення необхідних файлів конфігурації та службових ресурсів. Далі здійснюється формалізація мінімальної організації модульності ядра, що передбачає чітке розмежування відповідальностей між окремими підсистемами та забезпечує гнучкість подальшої розширюваності. Завершальним кроком цієї стадії є розробка детальної схеми бази даних, яка відображає основні сутності та їх взаємозв'язки на рівні моделі даних.

Другий етап охоплює створення прототипу мінімальних бекенд та фронтенд компонентів у складі «Web» модуля.

У межах серверної частини реалізується базовий веб-сервер, здатний обробляти вхідні HTTP-запити та передавати їх на подальшу обробку. Паралельно розробляється механізм обробки API-запитів веб-сторінки, що забезпечує стандартизовану взаємодію між клієнтською частиною та ядром

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

системи. На стороні інтерфейсу проводиться прототипування та первісна реалізація дизайну ключових сторінок, таких як пошук, перегляд контенту, завантаження, користувацький профіль та адмін-панель, із врахуванням базових принципів взаємодії з користувачем.

Третій етап присвячено організації модульного API та створенню репозиторію бази даних.

Тут розробляються спеціалізовані обробники API-запитів для «Web» та «TGBot» модулів, що гарантують уніфіковану роботу з бізнес-логікою платформи. Додатково створюється і документується набір API-ендпоінтів, які забезпечують всю необхідну функціональність для взаємодії з платформою через зовнішні системи. Паралельно здійснюється розробка репозиторію бази даних — набір SQL-запитів і процедур для ефективного зберігання, пошуку та маніпулювання даними, що підвищує продуктивність і безпеку системи.

Четвертий етап спрямований на повноцінну реалізацію «Web» модуля із розширеними можливостями та вдосконаленим інтерфейсом користувача.

Здійснюється глибинне втілення функціональності пошукової сторінки з урахуванням фільтрації, сортування та навігації. Також реалізується сторінка колекцій і механізм перегляду окремих елементів колекції з забезпеченням інтерактивності та адаптивності. Паралельно проводиться редизайн усіх сторінок із запровадженням єдиного стилю, а також створюється система тем оформлення та мультимовна підтримка, що забезпечують персоналізацію інтерфейсу під запити різних категорій користувачів.

П'ятий етап передбачає розробку окремого модуля «TGBot», який інтегрується у загальну архітектуру через модульний підхід.

Визначається та реалізується набір обробників команд авторизації користувачів, витягування та редагування метаданих постів (тегів, описів). Особлива увага приділяється механізму автоматичного вивантаження файлів при отриманні повідомлення з вкладенням, за умови успішного проходження перевірок авторизації, що забезпечує безперебійну та безпечну взаємодію бот-інтерфейсу з основною системою.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		12

Завершальний, шостий етап полягає в адаптації та розширенні «Web» модуля з урахуванням функціоналу, запровадженого в модулях «TGBot» та «Core». Впроваджуються опції отримання та обробки файлів безпосередньо через веб-інтерфейс, що забезпечує єдину точку доступу для користувачів. Окрім того, реалізується коректне відображення інформації, пов'язаної з діяльністю «TGBot» (наприклад, статуси завантаження чи метадані), що гарантує узгодженість даних та покращує користувацький досвід платформи.

Результатом виконання плану дій є створення вебплатформи, яка надасть можливість користувачам переглядати, оцінювати та коментувати пост. Адміністратори матимуть змогу керувати групами тегів, реагувати на запити користувачів та підключати незалежні модулі. Архів дозволить створювати персоналізовані профілі, редагувати, завантажувати та вивантажувати пости через Telegram-бота, надаючи доступ до безпечної та сучасної системи для обміну контентом та враженнями.

1.2. Аналіз аналогів програмного продукту

Для розробки системи «PI Archive» було проведено аналіз існуючих аналогів для виявлення переваг та недоліків конкурентних рішень. Основну увагу було приділено сервісам: Google Photos, Flickr, Pinterest.

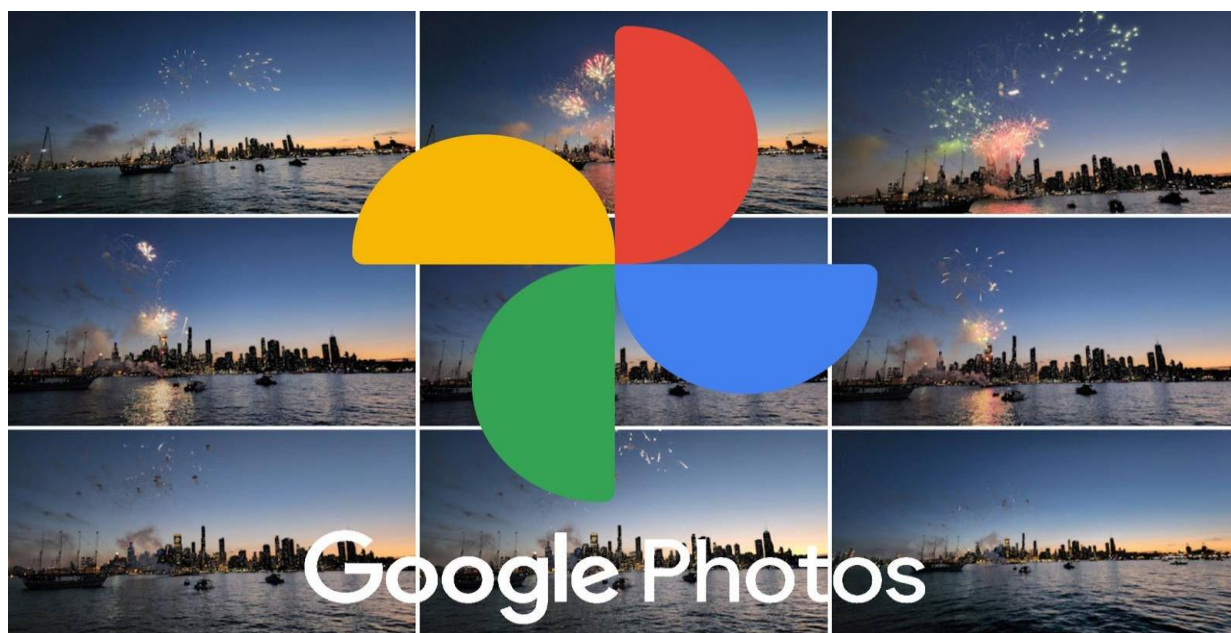


Рис. 1.1. Google Photos

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Google Фото пропонує значну кількість переваг, що роблять його популярним серед користувачів. Однією з головних особливостей є автоматичне розпізнавання зображень, яке дозволяє системі організовувати фотографії без необхідності вручну призначати теги чи сортувати їх. За допомогою цієї технології користувачі можуть швидко знайти конкретні зображення, навіть якщо вони не мають чітких назв чи підписів. Крім того, Google Фото автоматично створює альбоми на основі подій, місць або осіб, що зображені на фото, що значно спрощує організацію медіафайлів. Інтеграція з іншими сервісами Google, такими як Google Drive, Google Assistant та Google One, дозволяє забезпечити зручність в обміні файлами та їх збереженні. Висока швидкість роботи та інтуїтивно зрозумілий інтерфейс дозволяють користувачам без проблем навігувати в додатку і швидко знаходити потрібні фото, що значно покращує загальний досвід користування платформою.

Незважаючи на численні переваги, Google Фото має й певні недоліки, які можуть обмежувати користувачів. Одним із основних мінусів є обмеження на безкоштовне зберігання даних. Хоча Google Фото раніше пропонував безкоштовне зберігання в необмеженій кількості, тепер існують ліміти для високоякісних фотографій і відео, що змушує користувачів платити за додаткове сховище через Google One, коли обсяг даних перевищує певну межу. Іншим важливим недоліком є питання конфіденційності. Оскільки дані зберігаються на серверах Google, є певний ризик, що користувацька інформація може бути оброблена централізовано, що викликає побоювання щодо безпеки персональних даних. Хоча Google має політику конфіденційності, яку багато користувачів вважають достатньою, все ж зберігання приватних медіафайлів у хмарі великої корпорації не завжди викликає довіру серед усіх користувачів.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		14

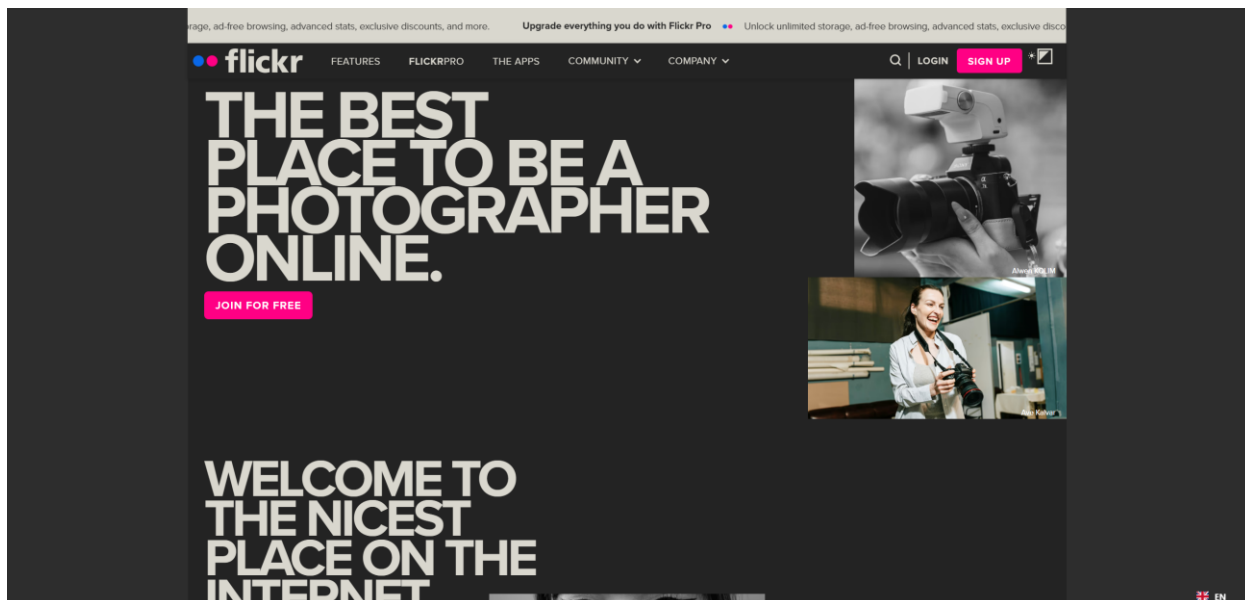


Рис. 1.2. Flickr

Flickr є одним із найвідоміших спеціалізованих сервісів для фотографів, який пропонує низку переваг, що вирізняють його серед інших платформ для зберігання та поширення зображень. Однією з ключових особливостей є можливість детального налаштування альбомів, що дозволяє користувачам організовувати власне портфоліо за темами, подіями або стилями з високим рівнем точності. Платформа також підтримує зображення у високій якості, що особливо важливо для професійних фотографів і тих, хто працює з великими файлами без втрати деталей. Крім цього, Flickr має розвинену соціальну складову: система коментарів, груп, можливість залишати вподобання та брати участь в обговореннях сприяє активній взаємодії між авторами та аудиторією. Це створює сприятливе середовище для зворотного зв'язку, творчої співпраці та натхнення в межах фотографічної спільноти.

Попри свої переваги, Flickr має і ряд недоліків, які можуть обмежити його привабливість для ширшої аудиторії. Найпомітнішим є обмеження в базовому тарифному плані: безкоштовна версія дозволяє завантажити лише обмежену кількість фотографій, а для отримання повного функціоналу користувачам необхідно оформлювати платну підписку. Це може бути суттєвим бар'єром для аматорів або новачків, які ще не готові інвестувати кошти у сервіс. Окрім того, інтерфейс платформи, хоч і функціонально

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк. 15
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

насичений, іноді виявляється надто складним для користувачів без спеціалізованого досвіду у сфері цифрової фотографії чи веб-інструментів. Надлишок функцій і складна навігація можуть створити додаткові труднощі при першому знайомстві з платформою, особливо для тих, хто шукає простий спосіб обміну фото без глибокого налаштування.

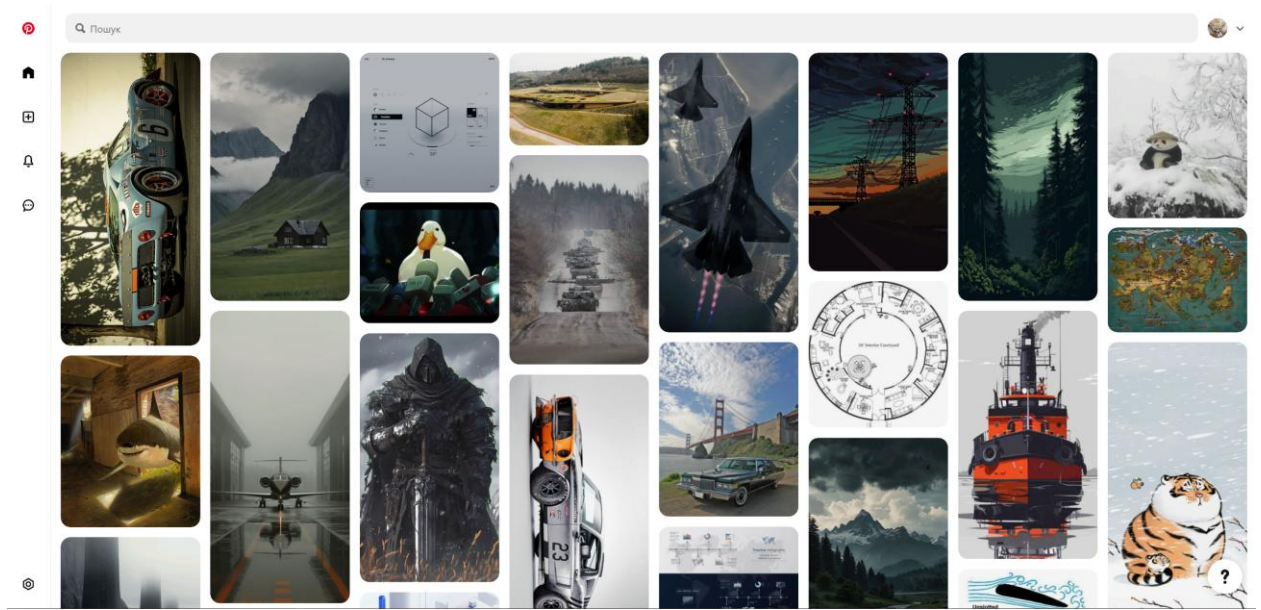


Рис. 1.3. Pinterest

Pinterest є потужним інструментом для візуального пошуку і збереження ідей, що вирізняється своєю естетикою та інтуїтивно зрозумілим підходом до організації контенту. Його головною перевагою є візуально привабливий інтерфейс, який дозволяє користувачам легко створювати та впорядковувати колекції зображень за допомогою так званих «дошок». Такий формат сприяє швидкому доступу до візуального матеріалу, надихає на творчість та відкриває нові дизайнерські й естетичні ідеї. Сервіс активно використовується для планування проєктів, добору стилістичних рішень у сфері моди, інтер'єру, гастрономії та інших креативних напрямків. Додатково, проста система категоризації дозволяє швидко зберігати й знаходити контент за інтересами без складних технічних налаштувань.

