

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 «Веб-технології») на тему:
**«Вебплатформа аніме та манги з рекомендаційною системою на
основі користувацьких списків»**

Виконав студент групи ВТ-22-1
ЗАЛОЗНИЙ Олександр Анатолійович

Керівник роботи:
САВІЦЬКИЙ Роман Святославович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

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
«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЗАЛОЗНИЙ Олександр Анатолійович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: «Вебплатформа аніме та манги з рекомендаційною системою на основі користувацьких списків»,

затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., № 92/с**

Вихідні дані для роботи: об'єктом дослідження є процес розробки вебплатформ цифрового медіаконтенту, предметом є методи й засоби створення вебплатформи аніме та манги з рекомендаційною системою та інтелектуальним пошуком.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р. С.	02.03.2026р.	02.03.2026р
2	Савіцький Р. С.	10.04.2026р	10.04.2026р
3	Савіцький Р. С.	22.04.2026р	22.04.2026р

РЕФЕРАТ

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел із 31 найменування та 1 додатку. Робота містить 48 рисунків і 9 таблиць. Повний обсяг роботи становить 97 сторінок, з них 88 сторінок основного тексту.

У роботі спроектовано та реалізовано вебплатформу аніме та манги з рекомендаційною системою на основі користувацьких списків. Проаналізовано предметну область і наявні аналоги, сформульовано постановку задачі та визначено функціональні й нефункціональні вимоги. Обґрунтовано клієнт-серверну архітектуру з гібридним рендерингом на основі Next.js, REST API та СУБД PostgreSQL. Спроектовано базу даних і реалізовано каталогізацію творів, персонажів і причетних осіб, ведення персональних списків, відгуки, коментарі та розмежування доступу за ролями. Розроблено гібридну рекомендаційну систему, що поєднує контентну й колаборативну фільтрацію, інтелектуальний пошук природною мовою та відстеження виходу серій із формуванням сповіщень. Платформу розгорнуто у хмарному середовищі з конвеєром безперервної інтеграції, а її працездатність підтверджено модульним, інтеграційним і навантажувальним тестуванням.

КЛЮЧОВІ СЛОВА: АНІМЕ, ВЕБПЛАТФОРМА, ІНТЕЛЕКТУАЛЬНИЙ ПОШУК, МАНГА, РЕКОМЕНДАЦІЙНА СИСТЕМА.

					ІПЗ.КР.Б.-121-26-ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа аніме та манги з рекомендаційною системою на основі користувацьких списків	Літ.	Арк.	Аркуші
Розробник		Залозний О.А.						
Керівник		Савіцький Р.С.					3	97
Рецензент		Єфремов Ю.М.				Житомирська політехніка, група ВТ-22-1		
Зав. каф.		Вакалюк Т. А.						

ABSTRACT

The qualification work consists of an introduction, three chapters, conclusions, a list of 31 references, and 1 appendix. The work contains 48 figures and 9 tables. The total length of the work is 97 pages, including 88 pages of the main text.

The work designs and implements a web platform for anime and manga with a recommender system based on user lists. The subject area and existing analogues are analyzed, the problem statement is formulated, and functional and non-functional requirements are defined. A client-server architecture with hybrid rendering based on Next.js, a REST API, and the PostgreSQL DBMS is justified. A database is designed, and cataloging of titles, characters, and related people, personal list management, reviews, comments, and role-based access control are implemented. A hybrid recommender system combining content-based and collaborative filtering, a natural-language intelligent search, and episode-release tracking with notification generation are developed. The platform is deployed in a cloud environment with a continuous integration pipeline, and its functionality is verified through unit, integration, and load testing.

KEYWORDS: ANIME, INTELLIGENT SEARCH, MANGA, RECOMMENDER SYSTEM, WEB PLATFORM.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	
		Савіцький Р.С.				
						4

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

AI – штучний інтелект

API – програмний інтерфейс застосунку

CDN – мережа доставки вмісту

CI/CD – безперервна інтеграція та доставка програмного забезпечення

CSR – рендеринг на боці клієнта

CSRF – міжсайтова підробка запиту

CSS – каскадні таблиці стилів

ER – модель «сутність–зв'язок»

GraphQL – мова запитів до програмного інтерфейсу

HMAC – код автентифікації повідомлень на основі хеш-функції

HTTP – протокол передавання гіпертексту

HTTPS – захищений протокол передавання гіпертексту

IDF – обернена частота документів

JSON – текстовий формат обміну даними

LLM – велика мовна модель

MMR – максимальна гранична релевантність

OAuth – відкритий протокол авторизації

ORM – об'єктно-реляційне відображення

PKCE – розширення захисту коду авторизації

REST – передавання репрезентативного стану

SEO – пошукова оптимізація

SQL – мова структурованих запитів

SSR – рендеринг на боці сервера

UI – інтерфейс користувача

UML – уніфікована мова моделювання

XSS – міжсайтовий скриптинг

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	5
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО ВЕБПЛАТФОРМИ.....	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного продукту	11
1.3. Вибір архітектури вебплатформи аніме та манги	18
1.4. Обґрунтування вибору інструментальних засобів	22
Висновки до першого розділу.....	25
РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ АНІМЕ ТА МАНГИ.....	27
2.1. Визначення варіантів використання та структури вебплатформи.....	27
2.2. Розробка бази даних	30
2.3. Огляд методів рекомендаційних систем.....	35
2.4. Проєктування та реалізація алгоритмів роботи	38
2.5. Реалізація вебплатформи аніме та манги	44
Висновки до другого розділу.....	52
РОЗДІЛ 3. ТЕСТУВАННЯ ТА РОЗГОРТАННЯ ВЕБПЛАТФОРМИ.....	54
3.1. Розгортання та безперервна інтеграція.....	54
3.2. Інтерфейс користувача	59
3.3. Тестування програмного забезпечення	78
Висновки до третього розділу	83
ВИСНОВКИ.....	84
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	86
ДОДАТКИ.....	89

ВСТУП

Актуальність теми. Індустрія аніме та манги стрімко зростає за обсягами контенту, кількістю адаптацій і франшиз, а українська аудиторія цих творів постійно збільшується й потребує україномовних сервісів для пошуку, систематизації та відстеження. Наявні популярні платформи переважно англо- чи япономовні, не мають повноцінної української локалізації та не враховують особливостей вітчизняного користувача. Орієнтацію ускладнює й велика кількість жанрів, форматів і франшиз, тому актуальною є розробка україномовної вебплатформи, що поєднує структуроване зберігання даних із каталогізацією, веденням персональних списків, персоналізованими рекомендаціями, інтелектуальним пошуком та інформуванням про вихід нових серій.

Метою кваліфікаційної роботи є розробка вебплатформи аніме та манги українською мовою, що забезпечує структуроване зберігання даних про твори, ведення персональних списків користувачів, гібридну рекомендаційну систему, інтелектуальний пошук та відстеження виходу нових серій.

Для досягнення поставленої мети визначено такі **завдання**:

- проаналізувати наявні аналоги та сформулювати функціональні вимоги до системи;
- обґрунтувати вибір архітектури та інструментальних засобів;
- спроектувати базу даних і сценарії використання;
- реалізувати основні модулі: каталог, персональні списки, рекомендації, пошук і сповіщення, а також механізми автентифікації та безпеки;
- розгорнути систему в хмарному середовищі з неперервною інтеграцією та доставкою;
- провести тестування системи.

Об'єктом дослідження є процес розробки вебплатформ у сфері цифрового медіаконтенту.

Предметом дослідження – методи, архітектурні рішення та інструментальні засоби проєктування й реалізації вебплатформи аніме та манги,

зокрема механізми організації даних, побудови рекомендаційної системи та інтелектуального пошуку.

Для досягнення мети використано методи системного аналізу для дослідження предметної області та формування вимог, порівняльного аналізу для оцінювання аналогів і вибору технологій, об'єктно-орієнтованого проєктування та UML-моделювання для побудови структури системи й бази даних, вебпрограмування для реалізації клієнтської та серверної частин, а також тестування й верифікації для перевірки коректності роботи реалізованого програмного комплексу.

Практичне значення одержаних результатів полягає у створенні готової до експлуатації україномовної вебплатформи аніме та манги, розгорнутої у хмарному середовищі. Розроблені архітектурні й технологічні рішення, структура бази даних, гібридна рекомендаційна система та інтелектуальний пошук можуть бути використані для подальшого розвитку платформи або як приклад побудови подібних веборієнтованих інформаційних систем.