Втім, Pinterest має обмеження, які знижують його ефективність для користувачів, що потребують більш глибокої організації архівів.

		Игнатов О.С.			ИПЗ.КР.Б – 121 – 25 – 3ПП	Арк.
		Годлевський Ю.О.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Орієнтованість платформи переважно на декоративний та інспіраційний контент означає, що вона менш придатна для структурованого та професійного зберігання великих обсягів медіа з чіткою ієрархією. Сервіс не підтримує розширених функцій тегування, метаданих або створення складних категорій, що може бути критичним для користувачів, які прагнуть систематизувати архіви за технічними чи тематичними параметрами. Крім того, можливості персоналізації й управління контентом є досить обмеженими: відсутні гнучкі налаштування сортування, автоматизації або імпорту-експорту колекцій, що ускладнює використання Pinterest як професійного інструменту для роботи з медіафайлами.

Таблиця 1.1

Таблиця порівняння сервісів

Характеристика	Google Photos	Flickr	Pinterest	PI Archive
Простота використання	Висока	Середня	Висока	Висока
Система організації (теги, групи)	Автоматична + ручна	Ручна з налаштуванням	Ручна (дошки)	Розширена (теги, групи, колекції)
Масштабованість та адаптивність	Висока	Середня	Середня	Висока, завдяки модульній архітектурі
Безпека даних	Середня	Висока	Середня	Висока (локальна ізоляція)
Підтримка локального запуску	Ні	Обмежена	Ні	Так (з використанням SQLite)
Фінансова модель	Freemium	Freemium	Freemium	Безкоштовний Open Source проект

У результаті аналізу функціональних можливостей популярних платформ для перегляду та зберігання зображень і візуального контенту було виокремлено низку характерних рис, які притаманні переважній більшості з них. До основних переваг належать: наявність великої бібліотеки візуального матеріалу (зокрема манги), інтерактивні елементи у вигляді системи оцінювання та коментування, інтерфейс, оптимізований для зручної навігації, а також можливість безкоштовного доступу до базового функціоналу. Ці особливості забезпечують базовий рівень зручності та користувацького досвіду. Водночас аналіз показав, що жодна з розглянутих платформ не забезпечує повноцінної взаємодії між користувачами, не передбачає можливості публікації авторського контенту в зручній формі, а також не підтримує створення персоналізованих профілів із розширеним набором функцій.

У зв'язку з виявленими обмеженнями наявних рішень було прийнято концептуальне рішення про розробку нової платформи, яка б поєднувала переваги вже існуючих сервісів і водночас долала їхні недоліки. Передбачається, що ця платформа стане не лише сховищем і переглядачем візуального контенту, а й інструментом соціальної взаємодії. Зокрема, користувачі матимуть змогу створювати персоналізовані акаунти, ділитися власними творами, отримувати зворотний зв'язок, а також формувати мережі за інтересами. Додатково планується реалізація інтеграції з Telegram-ботом для автоматизованої взаємодії, а також розширення доступу до альтернативних і менш відомих манга-творів, що сприятиме розвитку творчої спільноти навколо платформи.

1.3. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

У процесі розробки вебдодатку було прийнято рішення використовувати клієнт-серверну архітектуру. Такий підхід дозволяє чітко розмежувати функціональні зони між клієнтською та серверною частинами, забезпечуючи

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

ефективну взаємодію між компонентами системи. Архітектура сприяє масштабованості, підвищує гнучкість у підтримці та модернізації програмного забезпечення, а також забезпечує розділення відповідальності між інтерфейсною логікою та логікою обробки даних.

Для реалізації серверної частини було обрано середовище виконання Node.js, яке дозволяє використовувати JavaScript як на клієнтській, так і на серверній стороні. Node.js є асинхронним, неблокуючим середовищем, що дозволяє ефективно обробляти численні запити без затримок. Такий підхід є особливо актуальним для додатків, які працюють з медіа-контентом, де важлива швидкість обробки запитів та мінімальні затримки при завантаженні даних.

Для побудови RESTful API використовується фреймворк Express.js, який є одним з найбільш популярних рішень у екосистемі Node.js. Express.js забезпечує гнучку маршрутизацію, просту інтеграцію з middleware, обробку запитів та відповідей, а також масштабування логіки додатку без надмірної складності. Це дозволяє швидко створювати та підтримувати API, через яке здійснюється взаємодія клієнта з сервером.

Як систему управління базами даних було обрано SQLite. Вона є портативною, легкою, не потребує окремого серверного програмного забезпечення, а всі дані зберігаються у звичайному файлі на диску. Це забезпечує зручність у розробці, простоту розгортання та мінімальні вимоги до апаратного забезпечення. SQLite ідеально підходить для проєктів, які плануються для локального використання або невеликого навантаження, де важлива простота та швидкодія.

Фронтенд додатку реалізовано з використанням нативних технологій: HTML, CSS та JavaScript. Такий підхід забезпечує максимальну контрольованість коду, незалежність від сторонніх бібліотек та фреймворків, а також сприяє оптимізації часу завантаження сторінки. Використання чистого JavaScript дозволяє краще контролювати логіку відображення, взаємодії з елементами сторінки та зворотного зв'язку з користувачем.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

Для стилізації інтерфейсу застосовуються SCSS-файли, які компілюються на стороні сервера у звичайні CSS-стилі. SCSS забезпечує підтримку змінних, вкладених селекторів, міксинів та інших можливостей, що значно підвищують гнучкість та повторне використання стилів. Компіляція на сервері дозволяє зберігати стабільну структуру стилів, а також спрощує процес розгортання нових змін.

Загальна конфігурація платформи дозволяє забезпечити високу продуктивність, низькі вимоги до обчислювальних ресурсів та швидке розгортання на будь-якому локальному або хмарному середовищі. Для базового запуску додатку достатньо машини з 2 ГБ оперативної пам'яті, багатоядерним процесором (від 1.5 ГГц і вище) та мінімумом 1 ГБ вільного дискового простору для зберігання медіаконтенту.

Таким чином, обрані інструментальні засоби забезпечують баланс між функціональністю, простотою реалізації, продуктивністю та масштабованістю. Такий підхід дозволяє у майбутньому легко оновлювати функціональність, підключати сторонні сервіси та адаптувати систему до нових вимог без істотних змін у базовій архітектурі.

Висновки до першого розділу

У ході аналізу предметної області було визначено ключові потреби користувачів сучасних фото- та відеоплатформ: зручна організація контенту, можливість взаємодії з іншими користувачами, персоналізація інтерфейсу, а також публікація власного мультимедійного матеріалу. Було досліджено функціональні характеристики існуючих сервісів, таких як Google Фото, Flickr та Pinterest. Це дозволило виділити як сильні сторони — автоматичну організацію контенту, високу якість зображень, зручні інтерфейси — так і недоліки, серед яких: обмеження на безкоштовне зберігання, слабка підтримка персоналізованої структури, а також складність для користувачів без спеціальної підготовки.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

З огляду на виявлені потреби, було прийнято рішення про створення власної платформи — фото-відео форуму з розширеною соціальною взаємодією. Основними функціями майбутньої системи мають стати: підтримка обміну зображеннями та відео, коментування, система оцінювання, створення альбомів, а також можливість налаштовувати персоналізований профіль із системою підписок і сповіщень. Додатково передбачено інтеграцію з Telegram-ботом для зручного отримання оновлень і новин. Такий підхід дозволить створити спільноту, орієнтовану не лише на пасивне споживання контенту, а й на активну взаємодію між користувачами.

Для реалізації проєкту обрано сучасний технологічний стек: серверна частина реалізується з використанням Node.js та Express.js для організації API, база даних представлена у вигляді SQLite — портативної та ефективною для локального зберігання. Клієнтська частина побудована на основі чистих HTML, CSS (компільованих із SCSS) та JavaScript, що забезпечує контроль над зовнішнім виглядом та оптимізацію швидкодії. Такий вибір інструментів дозволяє створити швидкий, адаптивний та масштабований додаток, що повністю відповідає поставленим функціональним цілям.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2. ПРОЄКТУВАННЯ ФОТО-ВІДЕО ФОРУМУ PI-ARCHIVE

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

З метою забезпечення стабільної та повноцінної роботи веб платформи було здійснено проектування ключових функціональних модулів, орієнтованих на потреби користувачів та вимоги до сучасних медіа форумів. Розроблена система охоплює базові та розширені можливості, необхідні для комфортної взаємодії з контентом. Зокрема, реалізовано модулі пошуку публікацій, перегляду окремих постів і колекцій, завантаження нового контенту, а також компоненти для особистого листування, подання скарг, управління користувацьким профілем та адміністрування платформи.

Архітектура веб додатку передбачає збереження структурованих даних, що є необхідними для повноцінної роботи зазначених модулів. У сховищі бази даних зберігається інформація про користувачів, окремі пости, коментарі, теги, повідомлення, рейтинги, а також групування постів у тематичні колекції. Такий підхід до моделювання даних забезпечує ефективне виконання запитів, підтримку персоналізованої взаємодії та масштабування функціональності в майбутньому. З цілю надання користувачам повноцінного та безвідмовного функціоналу були спроектовані модулі: пошук постів, перегляд постів, перегляд колекцій та колекції, завантаження постів, особисті повідомлення та скарги, профіль користувача та панель адміністратора. Програмою передбачено зберігання даних про: користувачів, пости, теги, повідомлення та коментарі, групи постів, рейтинги.

Користувачі додатку

Для додатку було зроблено обов'язкову реєстрацію та авторизацію для захисту файлів від стороннього доступу, також користувачі були розділені на декілька рангів для реалізації обмеження доступу певних функцій на основі рангу, їх опис приведено нижче (табл. 2.1).

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Таблиця 2.1

Ранги користувачів

Ранг	Опис
Гість	Користувач, який уперше зайшов на сторінку, не має аккаунта та повинен створити його. Не має доступу нікуди окрім сторінок «вітання», «реєстрація», «авторизація»
Unverified	Користувач, який зареєстрував аккаунт, але ще не перевірений адміністратором чи володарем. Має доступ до усіх сторінок що й гість, але при спробі авторизації отримає відповідь що його аккаунт ще не перевірено та він не має доступу до сайту
User	Користувач, який зареєстрований та перевірений адміністратором, має доступ до усіх сторінок за виключенням панелі адміністратора. Може переглядати, коментувати, ставити оцінки постам та додавати їх у обране, може писати у особисті повідомлення людей та відправляти скарги на пости інших людей.
Moderator	Користувач наслідує усі можливості користувача, але може редагувати теги та описи чужих постів.
Admin	Користувач має усі права на редагування любого посту, навіть його видалення, має доступ до панелі адміністратора, але не має доступу до функції перезавантаження додатку.
Owner	Користувач має усі права, у тому числі перезавантаження додатку.

Варіанти використання

Дія	Ранг	Опис
Реєстрація	Гість	Створення нового аккаунту з поміткою статусу «unverified».
Авторизація	User+	Використання користувачем його логіну та паролю для виконання входу в додаток.
Створення посту	User+	Завантаження користувачем файлу. Опціонально: додати теги та опис до посту
Редагування тегів	User+ (Якщо користувач не автор то Moderator+)	Зміна тегів посту для спрощення його пошуку.
Редагування опису	User+ (Якщо користувач не автор то Moderator+)	Зміна опису посту.
Видалення посту	User+ (Якщо користувач не автор то Admin+)	Повне видалення посту.
Зміна рейтингу	User+	Встановлення «лайку» чи «дизлайку» на пост.
Додавання у «обране»	User+	Обирання посту до особистої групи «обране»
Обрання посту як «аватар» чи «фон профілю»	User+	Обирання посту, щоб той був відображений замість аватару чи фону на сторінці профілю користувача.
Обробка запитів нових користувачів	Admin+	Зміна мітки статусу іншого аккаунту з «unverified» на «user».

У процесі розробки вебплатформи було визначено низку функціональних та нефункціональних вимог, які повинні бути враховані для забезпечення стабільної та ефективної роботи системи. Функціональні вимоги регламентують основні можливості взаємодії користувача з системою, зокрема щодо авторизації, реєстрації та управління збереженими даними. Нефункціональні вимоги, у свою чергу, охоплюють аспекти сприйняття, надійності, продуктивності та експлуатаційної придатності, що безпосередньо

впливають на якість користувацького досвіду та технічну ефективність реалізації.

Узагальнені вимоги до системи наведено в таблиці 2.3

Таблиця 2.3

Вимоги додатку

Категорія	Підкатегорія	Вимога
Функціональні	Авторизація	Реалізація механізму реєстрації користувачів та присвоєння ролей (користувач, адміністратор).
	Збереження даних	Система повинна надавати можливість зберігання та керування інформацією користувачем.
Нефункціональні	Сприйняття	Час самостійного ознайомлення з продуктом – не більше 2 годин. Час відгуку для типових задач – до 15 секунд, для складних – до 30 секунд. Інтерфейс має бути інтуїтивно зрозумілим і супроводжуватися підказками.
	Надійність	Допустимий час обслуговування – не більше 6% від загального часу роботи. Середній час безвідмовної роботи – 15 робочих днів. Максимальна кількість дефектів – не більше 1 помилки на 10 000 рядків коду.
	Продуктивність	Система повинна підтримувати щонайменше 40 одночасних користувачів з доступом до спільної бази даних.
	Експлуатація	Система має забезпечувати можливість масштабування без погіршення її продуктивності при збільшенні навантаження.

Виходячи з аналізу вимог функціональності можемо виділити наступні сутності, що забезпечать реалізацію фото-відео форуму.

Об'єктно-орієнтована структура системи у вигляді діаграми класів приведена нижче (рис. 2.1), яка відображає зв'язки між сутностями.

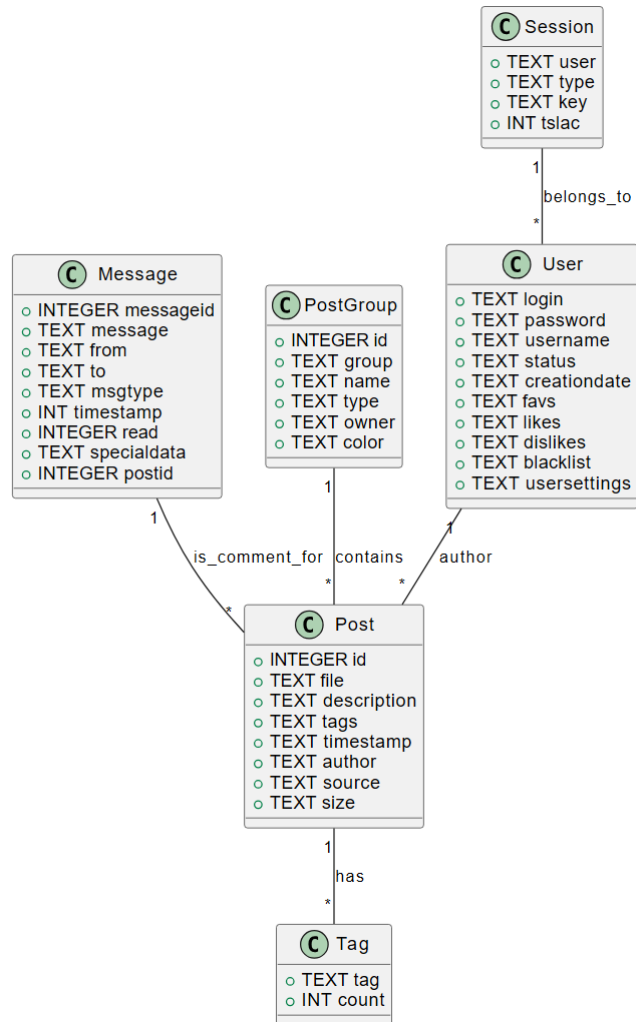


Рис. 2.1 – Діаграма класів

Діаграма класів, зображена на рисунку 2.1, демонструє об'єктно-орієнтовану структуру проєктованої інформаційної системи. Вона відображає основні сутності, їхні властивості та взаємозв'язки, що забезпечують повноцінну функціональність програмного продукту. Центральним елементом є клас User, який представляє обліковий запис користувача. До його складу входять поля для зберігання логіна, пароля, відображуваного імені, а також статусу, що визначає рівень прав доступу. Додатково передбачено можливість зберігати персональні списки, такі як улюблені пости, лайки, дизлайки, чорний список, а також об'єкт налаштувань користувача у вигляді серіалізованого JSON. Кожен користувач також має дату реєстрації, сесію та може публікувати контент і надсилати повідомлення.

Клас Post реалізує сутність публікації, що може містити зображення або відео з відповідним описом. До нього належать такі атрибути, як унікальний ідентифікатор, шлях до файлу, текстовий опис, список тегів (у форматі JSON), час публікації, логін автора, джерела походження контенту, а також інформація про розмір і тривалість медіафайлу. Таким чином, пост є основною одиницею контенту, створюваною користувачами.

Для класифікації постів використовується клас Tag, який зберігає назву тегу і кількість разів, коли його було застосовано. Теги сприяють тематичній організації контенту та полегшують його пошук. Крім того, вони можуть використовуватись для реалізації механізмів рекомендацій або фільтрації.

Організація постів у логічні об'єднання здійснюється за допомогою класу PostGroup, що містить унікальний ідентифікатор, список ID постів (у вигляді JSON), назву групи, її тип, логін власника, а також колірну мітку для візуального розрізнення. Ця сутність дозволяє створювати колекції або групи постів, що полегшує навігацію та структурування контенту.

Для забезпечення механізму автентифікації та відстеження активності користувачів у системі використовується клас Session. У ньому зберігаються логін користувача, тип доступу (наприклад, через веб або телеграм-бота), унікальний ключ сесії та мітка часу останньої активності. Така реалізація дозволяє забезпечити безпеку користувацьких дій та контроль доступу.

Клас Message реалізує функціональність комунікації між користувачами, а також використовується для зберігання коментарів до постів або службових повідомлень. Кожне повідомлення має унікальний ідентифікатор, текст, відправника та отримувача, тип повідомлення, мітку часу, статус прочитання, а також можливість зберігати додаткові службові дані у вигляді JSON.

Загалом, представлена модель охоплює основні функціональні аспекти системи — автентифікацію, управління користувачами, обробку контенту, комунікацію та категоризацію даних. Вона демонструє взаємозв'язки між

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

класами та створює гнучку основу для реалізації масштабованої інформаційної системи з багатим функціоналом та підтримкою персоналізації.

Повідомлення зв'язані з постами функціонують як коментарі, повідомлення зв'язані з логінами інших користувачів це особисті повідомлення або скарги.

2.2 Розробка бази даних

Дані додатку зберігаються у базі даних SQLite, було виділено таблиці: users, posts, posts_groups, tags, tags_groups, temp_tags, messages, sessions, temp_keys.

Після першочергового аналізу спланованої структури БД були виявлені недоліки, та деякі аспекти не були враховані на стадії планування, тому фінальний вигляд бази даних частково відрізняється від запланованого.

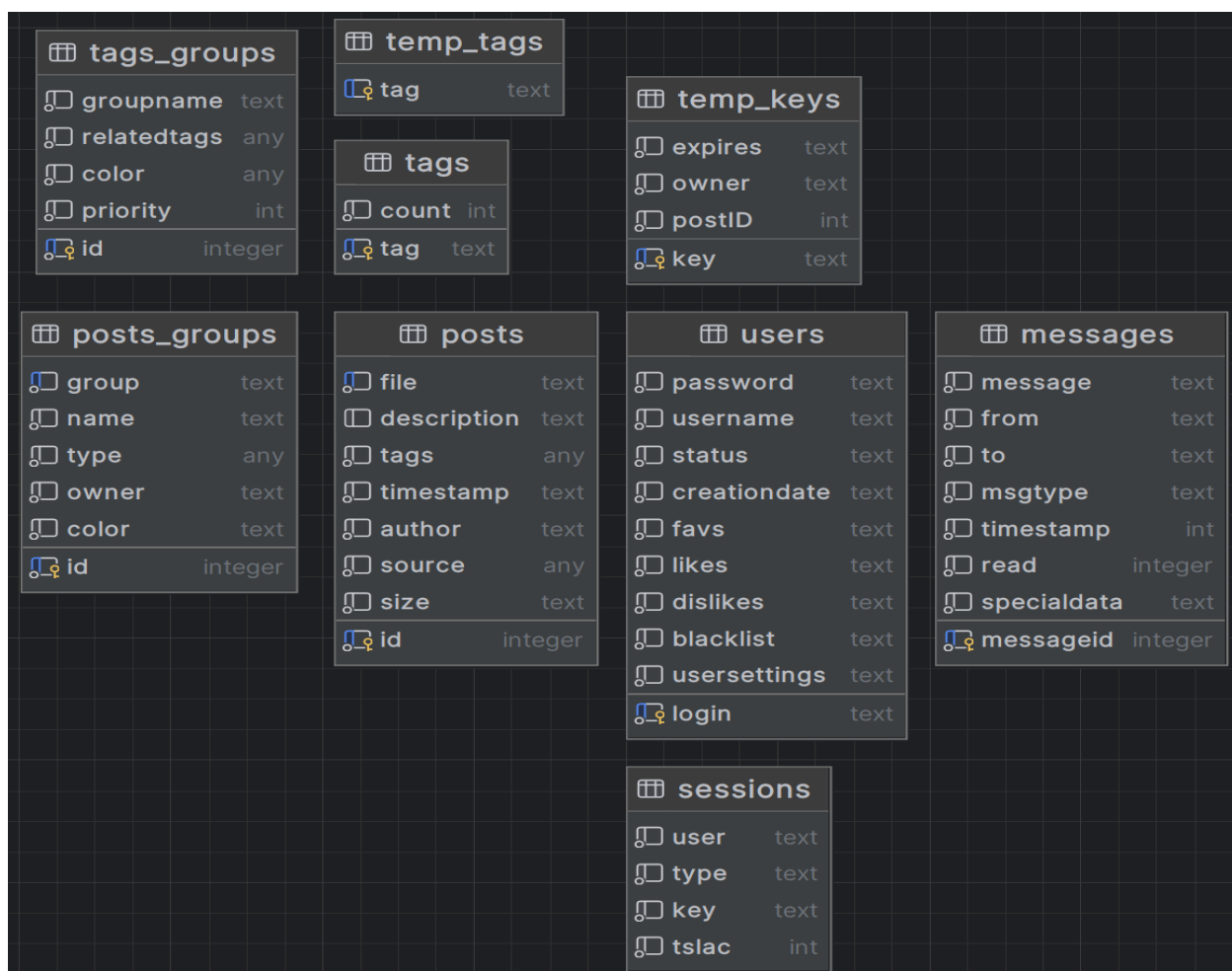


Рис 2.2 - структура БД

Опис по кожному полю та таблиці наведено у таблицях 2.4 – 2.9

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

Таблиця 2.4

Опис таблиці «users»

Поле	Опис
login (PK)	Логін користувача
password	Пароль користувача
username	Відображуване ім'я користувача
status	Статус/роль користувача
creationdate	Дата реєстрації профілю
favs	Обране
likes	Лайкнуті пости та групи
dislikes	Дизлайкнуті пости та групи
blacklist	Чорний список тегів
usersettings	Налаштування користувача у форматі JSON

Таблиця 2.5

Опис таблиці «posts»

Поле	Опис
id	Унікальний ідентифікатор посту
file	Ім'я файлу який асоційований з записом у БД
description	Текстовий опис посту
tags	Теги посту у форматі JSON
timestamp	Дата створення посту
author	Автор посту (логін користувача)
source	Оригінальне джерело файлу
size	Розмір файлу

Таблиця 2.6

Опис таблиці «post_groups»

Поле	Опис
id	Унікальний ідентифікатор групи постів
group	JSON з списком постів
name	JSON з іменами груп для усіх локалізацій
type	Тип групи, просто група або колекція
owner	Автор групи
color	Колір виділення групи у різних списках

Таблиця 2.7

Опис таблиці «tags»

Поле	Опис
tag	Назва тегу
count	Кількість постів пов'язаних з цим тегом

Таблиця 2.8

Опис таблиці «tags_groups»

Поле	Опис
groupname	JSON з назвами групи тегів для усіх мов
relatedtags	JSON список тегів які включені у цю групу
color	Колір який матимуть теги у цій групі
priority	Пріоритет відображення груп тегів при перегляді посту/колекції

Таблиця 2.9

Опис таблиці «temp_tags»

Поле	Опис
tag	Тимчасовий тег для коректного підрахунку кількості постів з цим тегом

Використання бази даних SQLite зумовлене тим, що вона проста в локальному розміщенні та запуску. SQLite не потрібні повноцінні системи керування базами даних, так як вона розміщується у формі файлу та з легкістю може використовуватись простим пакетом sqlite3 для Node.js. Подібний метод використання універсальний та кросплатформенний що дозволяє запускати додаток як на Windows так і на Linux.

2.3 Проектування та реалізація алгоритмів роботи системи

Архів у своєму складі має 3 шари, кожен з них в тій чи іншій мірі ізольований один від іншого та має певні права доступу один до одного, ці шари описані в таблиці 2.10.

Таблиця 2.10

Програмні шари додатку

Назва	Опис	Зв'язки
Зовнішні сервіси	На «зовнішньому» шарі знаходяться 2 сервіси 1. HTTP сервер – передає користувачу файли веб сторінки, оброблює HTTP API запити, надає доступ до файлів збережених у додатку та доступ до повного функціоналу додатку через веб-сторінку. 2. Telegram бот – надає користувачам можливість отримати файли, завантажити їх та редагувати теги та опис постів.	API Controller DB Controller в обхід API Controller для доступу до файлів.

API Controller	Виконує обробку API запитів всередині додатку, додатково виконує контроль за сесіями користувачів та доступом до певних функцій різних користувачів	DB Controller
DB Controller	Виконує запити до БД за запитами API	Physical DB file

Справжніх варіантів використання дуже багато, якщо описувати досягнення тієї чи іншої цілі певною послідовністю. Основним використанням є перегляд постів: відкрити сайт, відкрити бажаний пост, повторити.

Безпосередньо застосунок можна умовно поділити на певні модулі. Була побудована діаграма компонентів (рис 2.3) додатку для декількох модулів.

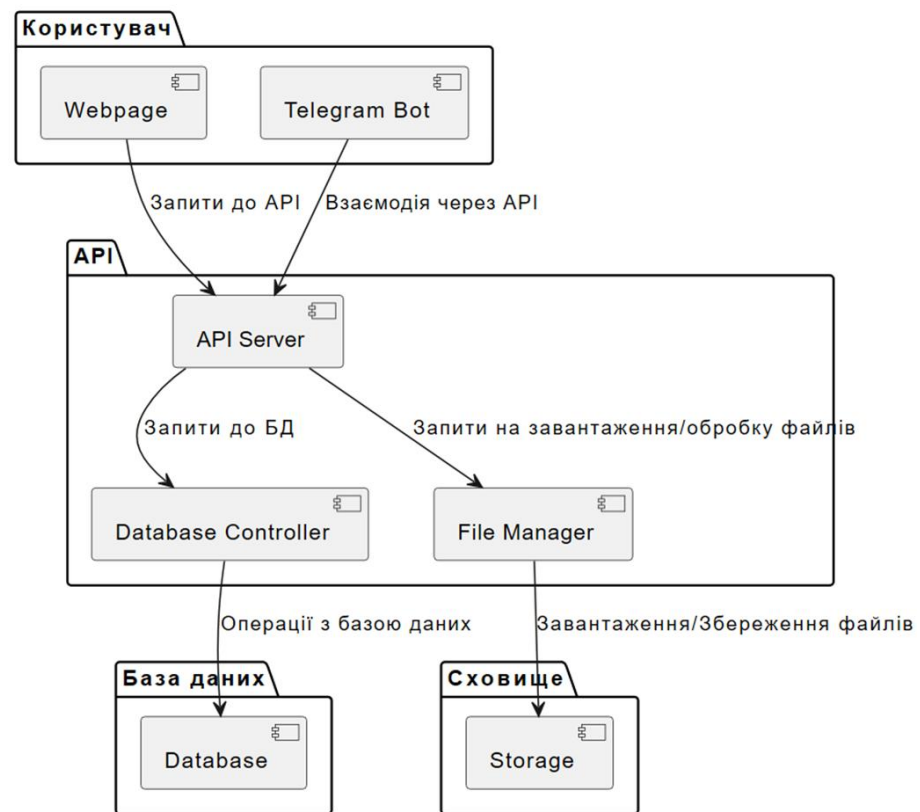


Рис 2.3 Діаграма компонентів додатку

Діаграма зображує архітектуру додатку, де взаємодіють кілька ключових компонентів. На верхньому рівні є елементи, такі як веб-сторінка та Telegram-бот, які слугують точками входу для користувачів. Вони спілкуються із системою через API, надсилаючи запити до API-сервера. Цей сервер є центральним вузлом, який обробляє вхідні запити та спрямовує їх до відповідних модулів усередині системи.

API-сервер може надсилати запити до двох основних компонентів: контролера бази даних та менеджера файлів. Контролер бази даних відповідає за взаємодію з базою даних, виконуючи операції, пов'язані з обробкою даних, наприклад, збереження чи отримання інформації. Менеджер файлів, у свою чергу, управляє операціями, пов'язаними з файлами, такими як їх завантаження чи збереження, і взаємодіє зі сховищем, де ці файли фізично зберігаються.