Апробація результатів роботи. За результатами дослідження опубліковані дві тези [1, 2].

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	8
		Савіцький Р.С.				

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ФОРМУВАННЯ ВИМОГ ДО ВЕБПЛАТФОРМИ

1.1. Постановка задачі

Аніме та манга належать до найпопулярніших напрямів сучасної медіакультури і характеризуються значним та постійно зростаючим обсягом контенту: щороку виходять сотні нових творів, адаптацій, продовжень і відгалужень, які формують розгалужені франшизи. Орієнтуватися в такому масиві без спеціалізованих інструментів складно, тому навколо індустрії сформувалася екосистема онлайн-каталогів і спільнот, де користувачі ведуть власні списки перегляду, виставляють оцінки, пишуть відгуки, обговорюють твори та отримують рекомендації.

Аудиторія таких сервісів неоднорідна, адже досвідчені учасники цінують зручні списки та гнучкий пошук, а новачки потребують структурованого подання й рекомендацій. Однак наявні рішення задовольняють ці потреби лише частково, тож огляд виявляє низку проблем, що визначають потребу в новій платформі. Передусім ринок є фрагментованим, оскільки більшість сервісів спеціалізуються лише на одному аспекті, як-от каталог, новини чи соціальна взаємодія, і не поєднують структурований каталог, розвинену аналітику та механізми персоналізації в межах єдиного рішення. Суттєвою перешкодою залишається мовний бар'єр, адже всі провідні платформи іншомовні, а якісного україномовного сервісу для каталогізації аніме й манги досі немає, що ускладнює доступ для вітчизняної аудиторії. Крім того, персоналізація і пошук у наявних системах здебільшого обмежені, адже рекомендації мають базовий характер, а пошук зводиться до фільтрації за ключовими словами без урахування змісту запиту.

Отже, актуальною є розробка веборієнтованої платформи, яка поєднує структурований каталог, гібридну рекомендаційну систему, інтелектуальний пошук та відстеження виходу серій і орієнтована на українську аудиторію.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	
		Савіцький Р.С.				
						9

Метою розробки є створення вебплатформи аніме та манги українською мовою для структурованого зберігання даних про твори, ведення персональних списків перегляду, формування персоналізованих рекомендацій, інтелектуального пошуку та відстеження виходу нових серій, із забезпеченням взаємодії користувачів (відгуки, коментарі). Систему реалізовано як клієнт-серверний вебзастосунок із гібридним підходом SSR/CSR: клієнтська частина та REST API на базі Next.js [19], зберігання даних у реляційній СУБД PostgreSQL.

Перелік задач, що підлягають автоматизації:

- каталогізація аніме й манги (зберігання, пошук, фільтрація, детальні сторінки творів, персонажів і причетних осіб);
- ведення персональних списків перегляду зі статусами та оцінками;
- формування персоналізованих рекомендацій;
- відстеження виходу нових серій та інформування користувачів;
- взаємодія спільноти (відгуки, коментарі);
- адміністрування контенту з розмежуванням прав доступу.

Функціональні вимоги охоплюють дії, які система забезпечує користувачам різних ролей, як-от перегляд каталогу, ведення персональних списків, отримання рекомендацій, інтелектуальний пошук та адміністрування контенту. Їх повний перелік наведено в табл. 1.1.

Таблиця 1.1

Функціональна вимога

№	Функціональна вимога	Пріоритет	Статус
1.	2.	3.	4.
1	Реєстрація та автентифікація користувачів (сесійний токен)	Високий	Реалізовано
2	Розмежування доступу за роллю (користувач, адміністратор)	Високий	Реалізовано
3	Каталог аніме й манги з детальними сторінками творів, персонажів і людей	Високий	Реалізовано
4	Персональні списки зі статусами та оцінками	Високий	Реалізовано
5	Додавання творів, персонажів і людей до обраного	Середній	Реалізовано
6	Відгуки з оцінкою та коментарі з відповідями	Високий	Реалізовано
7	Гібридна рекомендаційна система (контентна та колаборативна)	Високий	Реалізовано
8	Інтелектуальний пошук (стандартний та AI)	Високий	Реалізовано
9	Календар виходу серій (інтеграція AniList)	Середній	Реалізовано
10	Внутрішні сповіщення (серії, відповіді, системні події)	Середній	Реалізовано

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	10
		Савіцький Р.С.				

продовж. табл. 1.1

1.	2.	3.	4.
11	Зберігання зв'язків між творами (франшизи, пов'язані твори)	Середній	Реалізовано
12	Адміністрування контенту	Високий	Реалізовано
13	Повна рольова модель (контент-мейкер, перекладач, модератор)	Середній	Перспектива
14	Форумний модуль	Низький	Перспектива
15	Користувацькі колекції	Низький	Перспектива
16	Графова візуалізація франшиз	Низький	Перспектива
17	Публікація новин	Низький	Перспектива
18	Веб-пуш сповіщення	Низький	Перспектива

Окрім функціональних, до системи висунуто нефункціональні вимоги, що визначають якість її роботи та умови експлуатації, наприклад доступність, продуктивність, масштабованість, безпеку, зручність використання та україномовну локалізацію. Повний перелік цих вимог наведено в табл. 1.2.

Таблиця 1.2

Нефункціональні вимоги до системи

Категорія	Вимога
Доступність	Безперервна робота системи в режимі 24/7
Продуктивність	Час відповіді сервера не більше 1 с за одночасного навантаження до 100 користувачів
Масштабованість	Вертикальне масштабування ресурсів із можливістю обслуговування до 1000 користувачів
Безпека	Захист від SQL-ін'єкцій, XSS- та CSRF-атак
Зручність використання	Адаптивний інтерфейс для мобільних і настільних пристроїв
Локалізація	Інтерфейс виключно українською мовою (без багатомовності)
Розгортання	Розгортання у хмарному середовищі з інтеграцією CI/CD

Таким чином, поставлена задача полягає у проєктуванні та реалізації масштабованої вебплатформи аніме та манги з розширеним функціоналом взаємодії користувачів, аналітичними механізмами та можливістю подальшого розвитку складних інтелектуальних модулів.

1.2. Аналіз аналогів програмного продукту

Для визначення функціональних вимог і формування конкурентних переваг вебплатформи проведено аналіз існуючих вебсервісів у сфері аніме та манги: MyAnimeList, Anime News Network, AniList, Kitsu. Аналіз охоплює структуру каталогу, деталізацію сторінок творів і персонажів, систему списків,

механізми взаємодії користувачів, рекомендаційні інструменти, пошукові можливості та загальні особливості інтерфейсу.

MyAnimeList (MAL) – один із найбільших і найстаріших каталогів аніме та манги [3]. Головна сторінка (рис. 1.1) містить навігаційне меню, блоки популярних і сезонних творів та доступ до списків користувача.