База даних і сховище виступають як основні ресурси для зберігання інформації в системі. База даних використовується для структурованих даних, з якими працює контролер бази даних, а сховище призначене для файлів, які обробляє менеджер файлів. Таким чином, система забезпечує чіткий розподіл обов'язків: API-сервер координує запити, контролер бази даних і менеджер файлів виконують спеціалізовані завдання, а база даних і сховище забезпечують надійне зберігання даних і файлів для клієнтів, які взаємодіють через веб-сторінку або Telegram-бот.

Детальніше розглянемо алгоритм роботи модуля для створення постів. Для цього була побудована діаграма послідовності(рис. 2.4).

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		33

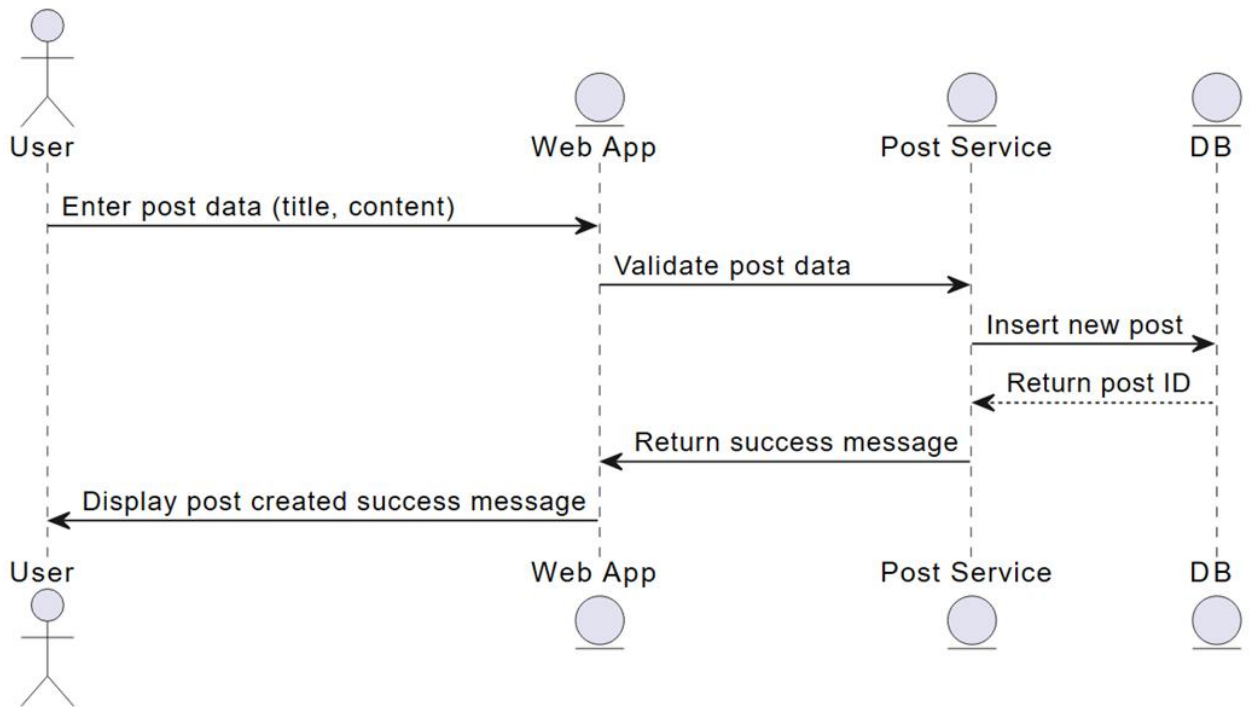


Рис 2.4 Діаграма послідовності модуля створення постів.

Діаграма послідовності для модуля створення постів зображує процес взаємодії між ключовими компонентами системи, які беруть участь у створенні нового посту. Вона ілюструє повний цикл від моменту, коли користувач ініціює запит, до отримання підтвердження про успішне збереження посту. Учасниками цього процесу є користувач, веб-додаток, сервіс постів і база даних. Кожен із цих компонентів відіграє свою роль у логічній послідовності подій, що забезпечує створення посту.

Процес створення посту розпочинається з того, що користувач надсилає дані посту до веб-додатку. Ці дані зазвичай включають заголовок і вміст посту. Наприклад, користувач заповнює форму на веб-сторінці та натискає кнопку "Відправити", що генерує HTTP-запит (найчастіше POST-запит) до веб-додатку. Цей запит є точкою входу в систему і запускає подальшу обробку.

Веб-додаток отримує запит від користувача і виконує первинну обробку даних. На цьому етапі система перевіряє, чи відповідають введені дані певним критеріям. Наприклад, веб-додаток може переконатися, що заголовок не є порожнім, а вміст не перевищує максимально допустиму довжину. Якщо валідація проходить успішно, веб-додаток передає валідовані дані до сервісу

постів для подальшої обробки. У разі, якщо дані не проходять валідацію, веб-додаток може повернути користувачеві повідомлення про помилку з проханням виправити введену інформацію.

Сервіс постів, який є частиною бекенду системи, отримує дані від веб-додатку. Цей компонент відповідає за виконання бізнес-логіки, пов'язаної з постами. Наприклад, сервіс може згенерувати унікальний ідентифікатор для нового посту, додати мітку часу створення або виконати інші необхідні операції. Після підготовки даних сервіс постів формує запит до бази даних для збереження нового запису. Цей запит зазвичай включає всі необхідні поля посту, такі як заголовок, вміст, ідентифікатор і час створення.

База даних отримує запит від сервісу постів і виконує операцію вставки нового посту. Якщо збереження проходить успішно, база даних повертає підтвердження, яке може включати, наприклад, унікальний ідентифікатор щойно створеного посту. У разі виникнення проблем, таких як помилка з'єднання або порушення унікальності, база даних може повернути повідомлення про помилку, яке потім буде оброблено сервісом постів і передано далі.

Після отримання підтвердження від бази даних сервіс постів надсилає повідомлення про успішне виконання операції назад до веб-додатку. Це повідомлення може супроводжуватися кодом статусу HTTP, наприклад, 201 (Created), а також додатковими даними, такими як ідентифікатор нового посту. Веб-додаток, у свою чергу, отримує цю інформацію і формує відповідь для користувача. Користувач бачить результат операції, наприклад, повідомлення на екрані типу "Пост успішно створено" або перенаправлення на сторінку з новим постом.

2.4 Реалізація фото-відео форуму.

Розглянемо бізнес логіку одного з модулів системи, а саме модуля створення постів. Робота з модулем починається з завантаження файлу через веб-інтерфейс.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

Процес створення посту розпочинається, коли користувач завантажує файл через веб-інтерфейс. У файлі index.js налаштований Express-сервер, який обробляє POST-запит, для цього створені модулі(рис 2.5-2.9):

```
const storage = multer.diskStorage({
  destination: (req, file, cb) => {
    cb(null, './storage/file_storage');
  },
  filename: (req, file, cb) => {
    const ext = path.extname(file.originalname);
    cb(null, `WEB-${sysController.hashGen(10)}-${Date.now()}${ext}`);
  }
});
```

Рис 2.5 Метод мультера

```
const fileFilter = (req, file, cb) => {
  const ext = path.extname(file.originalname).toLowerCase();
  const allowedImageExtensions = ['.jpg', '.jpeg', '.png', '.webp', '.gif'];
  const allowedVideoExtensions = ['.mp4', '.mov', '.avi', '.mkv'];
  const allowedAudioExtensions = ['.mp3', '.ogg', '.wav', '.flac']

  if (allowedImageExtensions.includes(ext)) {
    req.fileSizeLimit = 50 * 1024 * 1024;
  } else if (allowedVideoExtensions.includes(ext)) {
    req.fileSizeLimit = 5 * 1024 * 1024 * 1024;
  } else if (allowedAudioExtensions.includes(ext)) {
    req.fileSizeLimit = 25 * 1024 * 1024
  } else {
    return cb(new Error('Unsupported filetype'), false);
  }
  cb(null, true);
};
```

Рис 2.6 Метод фільтру

```
const upload = multer({
  storage: storage,
  fileFilter: fileFilter
}).single('file')
```

Рис 2.7 Налаштування завантаження файлу

```

const checkUserPermissionUpload = async (req, res, next) => {
  const userKey = req.headers['user-key'];
  if (!userKey) {
    return res.send('e/Access rejected.');
```

Рис 2.8 Перевірка прав доступу

```

app.post('/upload', checkUserPermissionUpload, (req, res, next) => {
  upload(req, res, async () => {
    if (!req.file) {
      res.send('e/No file.');
```

Рис 2.9 Обробка запиту на завантаження

Розглянемо наступний етап (рис 2.10 – 2.12), після завантаження файлу на сервер система обробляє його залежно від типу, який визначається за розширенням: для зображень отримуються метадані (ширина, висота, розмір), для аудіо — тривалість і бітрейт, а для відео(Додаток А) — метадані з можливою конвертацією у стандартний формат; потім визначається логін користувача, створюється пост у базі даних і додаються системні теги залежно від характеристик файлу, таких як роздільна здатність чи наявність звуку.

```

module.exports = (filePath, sessionData) => {
  return new Promise(async resolve => {
    try {
      const ext = path.extname(filePath).toLowerCase();
      const save_file_name = path.parse(filePath).name;
      let finalVideoPath = filePath;
```

Рис 2.10 Визначення типу файлу

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

```

if (['.jpg', '.jpeg', '.png', '.webp', '.gif'].includes(ext)) {
  const metadata = await sharp(filePath).metadata();
  const fileSize = { x: metadata.width, y: metadata.height, weight: fs.statSync(filePath).size };

  const dbuser = (await dbinteract.getUserBySessionData(sessionData.type, sessionData.key)).user.login;
  const result = await dbinteract.createPost(save_file_name + ext, fileSize, dbuser);

  const systemTags = [];
  if (fileSize.y >= 2160) systemTags.push('absurd_res');
  else if (fileSize.y >= 1080) systemTags.push('high_res');
  systemTags.push(ext === '.gif' ? 'gif' : 'image');

  await dbinteract.updateTags(result.id, systemTags);
  resolve(result.rslt === 's'
    ? new Response('s', `Post #${result.id} created`, { postID: result.id })
    : { rslt: 's', msg: `Error: ${result.msg}` });
  return;
}

```

Рис 2.11 Обробка зображень

```

if (['.mp3', '.ogg', '.wav', '.flac'].includes(ext)) {
  const mm = require('music-metadata');
  const metadata = await mm.parseFile(filePath);
  const stats = fs.statSync(filePath);

  const fileInfo = {
    duration: metadata.format.duration,
    bitrate: metadata.format.bitrate,
    weight: stats.size
  };

  const dbuser = (await dbinteract.getUserBySessionData(sessionData.type, sessionData.key)).user.login;
  const result = await dbinteract.createPost(save_file_name + ext, fileInfo, dbuser);

  const systemTags = ['audio'];
  if (fileInfo.bitrate >= 192000) systemTags.push('high_quality');
  if (metadata.common.genre) systemTags.push(...metadata.common.genre.split(/\s*,\s*/));

  await dbinteract.updateTags(result.id, systemTags);
  resolve(result.rslt === 's'
    ? new Response('s', `Audio Post #${result.id} created`, { postID: result.id })
    : { rslt: 's', msg: `Error: ${result.msg}` });
  return;
}

```

Рис 2.12 Обробка аудіо

Наступний етап – створення запису в БД. Після обробки файлу система створює запис у базі даних, зберігаючи інформацію про пост, таку як розмір файлу, його ім'я, час створення та автора, а після успішного збереження повертається унікальний ідентифікатор посту для подальшого використання.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		38

```

module.exports = (db, fileName, fileSize, user) => {
  return new Promise(async resolve => {
    db.run(`INSERT INTO posts('size', 'file', 'timestamp', 'author') VALUES(?, ?, ?, ?)`,
      [JSON.stringify(fileSize), fileName, Date.now(), user],
      (err) => {
        if (err) {
          resolve(new sysController.createResponse(
            '',
            '',
            {},
            err,
            'Error creating post'
          ))
        } else {
          db.get(`SELECT * FROM posts WHERE file = '${fileName}'`, (err, row) => {
            resolve(new sysController.createResponse(
              's',
              'Post created, heres ID',
              { id: row.id },
              err,
              'Post created, heres ID'
            ))
          })
        }
      })
  })
}

```

Рис 2.13 Запис даних в БД

Після створення посту система автоматично призначає йому набір заздалегідь визначених тегів. Ці теги оновлюються в записі посту в базі даних, де вони зберігаються у форматі JSON у полі 'tags'. Такий процес гарантує, що кожен пост належним чином позначений для подальшого використання та пошуку.

```

module.exports = (db, postID, tags) => {
  return new Promise(async resolve => {
    db.run(`UPDATE posts SET tags = ? WHERE id = ?`,
      [JSON.stringify(tags), postID],
      (err) => {
        resolve(new SysController.createResponse(
          's',
          `Updated tags for post ${postID}`,
          {},
          err,
          `Error updating tags for post ${postID}`
        ))
      })
  })
}

```

Рис 2.14 Оновлення тегів

Для визначення автора посту система використовує спеціальний механізм, який знаходить користувача на основі сесійних даних. Процес включає пошук сесії за її типом і ключем, після чого з бази даних отримується відповідна інформація про користувача. Таким чином, система ідентифікує автора, пов'язуючи сесійні дані з обліковим записом користувача, що забезпечує коректне визначення його особи без необхідності уточнення технічних деталей реалізації.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

```

module.exports = (db, type, key) => {
  return new Promise(async resolve => {
    db.get(`SELECT * FROM sessions WHERE type = ? AND key = ?`,
      [type, key],
      (err, row) => {
        if (err) {
          resolve(new sysController.createResponse(
            'e',
            `Error getting session data: ${err}`,
          ))
        } else {
          let login = ''
          if (row) login = row.user
          db.get(`SELECT * FROM users WHERE login = ?`, [login],
            (err, row) => {
              if (!row) {
                resolve(new sysController.createResponse(
                  'w',
                  'No user for this session data',
                  { user: null }
                ))
                return
              }
              row.favs = JSON.parse(row.favs)
              row.likes = JSON.parse(row.likes)
              row.dislikes = JSON.parse(row.dislikes)
              row.blacklist = JSON.parse(row.blacklist)
              row.usersettings = JSON.parse(row.usersettings)
              resolve(new sysController.createResponse(
                's',
                `Got user with session data`,
                { user: row },
                err,
                "Error getting user by session data"
              ))
            }
          )
        }
      }
    )
  })
}

```

Рис 2.15 Визначення автора посту

Усі операції з базою даних (пошук користувача, створення посту, додавання міток) проходять через спеціальний модуль, який управляє запитами. Він ставить запити в чергу і виконує їх по черзі, щоб уникнути проблем, якщо багато користувачів одночасно завантажують файли.

```

const queryQueue = [];
let isProcessing = false;
const DBProcessor = async (action, ...args) => {
  try {
    const request = require(`./${action}.js`);
    const requestResult = await request(db, ...args);
    return requestResult;
  } catch (err) {
    console.log(err);
    const ERRdata = new responseConstructor(
      'e',
      '',
      {},
      err.code,
      `Error executing ${action} DB request`
    );
    consoleLogger(`${ERRdata.rslt}/${ERRdata.msg}`, [{ txt: 'DB', txtc: 'red', txtb: 'white' }]);
    return ERRdata;
  }
};

const processQueue = async () => {
  if (isProcessing) return;
  isProcessing = true;
  while (queryQueue.length > 0) {
    const { action, args, resolve } = queryQueue.shift();
    try {
      const result = await DBProcessor(action, ...args);
      resolve(result);
    } catch (err) {
      resolve(err);
    }
  }
  isProcessing = false;
};

const queueRequest = (action, ...args) => {
  return new Promise((resolve) => {
    queryQueue.push({ action, args, resolve });
    processQueue();
  });
};

module.exports = new Proxy({}, {
  get: (target, prop) => {
    return (...args) => {
      return queueRequest(prop, ...args);
    };
  }
});

```

Рис 2.16 Операції з БД

Висновки до другого розділу

Під час проєктування фото-відео форуму були виявлені основні варіанти використання системи відносно наявних прав користувача, завдяки чому було спроектовано архітектуру системи. Для забезпечення чіткого розділення логіки презентації та бізнес-логіки були продумані способи комунікації між модулями, зокрема через використання спеціальних контролерів і чергування запитів до бази даних.

В даному розділі була розроблена структура бази даних відповідно до бізнес вимог до системи, яка зберігання та доступ до необхідних даних. База даних має 8 основних таблиць, а загальний обсяг 11 таблиць, в яких зберігається відповідна інформація про сутності системи, яка забезпечує потрібна відповідного виконання бізнес логіки.

В останньому пункті детально розглядається код, який є частиною модулю створення постів, який відповідає за завантаження файлів, їхню обробку залежно від типу, створення записів у базі даних та додавання тегів, що гарантує коректне функціонування модуля створення постів.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З СИСТЕМОЮ ДОДАТКУ

3.1 Порядок встановлення та налаштування системи

Діаграма розгортання виступає як візуальне уявлення конфігурації вузлів обробки та компонентів системи під час їх встановлення. Це один із типів структурних діаграм, які використовуються для моделювання різних фізичних аспектів об'єктно-орієнтованої системи.

На (рис. 3.1) зображена діаграма розгортання, на якій показано конфігурацію вузлів та компонентів системи під час розгортання.

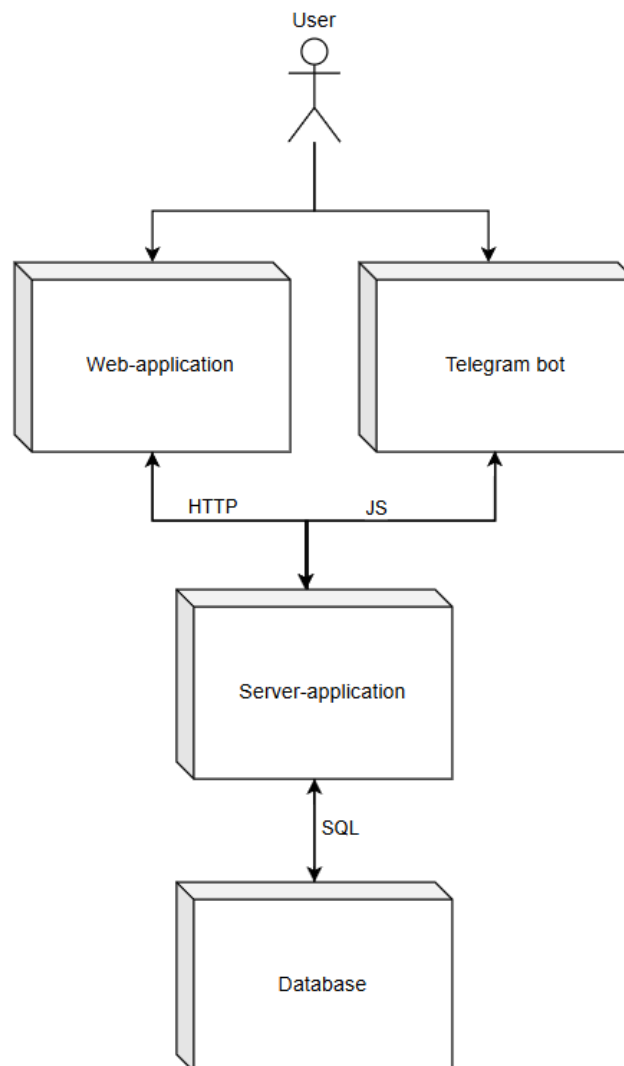


Рис 3.1 Діаграма розгортання додатку

Опишемо діаграму розгортання додатку: користувач взаємодіє з системою через веб-додаток або Telegram-бот, які виступають клієнтською частиною, забезпечуючи зручний інтерфейс для введення даних і їхньої візуалізації. Ці компоненти комунікують із серверною частиною за допомогою протоколів HTTP, передаючи запити та отримуючи відповіді, що дозволяє користувачу ефективно взаємодіяти із системою незалежно від платформи, яку він обирає — веб-браузер чи месенджер.

Серверна частина, розміщена на серверному додатку, відіграє ключову роль у процесі розгортання, забезпечуючи обробку запитів, які надходять від клієнтських інтерфейсів. Вона відповідає за реалізацію бізнес-логіки, обробку даних, їхню валідацію та координацію з іншими компонентами системи. Цей рівень забезпечує безперервність роботи, обробляючи вхідні дані в реальному часі та готуючи їх до подальшого зберігання, що робить серверний додаток центральним вузлом усієї архітектури.

База даних, як завершальний елемент розгортання, взаємодіє з серверною частиною через SQL-протокол, виступаючи надійним сховищем для всіх оброблених даних. Вона зберігає інформацію, отриману від користувачів через веб-додаток чи Telegram-бот, забезпечуючи її доступність і цілісність для подальшого використання. Така структура дозволяє системі ефективно управляти великими обсягами даних, підтримуючи стабільну роботу додатку навіть при значному навантаженні.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		45

3.2 Структура інтерфейсу та порядок взаємодії з фото-відео форумом

Продемонструємо інтерфейс системи на прикладі основного сценарію використання додатку. Коли користувач входить у систему, його зустрічає головна сторінка з постами, де він може побачити існуючі пости або перейти на інші сторінки такі як колекції, створення посту, чат, профіль і адмін-панель.

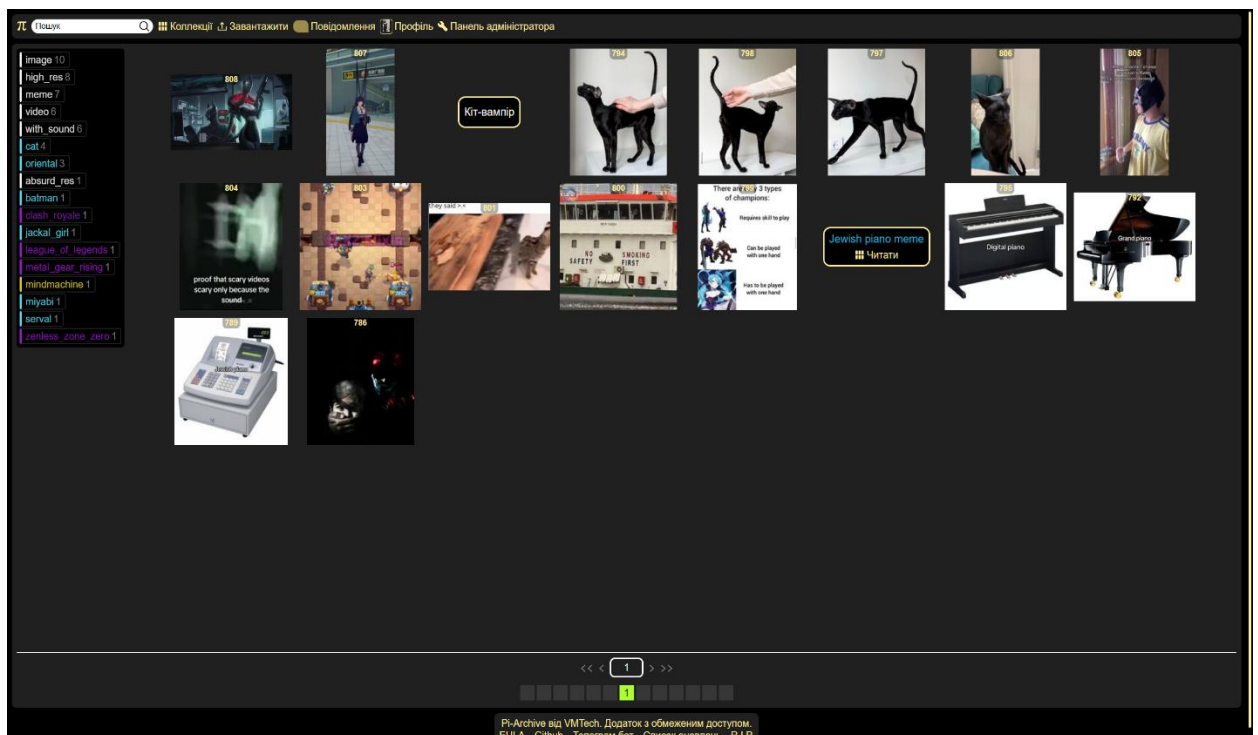


Рис 3.2 Головна сторінка

При виборі конкретного посту користувач переходить на сторінку детального перегляду, де відображаються зображення або відео, інформація про розмір файлу, теги та статус завантаження, ймовірні дії з цим постом (завантаження, обрати зображення як аватар профілю, отримати пост через телеграм-бота, редагування тегів та опису, додавання посту до групу або ж видалення посту), а також присутня можливість залишати коментарі під цим постом(рис. 3.3).

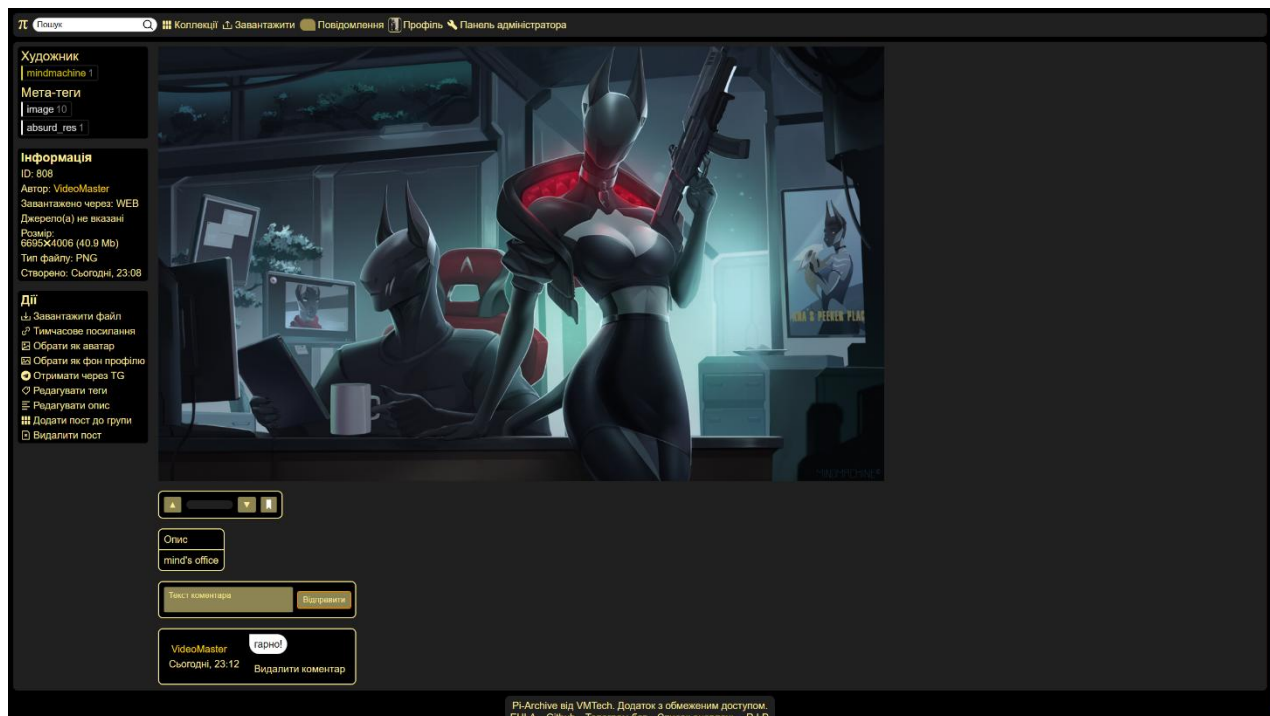


Рис 3.3 Сторінка перегляду посту

При переході на сторінку колекцій користувач може побачити всі існуючі колекції (рис. 3.4).

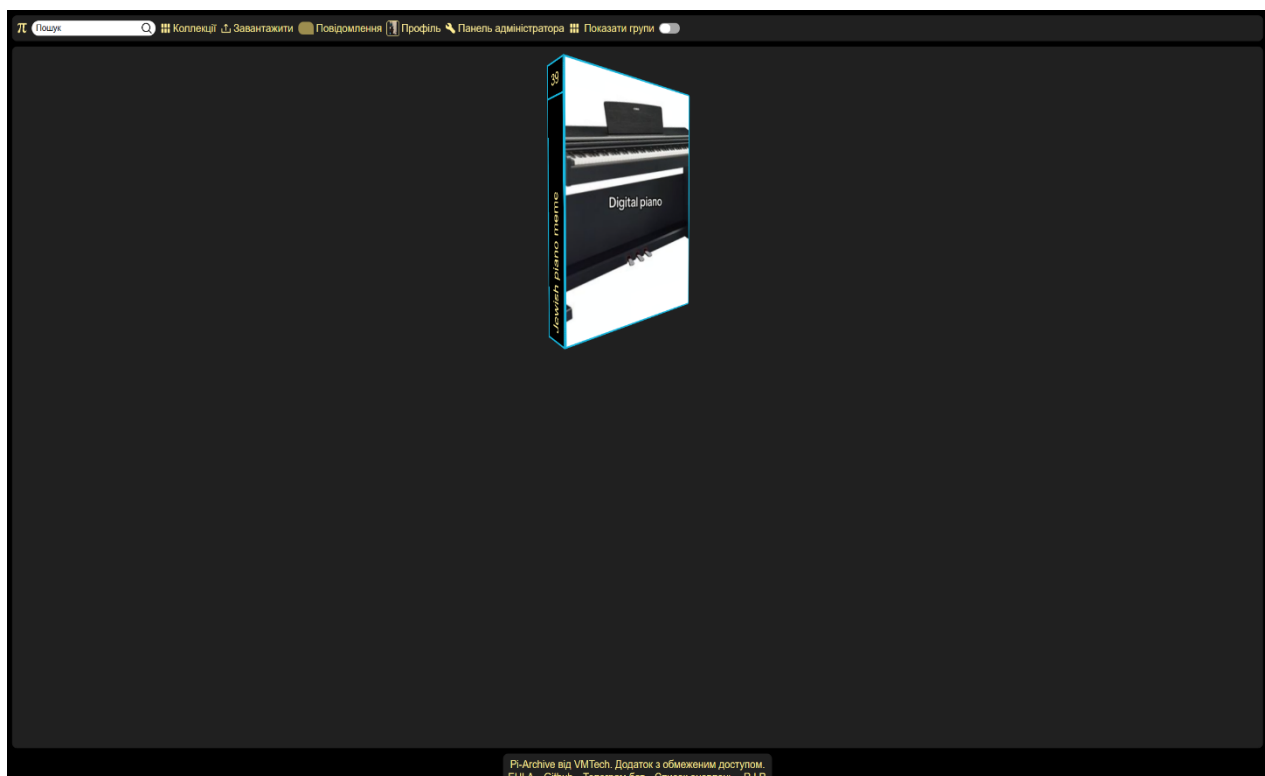


Рис 3.4 Сторінка колекцій

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк. 47
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		

При виборі конкретної колекції користувач переходить на сторінку детального перегляду, де відображаються колекція зображень, інформація про колекцію, ймовірні дії з цією колекцією (редагування тегів та колекції)(рис. 3.5).