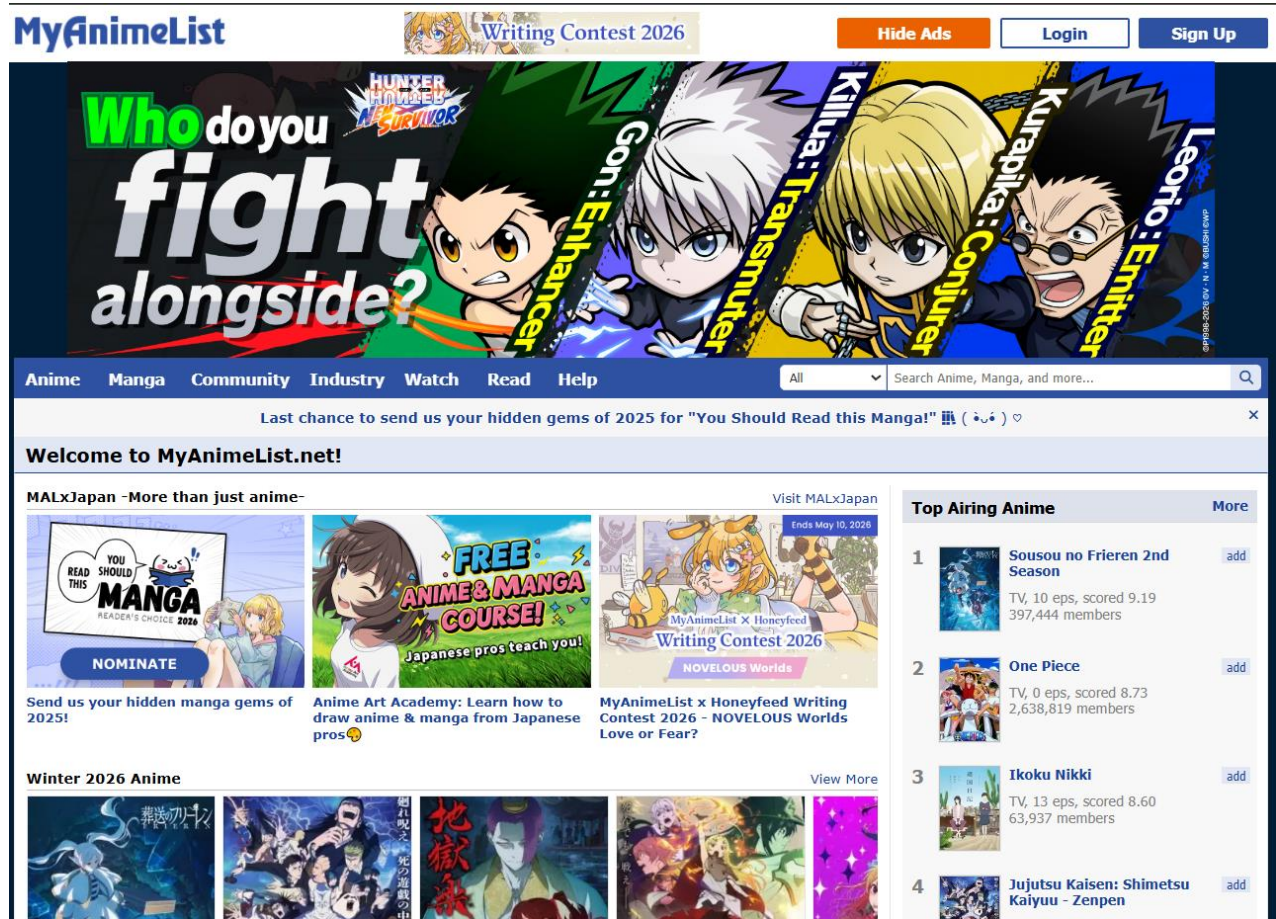


Рис. 1.1. Головна сторінка вебплатформи MyAnimeList

Сторінки творів (рис. 1.2) детально опрацьовані й містять опис, рейтинг, статистику переглядів, жанри, студію, тривалість, пов'язані роботи та персонажів, а для кожного персонажа передбачено окрему сторінку з біографією й акторами озвучення. Реалізовано розвинену систему персональних списків зі статусами та оцінками, а також рецензії, коментарі й форум. Пошук фільтрує за жанрами й типами, проте обмежується ключовими словами без семантичної обробки; зв'язки між творами подано лише текстово.

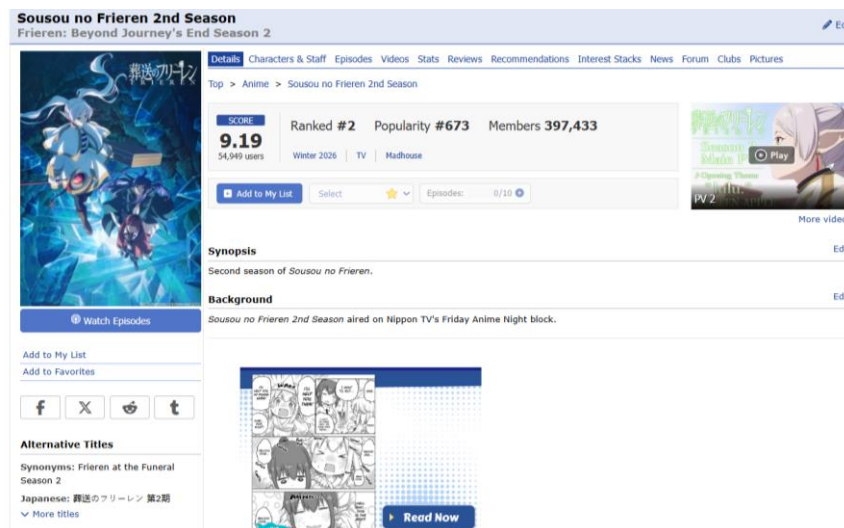


Рис. 1.2. Сторінка аніме на MyAnimeList

MyAnimeList вирізняється повнотою каталогу та активною спільнотою, натомість його інтерфейс перевантажений, а української локалізації немає. Із цього сервісу варто перейняти глибину деталізації сторінок і гнучкість персональних списків.

Anime News Network (ANN) – інформаційний портал з акцентом на новинах індустрії [4]. Головна сторінка (рис. 1.3) побудована навколо новинних і редакційних матеріалів та анонсів, що визначає її як основний робочий простір сервісу.

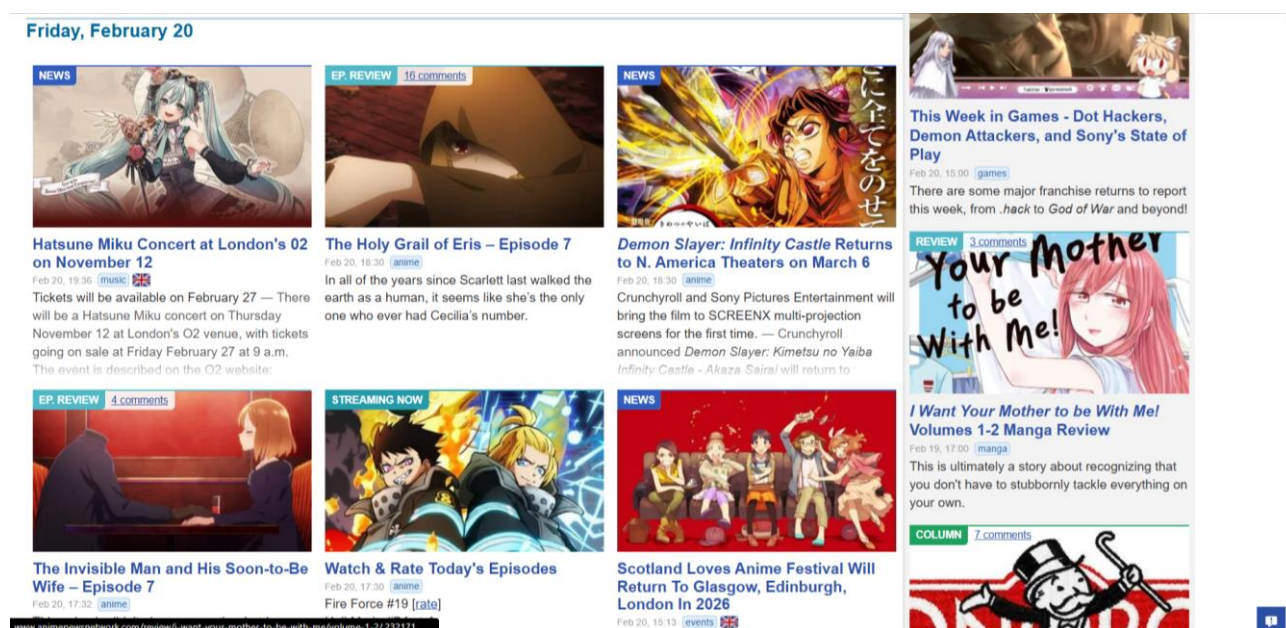


Рис. 1.3. Головна сторінка Anime News Network

		Залозний О.А.				
		Савіцький Р.С.				

База творів виконує допоміжну роль і містить лише базову інформацію про твори та посилання на пов'язані матеріали; сторінки персонажів обмежені, повноцінної системи персональних списків немає, коментарі доступні переважно в межах новин, рекомендаційна система відсутня, а пошук має базовий характер.

Anime News Network тримається на якісному новинному наповненні та редакційній аналітиці індустрії, проте його каталог слабкий, а списків і рекомендацій немає. Звідси варто перейняти підхід до подачі новин та анонсів.

AniList – сучасна платформа з орієнтацією на персоналізацію та аналітику [5]. Інтерфейс мінімалістичний, із гнучким фільтруванням; профіль користувача (рис. 1.4) містить розширену статистику переглядів та вподобань.

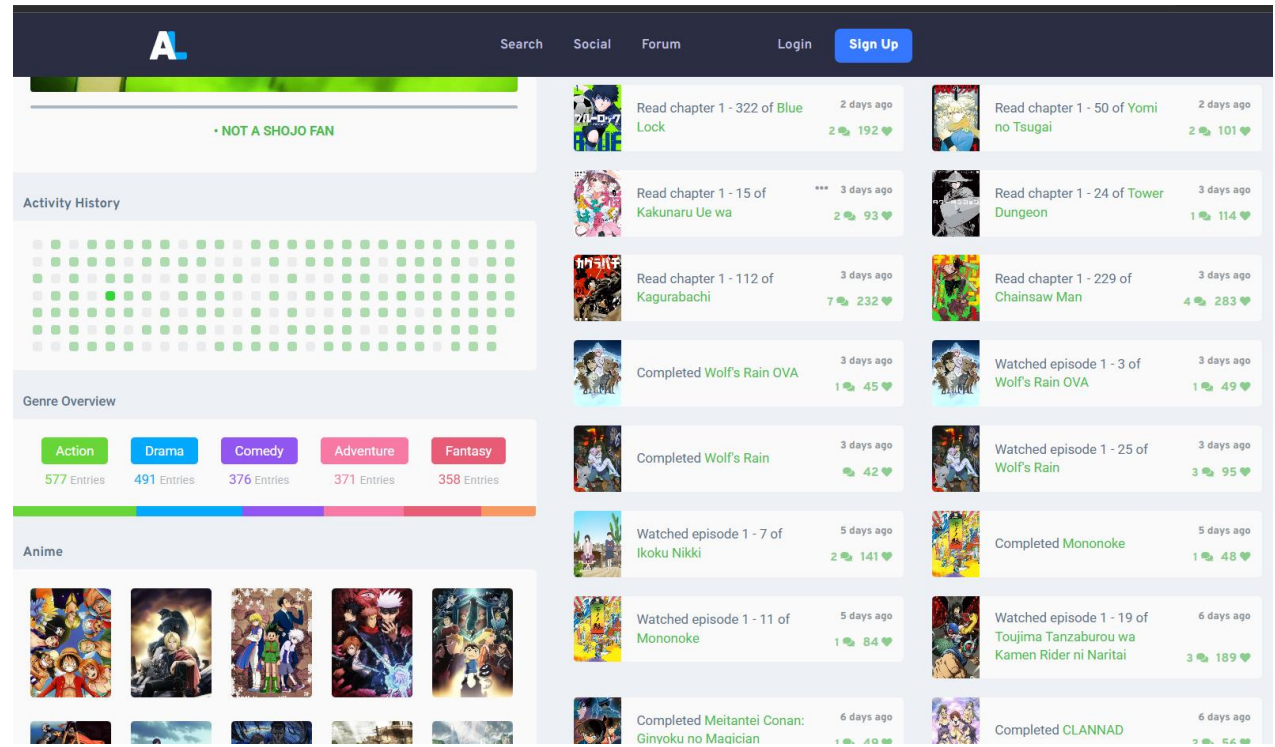


Рис. 1.4. Профіль користувача на AniList

Сторінки творів (рис. 1.5) містять опис, жанри, теги, статистику, пов'язані роботи та персонажів. Реалізовано гнучкі списки, розширений пошук за жанрами, роками й іншими параметрами та рекомендації на основі вподобань, проте зв'язки між творами відображаються лише текстово, а графова модель відсутня.

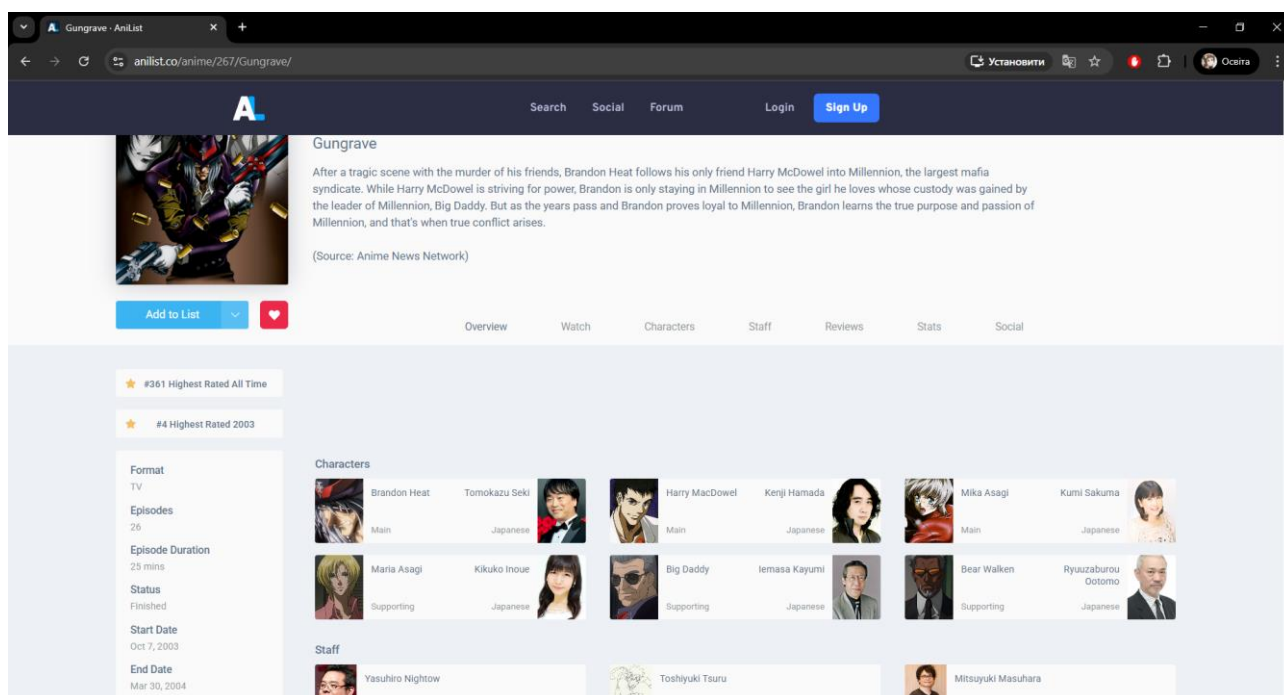


Рис. 1.5. Сторінка аніме на AniList

AniList виділяється персоналізацією, аналітикою профілю та сучасним інтерфейсом, хоча української локалізації та графового подання зв'язків йому бракує. Із нього варто перейняти гнучкість фільтрів, аналітику профілю та якість взаємодії.

Kitsu – платформа з вираженою соціальною складовою та стрічковим форматом взаємодії [6]. Профіль користувача орієнтований на соціальну активність і відображає стрічку дій та вподобань. У верхній частині розміщено основні відомості про користувача та статистику його активності, а нижче – стрічку оновлень, що формується з оцінок, доданих творів і взаємодій спільноти. Такий стрічковий підхід наближає платформу до соціальної мережі та стимулює регулярне повернення користувачів, проте зміщує акцент із каталогізації творів на соціальну комунікацію.

Помітною рисою Kitsu є тісна інтеграція персональних списків зі стрічкою активності, адже зміна статусу чи оцінки твору одразу відображається у стрічці підписників користувача. Платформа підтримує підписки на інших учасників і базові соціальні взаємодії, що наближає її до тематичної соціальної мережі, тоді як довідкові та аналітичні можливості залишаються допоміжними. Вигляд профілю користувача наведено на рис. 1.6.