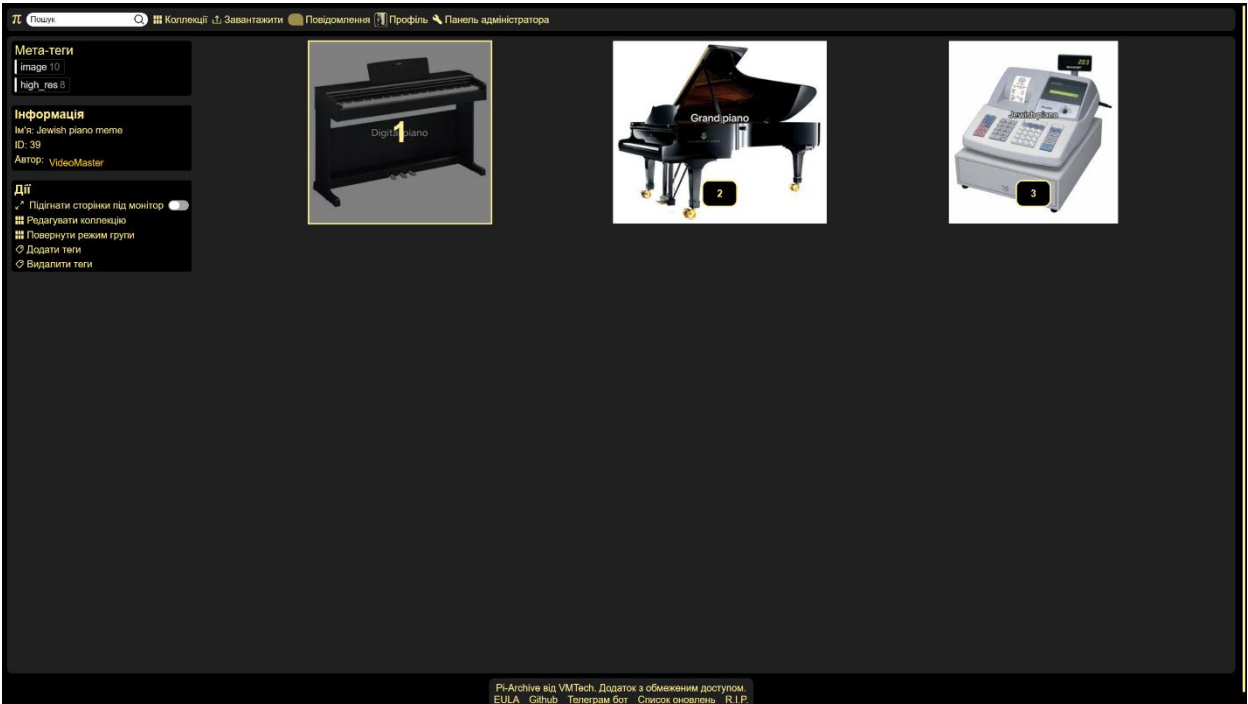


Рис 3.5 Сторінка перегляду колекцій



Рис 3.6 Сторінка перегляду фото

Під час взаємодії з колекціями користувач може редагувати теги та колекції за допомогою меню редагування(рис 3.7-3.8).

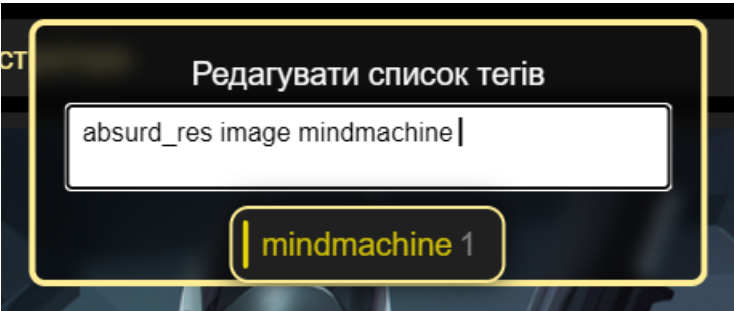


Рис 3.7 Меню редагування списку тегів

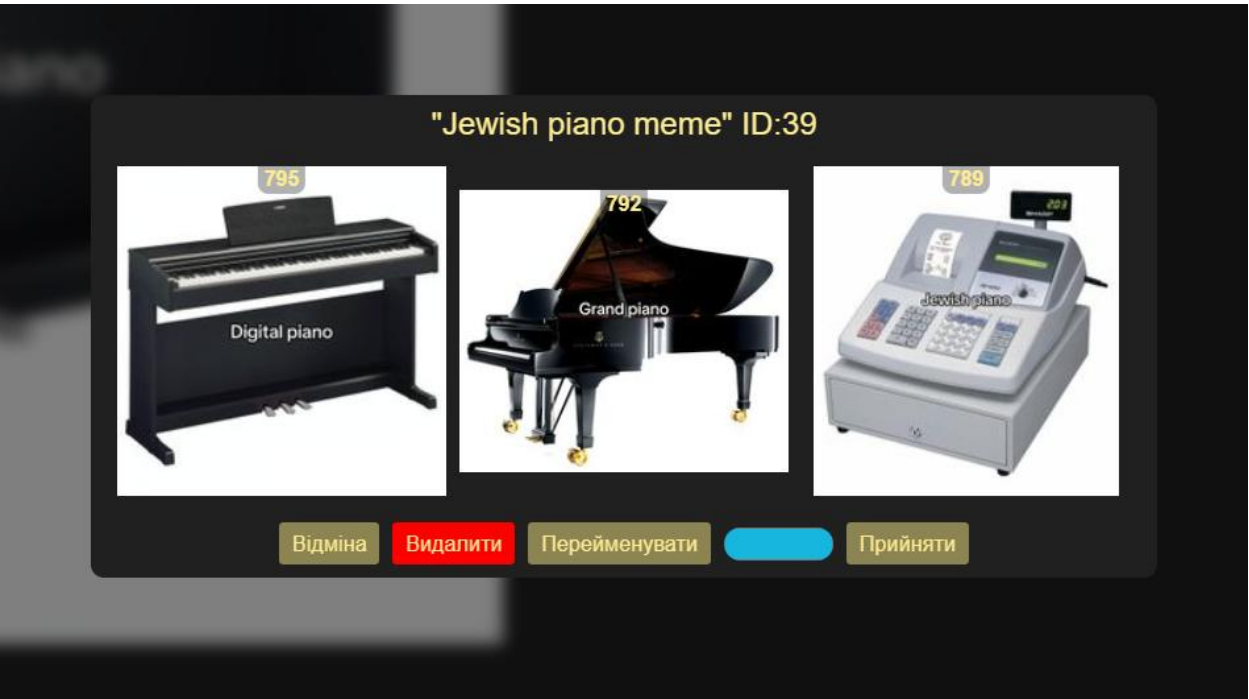


Рис 3.8 Меню редагування колекції

На головній сторінці та сторінці колекцій користувач може оцінювати як пости, так і колекції постів.(рис. 3.9-3.10)

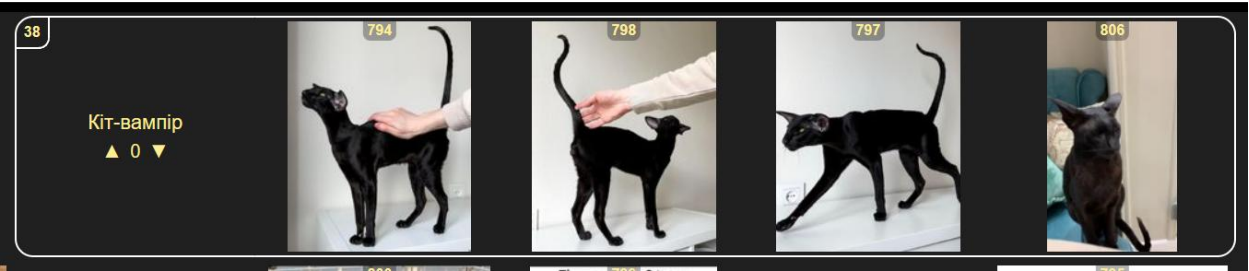


Рис 3.9 Відображення групи

Важливим елементом інтерфейсу є пошук. При використанні пошуку користувач бачить сторінку з автодоповненням, де відображаються відповідні пости, із підсвіткою знайдених збігів для швидкого доступу (рис. 3.10).

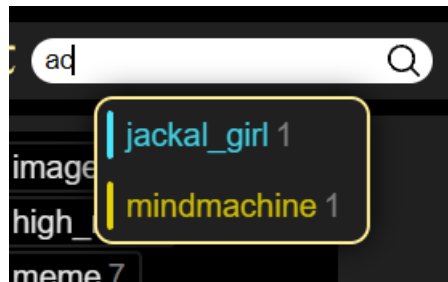


Рис 3.10 Пошук по тегам

При пошуку користувач може шукати пости по тегам або ж по анти-тегам(рис. 3.11).

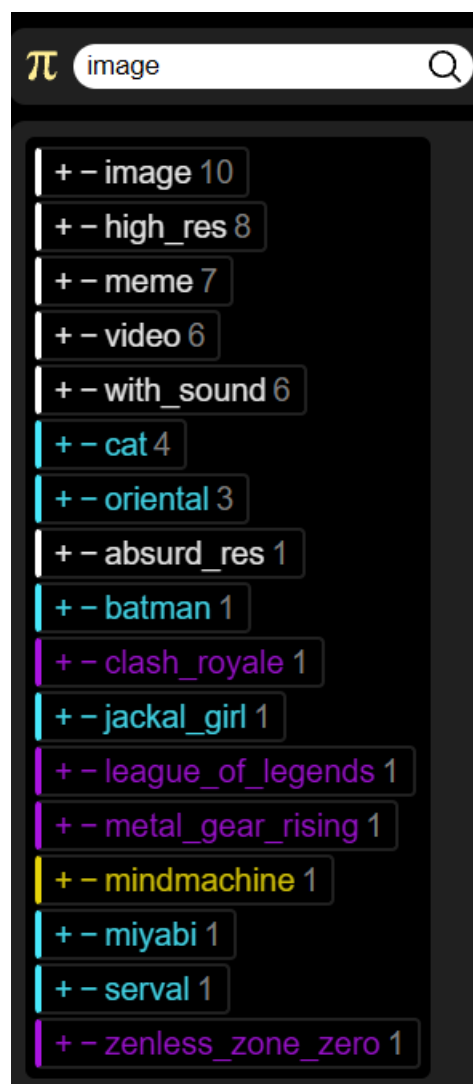


Рис 3.11 Пошук по тегам

При переході на сторінку створення посту користувач бачить вікно з можливістю обрати або перетягнути файли(рис. 3.12).

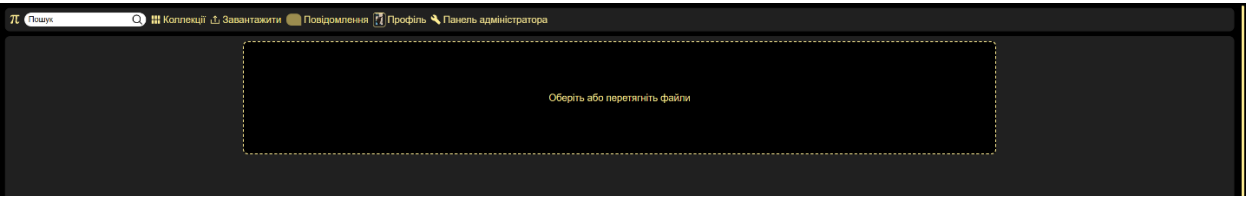


Рис 3.12 Вікно для завантаження файлів

Після перенесення або обрання файлу користувач переходить на сторінку створення посту де він може додати теги та опис для створюваного посту(рис. 3.13).