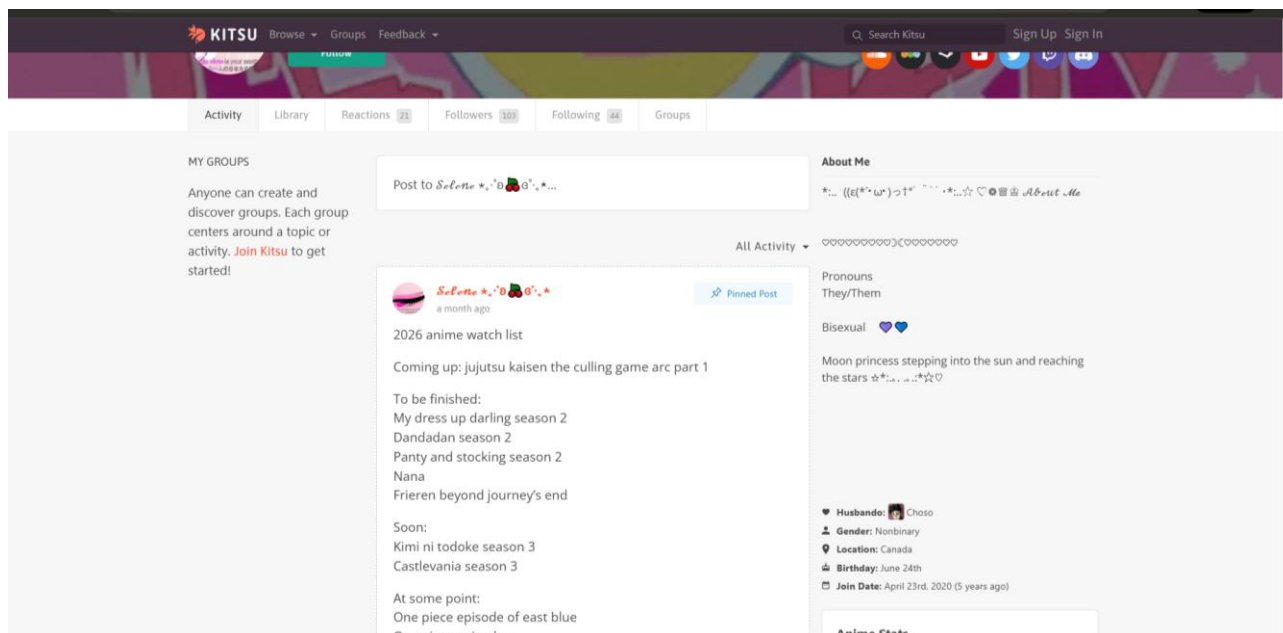


Рис. 1.6. Профіль користувача на Kitsu

Сторінки творів (рис. 1.7) містять опис, оцінку та список персонажів, проте поступаються деталізацією MyAnimeList та AniList. Реалізовано систему списків, відгуки й коментарі, однак рекомендації та пошук мають базовий характер, а візуалізація франшиз відсутня.

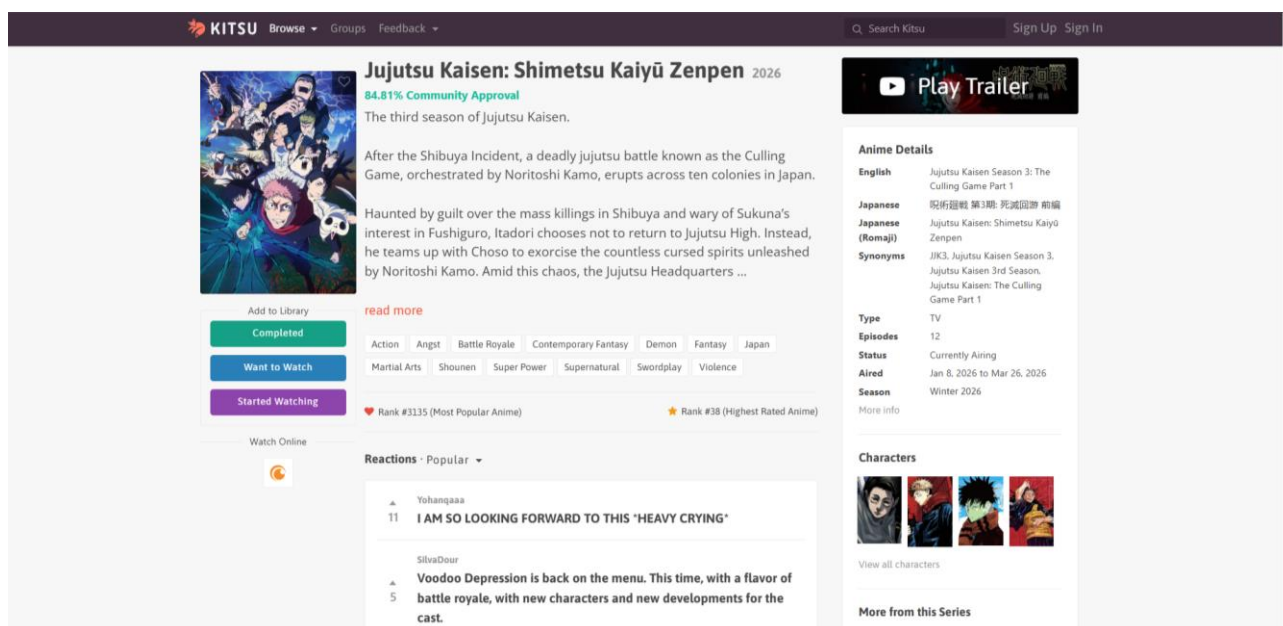


Рис. 1.7. Сторінка аніме на Kitsu

Kitsu приваблює соціальною взаємодією та сучасним виглядом, але поступається деталізацією каталогу, а рекомендації й пошук у ньому базові. Корисно перейняти елементи соціальної взаємодії та зручність стрічкового подання.

Проведений аналіз показує, що кожна з розглянутих платформ сильна у своєму напрямі. MyAnimeList вирізняється повнотою каталогу, Anime News Network зосереджений на новинах індустрії, AniList пропонує розвинену персоналізацію та аналітику, а Kitsu робить акцент на соціальній взаємодії. Проте жодна з них не поєднує структурований каталог, гібридну рекомендаційну систему та сучасний адаптивний інтерфейс у межах єдиного рішення, а всі ці сервіси іншомовні, тож якісної платформи українською мовою не існує.

Отже, проєктована система переймає найкращі практики аналогів, детальні сторінки творів, персональні списки та фільтрований пошук, і доповнює їх українськомовним інтерфейсом, гібридними рекомендаціями, інтелектуальним пошуком, календарем виходу серій та зв'язками між творами. Порівняння функціональних можливостей розглянутих платформ і проєктованої системи наведено в табл. 1.3.

Таблиця 1.3

Порівняльна таблиця функціональних можливостей

Критерій	MAL	ANN	AniList	Kitsu	AniMan
Повноцінний каталог	Так	Частково	Так	Так	Так
Деталізація сторінки твору	Висока	Середня	Висока	Середня	Висока
Окремі сторінки персонажів	Деталізовані	Обмежені	Деталізовані	Частково	Деталізовані
Персональні списки	Розширені	Відсутні	Гнучкі	Реалізовані	Розширені (статуси, оцінки, прогрес)
Відгуки та коментарі	Так	До новин	Так	Так	Так
Форумний модуль	Так	Частково	Частково	Частково	Перспектива
Рекомендації	Так	Ні	Так	Базові	Гібридні (контент + колаборація)
Розширений пошук	Обмежений	Базовий	Розширений	Базовий	Розширений + AI-пошук
Зв'язки франшиз	Текстові	Текстові	Текстові	Текстові	Структуровані (граф перспектива)
Аналітика профілю	Базова	Ні	Розширена	Обмежена	Так

1.3. Вибір архітектури вебплатформи аніме та манги

Архітектура визначає спосіб розподілу відповідальності між частинами системи та безпосередньо впливає на її надійність, продуктивність і придатність до подальшого розвитку. Для платформи, що оперує складно пов'язаними даними про твори, персонажів, осіб і франшизи, виваженість архітектурних рішень є особливо важливою, оскільки саме вона визначає, наскільки легко система витримуватиме зростання обсягу даних і додавання нових можливостей. Вебплатформу спроектовано за клієнт-серверною моделлю з поділом на три логічні рівні – представлення, прикладний та рівень даних.

Рівень представлення відповідає за відображення інформації та взаємодію з користувачем, прикладний рівень зосереджує бізнес-логіку, обробку запитів і застосування правил предметної області, а рівень даних забезпечує надійне зберігання та узгоджений доступ до інформації. Таке розмежування дає змогу змінювати інтерфейс, не зачіпаючи логіки, а структуру бази даних без переписування клієнтської частини. Це спрощує супровід, полегшує тестування окремих частин незалежно одна від одної та уможливорює поступове нарощування функціональності без ризику порушити вже наявну.