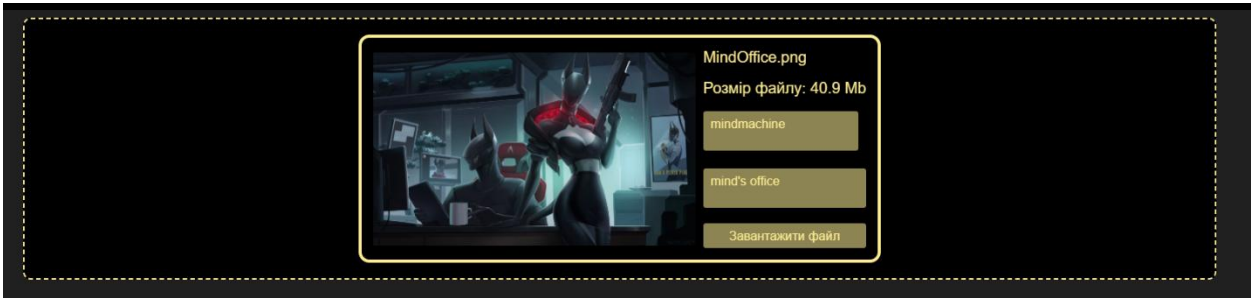


Рис 3.13 Вікно створення посту

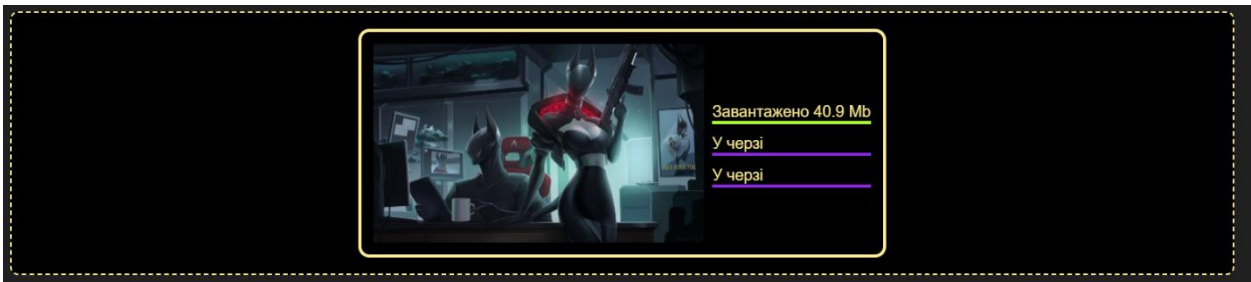


Рис 3.14 Перебіг створення посту

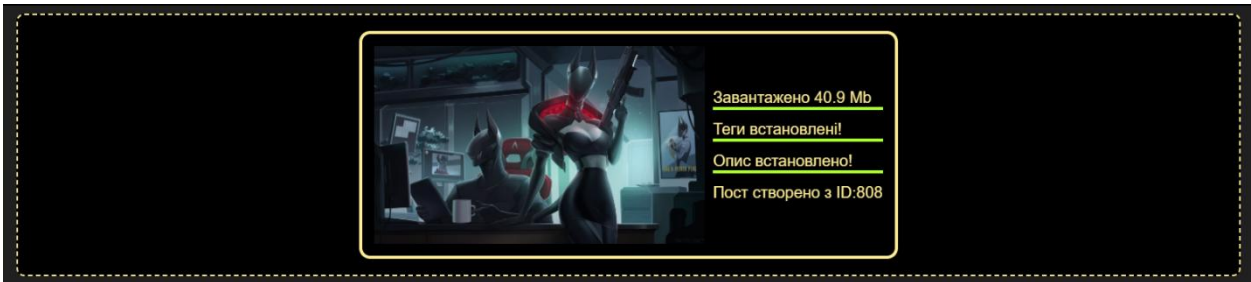


Рис 3.15 Вдале створення посту

3.3 Тестування роботи програмного продукту

При розробці, кожен програмний продукт містить певні явні та неявні дефекти які потрібно виправляти. Для створення умов наближених до реальності, в рамках роботи з API для тестування певних запитів був використаний Postman. Структура запитів реалізована нетипово, усі запити передаються через APIController який є частиною фронтенду, та надає уніфікований спосіб отримувати інформацію клієнтам. У запиті передається об'єкт що складається з параметру «action» який відповідає за запитувану дію на бекенді, «user» де надається ключ сесії та тип сесії актуального клієнту, «request» в собі несе усі дані потрібні для виконання запиту. Сервер не вміє вірізняти помилки запитів у кодах відповіді, тому усе що йде на API повертається з 200 кодом, але може отримати в тілі відповіді помилку та опис помилки.

На рисунку 3.16 продемонстрована робота запиту на логін, передані усі дані, в особливості – захешований пароль користувача. Сервер у свою чергу повернув у відповідь успішний результат та усі налаштування користувача для їх прийняття на клієнті при авторизації.

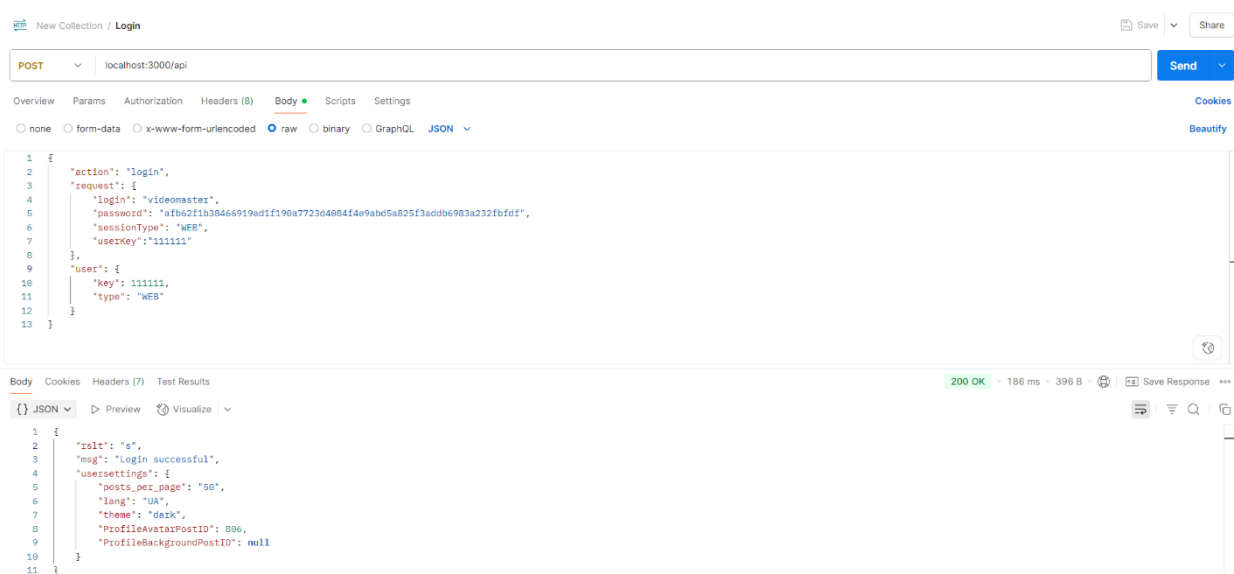


Рис. 3.16

		Ігнатів О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		52

На рисунку 3.17 продемонстрований запит без ввідних даних, але сервер повернув результат з списком сесій користувача.

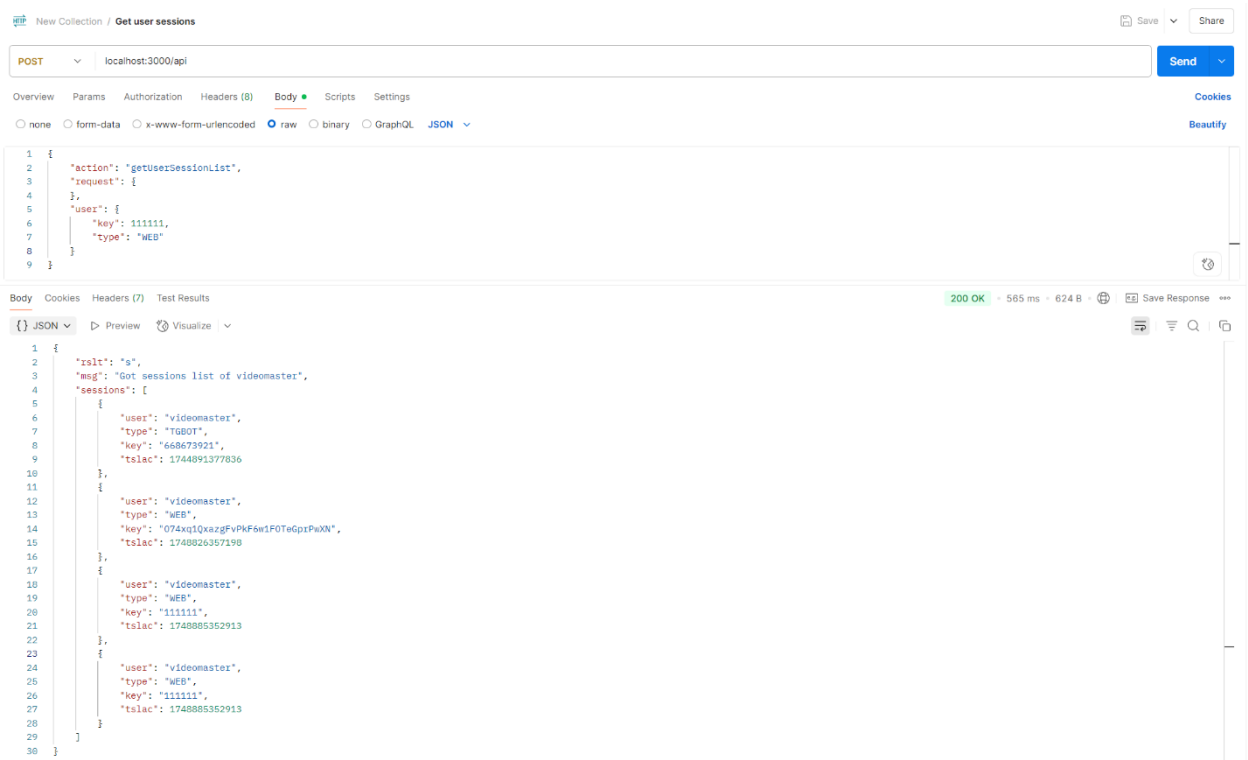


Рис. 3.17

На рисунку 3.18 показано яка кількість даних пов'язана з одним користувачем, у тому числі різні списки та інформація що повинна бути відображена на сторінці профілю.

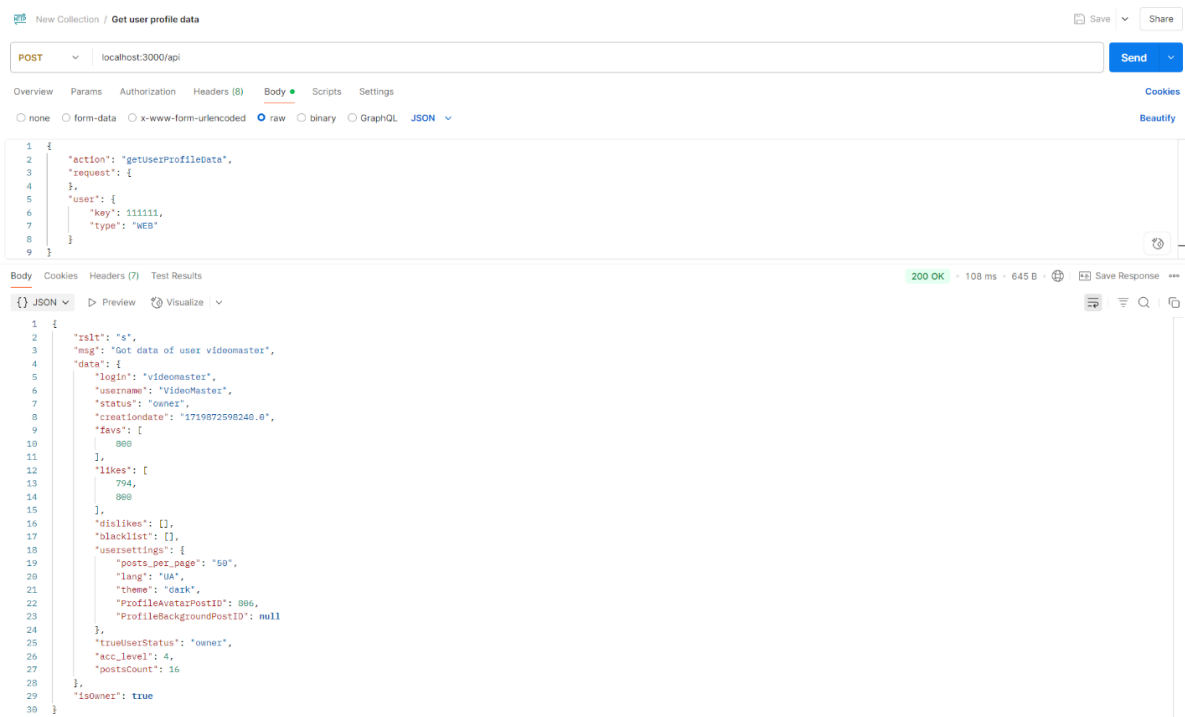


Рис. 3.18

На рисунку 3.19 відображено використання роуту отримання постів для отримання постів у великій кількості, на основі налаштувань пагінації чи певної кількості основуючись на певній змінній, також надано пустий текст запиту.

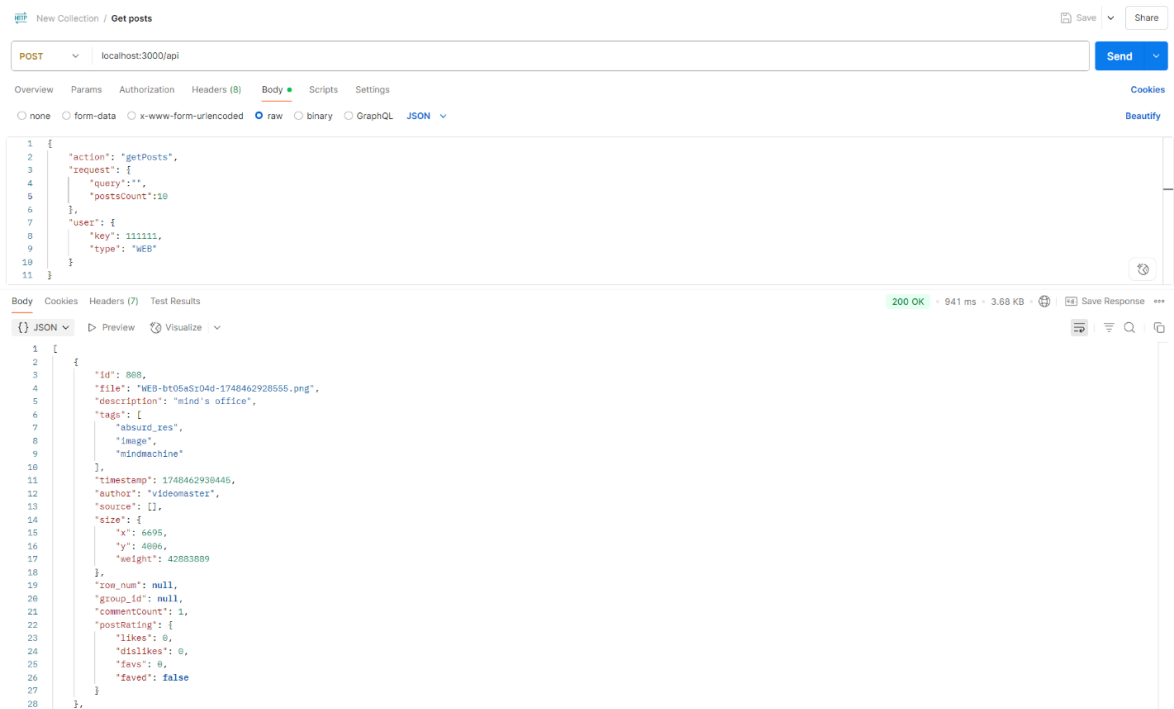


Рис. 3.19

На рисунку 3.20 продемонстроване отримання спрощених даних посту.