Монолітна архітектура реалізує систему єдиним розгортаним застосунком, що спрощує розробку, налагодження та розгортання, проте з часом ускладнює незалежне масштабування й оновлення окремих частин. Мікросервісна архітектура розбиває систему на самостійні служби, кожен з яких можна масштабувати та розгортати окремо, однак така гнучкість дається ціною значно складнішої інфраструктури, потреби в оркестрації та налагодженні міжсервісної взаємодії, що є надмірним для проекту, який створює одна особа. Безсерверний підхід забезпечує автоматичне масштабування й оплату лише за фактичні виклики, але накладає обмеження на час виконання функцій та ускладнює тривалі фонові операції. З огляду на масштаб роботи, орієнтацію на швидке хмарне розгортання й підтримку однією особою обрано монолітну архітектуру на базі Next.js, у межах якої клієнтська частина та серверний

програмний інтерфейс розміщено в одному проєкті. Порівняння архітектурних підходів за ключовими критеріями наведено в табл. 1.4.

Таблиця 1.4

Порівняння архітектурних підходів для розробки вебплатформи

Критерій	Моноліт	Мікросервіси	Serverless
Складність розробки	Низька	Висока	Середня
Складність розгортання	Низька	Висока	Низька
Масштабування	Цілісне (обмежене)	Гнучке, незалежне	Автоматичне
Інфраструктурні витрати	Низькі	Високі	Оплата за виклик
Незалежність компонентів	Низька	Висока	Середня
Придатність (одна особа, хмара)	Висока	Низька	Середня

Попри монолітну організацію, застосунок побудовано як модульний моноліт, у якому логіку згруповано за функціональними теками відповідно до окремих можливостей системи, як-от каталог, рекомендації, пошук чи сповіщення. Кожна така тека містить власні компоненти, бізнес-логіку та доступ до даних, що зменшує зв'язність між частинами й полегшує орієнтацію в кодовій базі. Такий підхід поєднує простоту моноліту з впорядкованістю, властивою сервісній архітектурі, і за потреби дає змогу виокремити найнавантаженіші модулі в незалежні служби надалі, не перебудовуючи систему загалом.

Спосіб взаємодії між клієнтом і сервером визначає, як передаються дані та наскільки зручно інтегрувати зовнішні системи. Архітектурний стиль REST організовує доступ до ресурсів через стандартні методи протоколу передачі гіпертексту, вирізняється простотою, передбачуваністю, придатністю до кешування та широкою підтримкою інструментів, хоча для отримання складно пов'язаних даних інколи потребує кількох запитів. Альтернативний підхід на основі мови запитів до даних дозволяє отримати потрібний набір полів одним зверненням, проте ускладнює налаштування сервера, контроль навантаження та кешування відповідей. Оскільки модель даних платформи добре лягає на ресурсну організацію, а більшість сторінок завантажують цілісні набори даних, обрано REST як простіший у реалізації, прозоріший в експлуатації та зручніший для інтеграції із зовнішніми сервісами. Відсутність стану на боці сервера між запитами додатково спрощує горизонтальне відтворення застосунку.

Окрему увагу приділено моделі формування сторінок, оскільки вона безпосередньо впливає на швидкість завантаження, видимість для пошукових систем та інтерактивність. Серверне формування сторінок забезпечує швидке первинне відображення й повноцінну індексацію вмісту, але створює додаткове навантаження на сервер. Формування на боці клієнта дає багату інтерактивність і розвантажує сервер ціною повільнішого першого відображення та гіршої видимості для пошукових систем. Статична генерація є найшвидшою, проте погано надається до часто змінюваного та персоналізованого вмісту. Тому в системі застосовано гібридну модель, у якій публічні сторінки каталогу й творів формуються на сервері заради швидкого завантаження та індексації, а інтерактивні розділи, такі як персональні списки, профіль, рекомендації й налаштування, працюють переважно на боці клієнта. Сучасний підхід серверних компонентів дає змогу поєднати обидві стратегії в межах однієї сторінки, виконуючи звернення до даних на сервері й залишаючи інтерактивні елементи клієнтові, що зменшує обсяг коду, який передається у браузер.

Логічно система складається з трьох взаємодійних рівнів, узагальнену структуру яких наведено на рис. 1.8. Рівень представлення утворюють сторінки та компоненти інтерфейсу, прикладний рівень охоплює обробники маршрутів програмного інтерфейсу й виокремлений шар доступу до даних із бізнес-логікою, а рівень даних представлений реляційною базою даних. Типове звернення проходить від інтерфейсу до прикладного рівня, який перевіряє права доступу, застосовує правила предметної області та звертається до сховища, після чого сформований результат повертається користувачеві. Такий упорядкований рух даних робить поведінку системи передбачуваною й полегшує локалізацію та виправлення помилок.

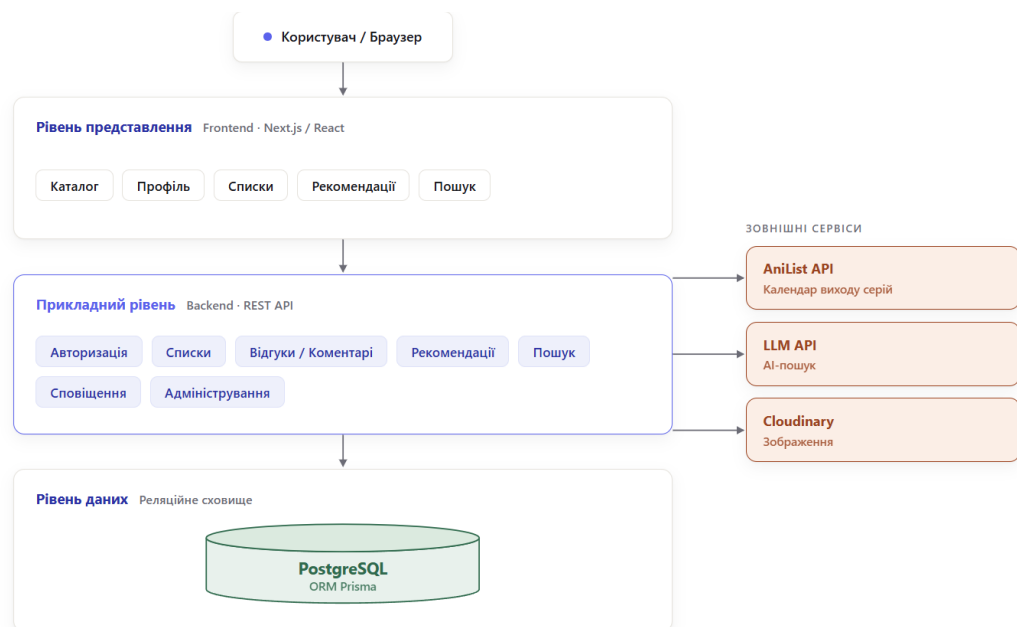


Рис. 1.8. Архітектура вебплатформи аніме та манги

Поза основними рівнями виокремлено шар зовнішніх інтеграцій, що відмежовує залежності від сторонніх сервісів від ядра системи. Через цей шар платформа звертається до програмного інтерфейсу AniList для синхронізації розкладу виходу серій і формування сповіщень, до мовних моделей для опціонального інтелектуального пошуку та до хмарного сховища Cloudinary для зберігання й оптимізованої доставки зображень. Зосередження зовнішніх звернень в окремому шарі дає змогу замінювати чи доповнювати постачальників послуг без впливу на решту коду, а також ізолювати систему від тимчасової недоступності сторонніх сервісів.

Архітектура застосунку забезпечує безпеку на кількох рівнях. Проміжне програмне забезпечення перехоплює звернення до захищених та адміністративних розділів, перевіряє підписаний сесійний токен ще до обробника маршруту й звіряє права за роллю користувача. Сервер не зберігає стан сесії, бо токен сам містить потрібні відомості й захищений криптографічним підписом, тому систему простіше масштабувати. Доступ до бази через об'єктно-реляційне відображення зменшує ризик впровадження стороннього коду в запити. Поділ публічних і захищених маршрутів та перевірка походження запитів обмежують міжсайтові атаки.

До наскрізних механізмів системи належать кешування, обмеження частоти запитів і журналювання. Результати ресурсомістких операцій, звернень до мовних моделей, тимчасово зберігаються, що зменшує навантаження й витрати, а обмеження частоти захищає зовнішні інтеграції від надмірних звернень. Журналювання ключових подій забезпечує спостережуваність системи та полегшує діагностику в експлуатації. Винесення цих механізмів на спільний рівень дає змогу застосовувати їх однаково в усіх модулях.

Систему розраховано на безперервну роботу та зростання навантаження. Завдяки відсутності серверного стану застосунок легко відтворюється в кількох екземплярах, а в межах обраної моделі передбачено вертикальне масштабування через збільшення обчислювальних ресурсів та оптимізацію доступу до бази даних. Розгортання здійснюється у хмарному середовищі, що бере на себе балансування навантаження й доставку статичних ресурсів через мережу розповсюдження вмісту, а оновлення проходять через конвеєр неперервної інтеграції та доставки на основі системи контролю версій. Такий підхід забезпечує автоматизацію перевірок, збірки й розгортання, відтворюваність середовища та можливість оновлювати систему без тривалих зупинок.

1.4. Обґрунтування вибору інструментальних засобів

Основними критеріями були безпека та надійність коду, підтримка гібридного формування сторінок, масштабованість, зручність супроводу однією особою й просте хмарне розгортання. Перевагу надавали зрілим інструментам з активною спільнотою та доброю сумісністю між собою, що дає змогу зосередитися на прикладній логіці, а не на налаштуванні інфраструктури.

Кодову базу написано мовою TypeScript, статична типізація якої виявляє помилки вже під час написання коду, до його виконання, і робить масштабні зміни безпечнішими. Один опис типів діє і на клієнті, і на сервері, тож обидві частини застосунку лишаються узгодженими.

Клієнтську частину побудовано на React, компонентний підхід якого дає змогу зібрати складні екрани каталогу, профілю чи списків із менших повторно

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	22
		Савіцький Р.С.				

використовуваних елементів, а декларативний опис інтерфейсу спрощує керування станом. На відміну від цілісних каркасів на кшталт Angular, React не нав'язує жорсткої структури й має найбільшу серед подібних бібліотек спільноту та набір готових рішень. Поверх нього використано каркас Next.js, що об'єднує клієнтську частину й серверні маршрути в одному проєкті та з коробки надає гібридне формування сторінок, файлову маршрутизацію, серверні компоненти й оптимізацію зображень. Зв'язка окремого сервера на Express із клієнтом на React вимагала б підтримувати два проєкти й дублювати конфігурацію, тому єдиний каркас тут практичніший.

Оформлення інтерфейсу виконано засобами Tailwind CSS, де стилі задаються готовими класами прямо в розмітці, що прискорює розробку й тримає вигляд платформи в єдиній дизайн-системі через спільні токени кольорів, відступів і типографіки. Порівняно зі звичайними таблицями стилів це зменшує дублювання й не змушує вигадувати назви класів, а порівняно з рішеннями, що вбудовують стилі в код, не роздуває обсяг готового застосунку.

Дані зберігаються в реляційній СУБД PostgreSQL, адже предметна область творів, жанрів, персонажів і франшиз складається з численних пов'язаних сутностей, де важливі суворі схема, транзакційність і цілісність, тому реляційна модель підходить краще за документоорієнтовану на кшталт MongoDB. PostgreSQL до того ж працює з даними у форматі JSON і повнотекстовим пошуком, що покриває й менш структуровані випадки. Саму базу розгорнуто в хмарному сервісі Neon, який надає безсерверну PostgreSQL з автоматичним масштабуванням і не потребує адміністрування власного сервера.

Доступ до даних реалізовано через об'єктно-реляційне відображення Prisma, яке надає типобезпечні запити, узгоджені зі схемою бази, автоматично генерований клієнт і зручний механізм міграцій, що синхронізує структуру бази зі змінами в коді. Порівняно з написанням запитів мовою структурованих запитів вручну це суттєво знижує ризик помилок і прискорює розробку, а порівняно з іншими інструментами подібного призначення Prisma вирізняється якіснішою

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	23
		Савіцький Р.С.				

інтеграцією із системою типів. Звернення до бази зосереджено в окремому шарі доступу до даних, що відмежовує роботу зі сховищем від решти логіки.

Автентифікацію побудовано на основі підписаного сесійного токена, а доступ розмежовано за ролями користувачів. Токен формується на сервері, захищається криптографічним підписом і передається у захищеному файлі cookie, що дає змогу перевіряти права без зберігання сесій на сервері. Паролі зберігаються у вигляді хешу, обчисленого стійкою до підбору функцією з індивідуальною сіллю. Окрім входу за логіном і паролем, передбачено автентифікацію через зовнішні сервіси Google та X за відкритим протоколом авторизації, що спрощує реєстрацію для користувачів.

Медіафайли, як-от обкладинки творів, зображення персонажів і аватари користувачів, зберігаються й доставляються через хмарний сервіс Cloudinary. Він бере на себе оптимізацію, перетворення форматів і роздачу зображень через мережу розповсюдження вмісту, що зменшує навантаження на застосунок і пришвидшує завантаження сторінок. Такий підхід практичніший за самостійне зберігання файлів, оскільки не потребує власної інфраструктури для обробки та доставки медіа.

Для календаря виходу серій інтегровано програмний інтерфейс AniList, що є джерелом розкладу й даних про нові епізоди. Опціональний інтелектуальний пошук спирається на зовнішні мовні моделі, доступ до яких організовано через ланцюг постачальників із резервуванням, де основним є сервіс OpenAI, а за його недоступності використовуються безкоштовні моделі через OpenRouter. Винесення цих функцій на зовнішні сервіси дає змогу скористатися можливостями сучасних мовних моделей без потреби розгортати й утримувати власну обчислювальну інфраструктуру.

Версіонування коду здійснюється в системі Git, а автоматизацію перевірок, збірки й розгортання забезпечує GitLab CI/CD. Хостинг надає хмарна платформа Vercel, яка нативно підтримує Next.js, бере на себе балансування навантаження й доставку статичних ресурсів та автоматично розгортає нову версію після проходження конвеєра. Для перевірки коректності роботи застосовано засіб

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	24
		Савіцький Р.С.				

модульного тестування Vitest, а навантажувальне тестування виконується інструментом Apache JMeter. Зведений технологічний стек системи наведено в табл. 1.5.

Таблиця 1.5

Технологічний стек системи

Складова	Технологія	Призначення
Мова програмування	TypeScript	Типобезпечна розробка
Клієнтська частина	React + Next.js	Компонентний UI, SSR/CSR
Стилізація	Tailwind CSS	Адаптивний інтерфейс, єдина дизайн-система
Серверна частина	Next.js (REST API)	Бізнес-логіка та ендпоїнти
Доступ до даних	Prisma (ORM)	Типобезпечні запити, міграції
База даних	PostgreSQL (Neon)	Зберігання даних, складні зв'язки
Автентифікація	Сесійний токен	Доступ без зберігання сесій на сервері
Зображення	Cloudinary	Зберігання й доставка медіа (CDN)
Зовнішні API	AniList, LLM (OpenAI/OpenRouter)	Календар виходу серій, AI-пошук
Контроль версій / CI/CD	Git, GitLab CI	Версіонування, автоматизація
Розгортання	Vercel	Хмарний хостинг Next.js

Для зростання навантаження до приблизно тисячі користувачів передбачено вертикальне масштабування шляхом збільшення обчислювальних ресурсів та оптимізації роботи бази даних. Серверне середовище має забезпечувати стабільну роботу цілодобово, резервне копіювання бази даних, контроль журналів помилок і можливість оновлення без повної зупинки системи. Оскільки систему розгорнуто на хмарних платформах Vercel і Neon, ці вимоги задовольняються ресурсами хмарного середовища без потреби в окремому фізичному сервері.

Висновки до першого розділу

У першому розділі проаналізовано предметну область і сформовано вимоги до вебплатформи аніме та манги. Сформульовано постановку задачі, визначено перелік задач автоматизації, функціональні та нефункціональні вимоги до системи з урахуванням ролей користувачів, інтеграції зовнішніх сервісів та орієнтації на промислове розгортання.

Проведено аналіз чотирьох наявних платформ (MyAnimeList, Anime News Network, AniList, Kitsu), що дозволило визначити їхні сильні сторони й обмеження. Встановлено, що жодна з них не поєднує структурований каталог, гібридну рекомендаційну систему та сучасний адаптивний інтерфейс у межах єдиного рішення, а всі розглянуті сервіси є іншомовними. Це обґрунтовує доцільність розробки платформи українською мовою, що має власні переваги, наприклад гібридні рекомендації, інтелектуальний пошук, календар виходу серій і структуроване представлення зв'язків між творами.