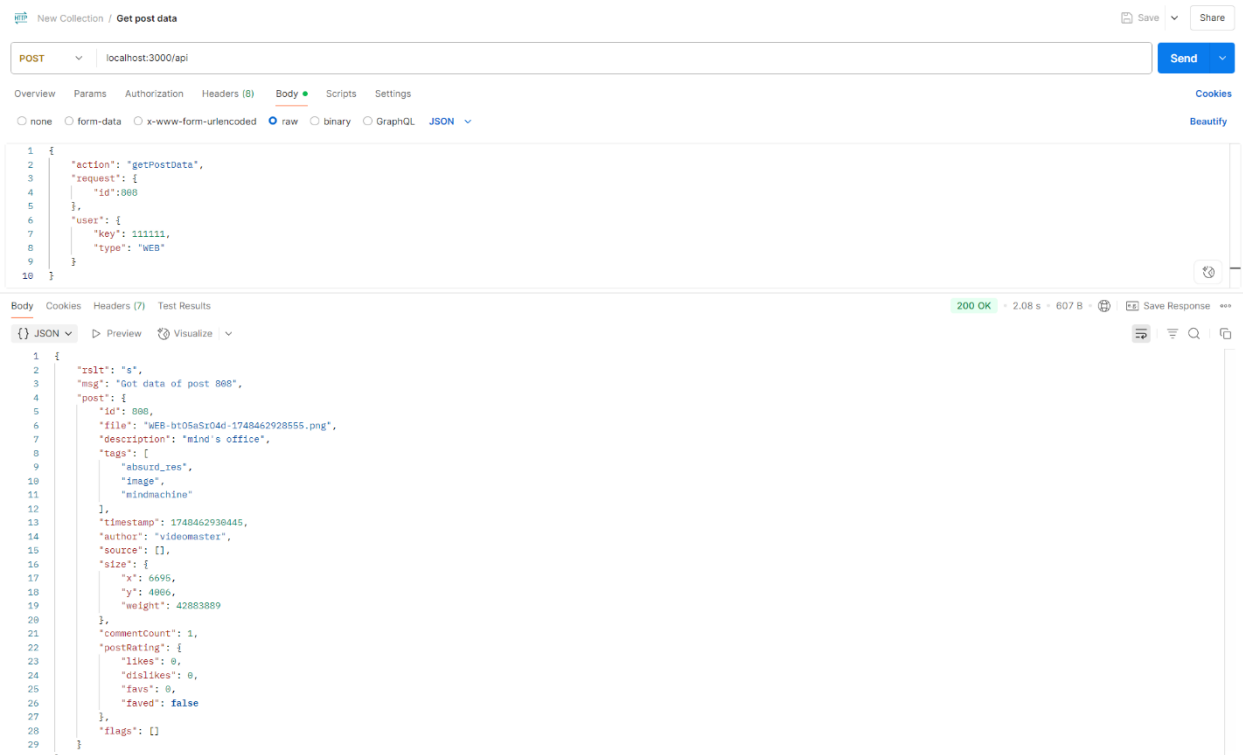


Рис 3.20

На рисунку 3.21 продемонстроване отримання списку тегів для відображення на сторінці пошуку, також цей роут використовується для отримання тегів на сторінці з постом та колекціями.

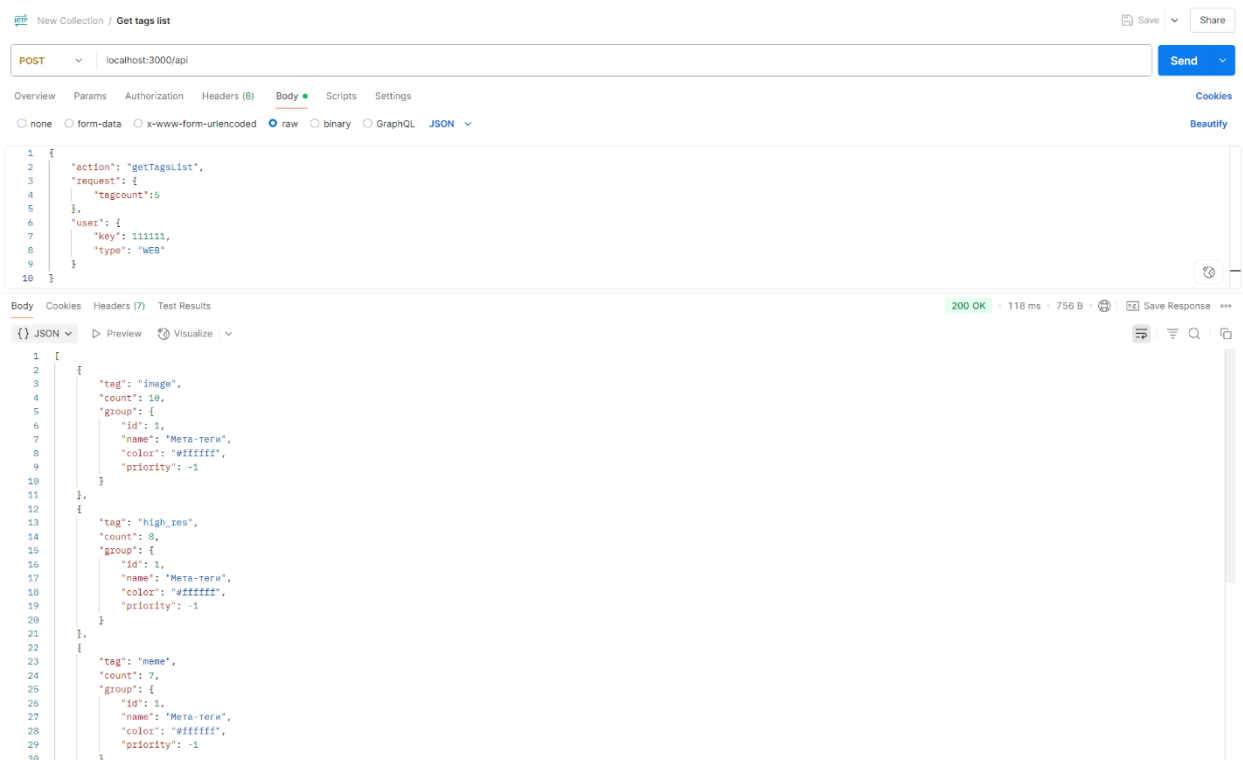


Рис. 3.21

Висновки до третього розділу

В даному розділі було продемонстровано використання Postman для тестування певних ендпоінтів програми. Наведений детальний опис незвичайної роботи API додатку в якому описано використання тіла запиту для передавання інформації та вибору ендпоінту з передаванням даних користувача. За результатами тестування можна зробити висновок що ендпоїнти функціонують коректно.

ВИСНОВКИ

Під час виконання випускної кваліфікаційної роботи бакалавра було спроектовано та розроблено фото-відео форум, проведено дослідження предметної області і ринку, з порівняльним аналізом аналогів фото-відео форумів. На основі результатів дослідження та аналізу були визначені вимоги для розроблювальної системи.

Проектування фото-відео форуму включало аналіз варіантів використання програмного продукту та проектування структури проєкту, включаючи моделювання взаємодії між складовими частинами проєкту. Також було спроектовано базу даних для зберігання інформації.

Продемонстровано та описано основний потік роботи з системою. Розглянуто основний функціонал, який включає перегляд, створення та редагування постів і колекцій.

Була надана детальна схема розгортання фото-відео форуму, описано взаємодію структурних елементів системи та наведено опис інтерфейсу розробленого програмного продукту. Система має доволі легкий для сприйняття та освоєння інтерфейс на відміну від аналогів.

Результатом кваліфікаційної роботи є готовий до експлуатації фото-відео форум, який відповідає сучасним вимогам користувачів та забезпечує зручний і функціональний інтерфейс для роботи з фото та відео контентом. Розроблена система дозволяє ефективно публікувати, переглядати та редагувати пости й колекції, а також забезпечує стабільне зберігання даних завдяки спроектованій базі даних. У порівнянні з аналогами, форум вирізняється простотою використання та інтуїтивно зрозумілим дизайном, що сприяє швидкому освоєнню користувачами. У перспективі можливе розширення функціоналу системи, зокрема додавання інструментів для спільної роботи, інтеграції з соціальними мережами та впровадження нових технологій для підвищення продуктивності й зручності використання.

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Node.js. Офіційна документація. URL: <https://nodejs.org/en/docs/> (дата звернення: 03.06.2025).
2. Express.js. Офіційна документація. URL: <https://expressjs.com/> (дата звернення: 03.06.2025).
3. SQLite. Офіційна документація. URL: <https://www.sqlite.org/docs.html> (дата звернення: 03.06.2025).
4. FFmpeg. Офіційна документація. URL: <https://ffmpeg.org/documentation.html> (дата звернення: 03.06.2025).
5. Google Photos. Офіційний сайт. URL: <https://photos.google.com/> (дата звернення: 03.06.2025).
6. Flickr. Офіційний сайт. URL: <https://www.flickr.com/> (дата звернення: 03.06.2025).
7. Pinterest. Офіційний сайт. URL: <https://www.pinterest.com/> (дата звернення: 03.06.2025).
8. REST API With NodeJS Using SQLite3 and Express. Medium. URL: <https://medium.com/@codesprintpro/rest-api-using-sqlite3-nodejs-and-expressjs-f8c0c0847fe5> (дата звернення: 03.06.2025).
9. How To Build a Media Processing API in Node.js With Express and FFmpeg.wasm. DigitalOcean. URL: <https://www.digitalocean.com/community/tutorials/how-to-build-a-media-processing-api-in-node-js-with-express-and-ffmpeg-wasm> (дата звернення: 03.06.2025).
10. Writing a Video Encoder using Node JS and FFmpeg. Medium. URL: <https://medium.com/nerd-for-tech/writing-a-video-encoder-using-node-js-and-ffmpeg-b909442472a9> (дата звернення: 03.06.2025).
11. Pixie: A System for Recommending 3+ Billion Items to 200+ Million Users in Real-Time. arXiv. URL: <https://arxiv.org/abs/1711.07601> (дата звернення: 03.06.2025).

		Ігнатов О.С.			ІПЗ.КР.Б – 121 – 25 – ЗПП	Арк.
		Годлевський Ю.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		57

12. Node.js SQLite: Build a simple REST API with Express step-by-step. Reddit. URL: https://www.reddit.com/r/node/comments/qia981/nodejs_sqlite_build_a_simple_rest_api_with/ (дата звернення: 03.06.2025).
13. leafac/sqlite: The best way to use SQLite in Node.js. Reddit. URL: https://www.reddit.com/r/node/comments/mvptoc/leafacsqlite_the_best_way_to_use_sqlite_in_nodejs/ (дата звернення: 03.06.2025).
14. Node express sqlite3. Stack Overflow. URL: <https://stackoverflow.com/questions/73869921/node-express-sqlite3-when-i-use-a-service-for-the-db-query-i-get-nothing-back> (дата звернення: 03.06.2025).
15. Node.js Express FFMPEG Project to Build Video Generator. YouTube. URL: <https://www.youtube.com/watch?v=h4M89sd-k7A> (дата звернення: 03.06.2025).
16. Up and Running with SQLite3 in a NodeJS API. YouTube. URL: https://www.youtube.com/watch?v=_RtpUaBSie0 (дата звернення: 03.06.2025).
17. How to Build a Simple Video Converter with Node.js. LevelUp. URL: <https://levelup.gitconnected.com/how-to-build-a-simple-video-converter-52eaeaf248f1> (дата звернення: 03.06.2025).
18. Node.js - Passing Arguments to Flickr SDK Functions. Flickr. URL: <https://www.flickr.com/groups/51035612836@N01/discuss/72157717368151637/> (дата звернення: 03.06.2025).
19. ffmpeg is really hard, so I created an NPM library called ezffmpeg. Reddit. URL: https://www.reddit.com/r/node/comments/1j5znwm/show_reddit_ffmpeg_is_really_hard_so_i_created_an/ (дата звернення: 03.06.2025).
20. Build a REST API with Node.js SQLite and Express JS. DeveloperHowTo. URL: <https://developerhowto.com/2018/12/29/build-a-rest-api-with-node-js-and-express-js/> (дата звернення: 03.06.2025).

ДОДАТКИ

```

if (!['.mp4', '.mov', '.avi', '.mkv'].includes(ext)) {
  resolve({ rslt: 'e', msg: `Unsupported filetype [${ext}]!` });
  return;
}

ffmpeg.ffprobe(filePath, async (err, metadata) => {
  if (err) {
    log('e/Error executing FFMPEG [metadata]: ' + err.message);
    resolve({ rslt: 'e', msg: `Error executing FFMPEG[metadata]: ${err.message}` });
    return;
  }

  const videoStream = metadata.streams.find(stream => stream.codec_type === 'video');
  const audioStream = metadata.streams.find(stream => stream.codec_type === 'audio');
  const isMp4 = ext === '.mp4';
  const needsConversion = !isMp4 || videoStream.codec_name !== 'h264' || (audioStream && audioStream.codec_name !== 'aac');

  if (needsConversion) {
    const tempPath = path.join(path.dirname(filePath), `${save_file_name}-temp.mp4`);
    await new Promise((resolveConvert, rejectConvert) => {
      ffmpeg(filePath)
        .output(tempPath)
        .videoCodec('libx264')
        .audioCodec('aac')
        .audioChannels(2)
        .format('mp4')
        .outputOptions('-movflags +faststart')
        .on('end', () => {
          log(`s/Video converted: ${tempPath}`);
          fs.unlinkSync(filePath);
          fs.renameSync(tempPath, filePath);
          finalVideoPath = filePath;
          resolveConvert();
        })
        .on('error', (err) => {
          log(`e/Error converting video: ${err.message}`);
          rejectConvert(err);
        })
        .run();
    });

    metadata = await new Promise((resolveMeta, rejectMeta) => {
      ffmpeg.ffprobe(finalVideoPath, (err, newMeta) => {
        if (err) {
          rejectMeta(err);
        } else {
          resolveMeta(newMeta);
        }
      });
    });
  }

  const updatedVideoStream = metadata.streams.find(stream => stream.codec_type === 'video');
  const updatedAudioStream = metadata.streams.find(stream => stream.codec_type === 'audio');

  const fileSize = {
    x: updatedVideoStream.width,
    y: updatedVideoStream.height,
    duration: metadata.format.duration,
    weight: fs.statSync(finalVideoPath).size
  };

  const dbuser = (await dbinteract.getUserBySessionData(sessionData.type, sessionData.key)).user.login;
  const result = await dbinteract.createPost(save_file_name + '.mp4', fileSize, dbuser);

  const systemTags = [];
  if (fileSize.y >= 2160) systemTags.push('absurd_res');
  else if (fileSize.y >= 1080) systemTags.push('high_res');
  if (updatedAudioStream) systemTags.push('with_sound');
  systemTags.push('video');

  await dbinteract.updateTags(result.id, systemTags);

  resolve(result.rslt === 's'
    ? new Response('s', { postID: result.id })
    : { rslt: 's', msg: `Error while processing file: ${result.msg}` });
});

```