Обґрунтовано вибір клієнт-серверної архітектури з гібридним рендерингом SSR/CSR на базі Next.js, взаємодії через REST API та реляційної СУБД PostgreSQL. Визначено технологічний стек реалізації (TypeScript, React, Tailwind CSS, Prisma, Cloudinary), засоби розгортання (Vercel, GitLab CI/CD) і вимоги до апаратного забезпечення, що забезпечують стабільну роботу та можливість подальшого масштабування.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ АНІМЕ ТА МАНГИ

2.1. Визначення варіантів використання та структури вебплатформи

Варіанти використання описує закінчену взаємодію актора із системою, що приносить йому певний корисний результат, а актор позначає роль, у якій користувач або зовнішня система звертається до платформи. Сукупність варіантів використання разом з акторами та відношеннями між ними утворює модель прецедентів, яка окреслює межі системи й розподіл функцій між ролями. Між акторами застосовано відношення узагальнення, за якого похідна роль успадковує всі можливості базової та додає власні, що дає змогу не дублювати спільні дії й тримати модель компактною.

У системі визначено трьох основних акторів, які відрізняються обсягом доступних можливостей. Гість є неавторизованим відвідувачем і має доступ лише до публічної частини платформи, перегляду каталогу, сторінок творів, персонажів та осіб і користування пошуком. Зареєстрований користувач успадковує всі можливості гостя й додатково веде персональні списки перегляду та читання, оцінює твори, залишає відгуки й коментарі, додає твори до обраного, отримує персональні рекомендації, переглядає календар виходу серій і сповіщення та налаштовує власний профіль. Адміністратор успадковує можливості зареєстрованого користувача й додатково керує наповненням платформи, додає та редагує твори, персонажів, осіб, студії й видавництва, опрацьовує запропоновані правки та публікує оголошення. Ролі автора контенту, перекладача й модератора передбачено як напрям подальшого розвитку рольової моделі.

Узагальнені варіанти використання за акторами наведено на рис. 2.1. На діаграмі прецедентів кожного актора пов'язано з доступними йому варіантами використання, а відношення узагальнення між акторами відображає успадкування можливостей, за якого спільні прецеденти не дублюються, а для кожної ролі додаються лише специфічні саме для неї дії. Така модель

залишається компактною й наочно показує, як зростає обсяг доступних функцій від гостя до зареєстрованого користувача та далі до адміністратора.

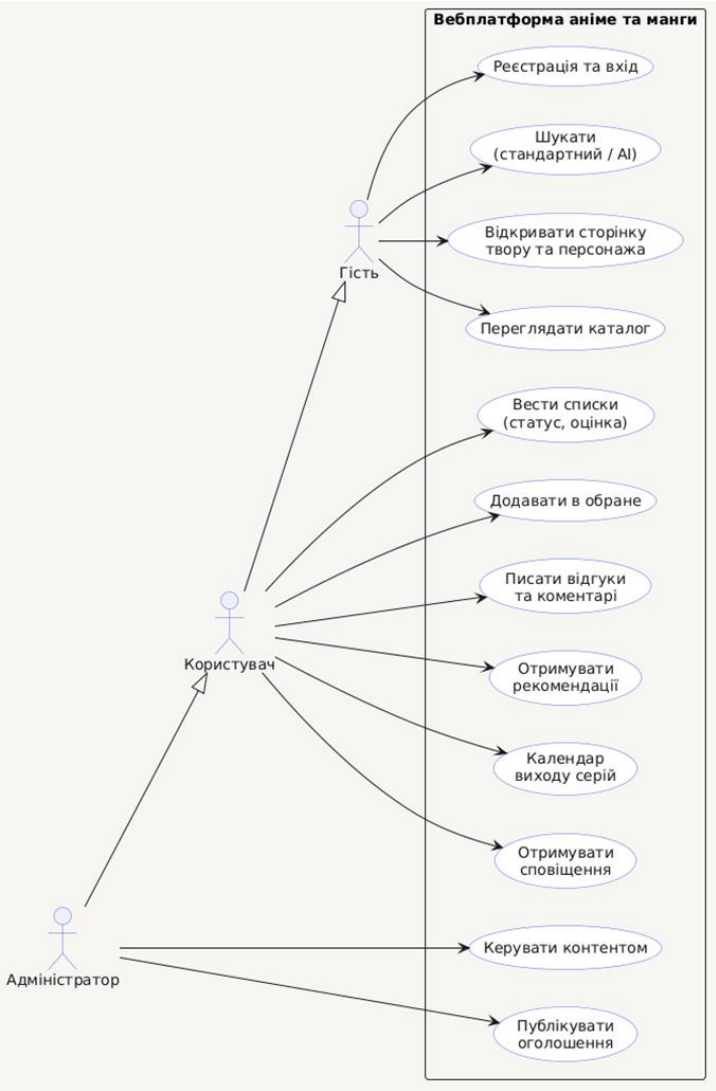


Рис. 2.1. Діаграма варіантів використання

Для уточнення поведінки системи ключові варіанти використання деталізовано у вигляді сценаріїв, що описують передумову, послідовність дій і очікуваний результат взаємодії.

Реєстрація та автентифікація є вихідним сценарієм для гостя, який ще не має облікового запису. Користувач заповнює форму реєстрації або входу, після чого система перевіряє введені дані, створює обліковий запис або видає підписаний сесійний токен. Унаслідок цього гість набуває статусу зареєстрованого користувача й дістає доступ до всіх відповідних можливостей платформи.

Під час додавання твору до особистого списку зареєстрований користувач, перебуваючи на сторінці твору, обирає потрібний статус, наприклад «Дивлюся» чи «Заплановано». Система зберігає відповідний запис у персональному списку, оновлює прогрес перегляду та, якщо користувач виставив оцінку, фіксує її. У результаті твір з'являється в списку користувача з обраним статусом і враховується надалі під час формування рекомендацій.

Отримання рекомендацій доступне зареєстрованому користувачеві, який уже має історію оцінок або сформовані списки. Відкривши відповідний розділ, користувач ініціює формування переліку творів, що його система будує за гібридним підходом, поєднуючи властивості творів і вподобання схожих користувачів та враховуючи задані фільтри. Користувач переглядає видачу й може перейти до будь-якого рекомендованого твору, отримавши таким чином персоналізований перелік.

Окремим сценарієм є інтелектуальний пошук, доступний будь-якому користувачеві. Користувач формулює запит природною мовою, наприклад описує бажаний сюжет чи настрій твору, а система за допомогою мовної моделі інтерпретує запит, зіставляє його з даними каталогу й повертає релевантні твори з коротким поясненням добору. Так користувач знаходить потрібні твори навіть тоді, коли не може точно задати критерії через стандартні фільтри.

Перегляд календаря виходу серій призначений для авторизованого користувача, який стежить за аніме зі своїх списків. Відкривши календар, користувач бачить розклад виходу нових серій, що його система формує на основі даних, синхронізованих із зовнішнім сервісом, і впорядковує за датами. Так користувач отримує наочне уявлення про найближчі виходи серій цікавих йому творів.

Отримання сповіщень відбувається у відповідь на події, пов'язані з активністю користувача, як-от вихід нової серії, відповідь на коментар чи системне оголошення. Система формує сповіщення для відповідних користувачів, а користувач переглядає їх у спеціальному розділі та позначає прочитаними, автоматично під час перегляду. Це тримає користувача

поінформованим про важливі для нього події без потреби самостійно відстежувати зміни.

На основі визначених варіантів використання сформовано узагальнену функціональну структуру платформи, у якій систему поділено на взаємодійні модулі, і кожен з них відповідає за окрему групу можливостей. Модуль каталогу забезпечує зберігання й перегляд творів, персонажів та осіб, модуль персональних списків та обраного веде колекції користувача, а модулі рекомендацій і пошуку відповідають за персоналізований добір і знаходження творів. Окремо виділено модулі відгуків та коментарів, сповіщень разом із календарем виходу серій і адміністрування наповнення. Кожен модуль звертається до рівня даних через програмний інтерфейс відповідно до архітектури, і реалізує саме ті варіанти використання, що закріплені за відповідними ролями.

Визначені актори, варіанти використання та модульна структура задають функціональний каркас платформи й безпосередньо впливають на проєктування бази даних, оскільки кожна група можливостей потребує відповідних сутностей і зв'язків між ними.

2.2. Розробка бази даних

База даних вебплатформи реалізована як реляційна структура на основі СУБД PostgreSQL (хмарний сервіс Neon) із використанням ORM Prisma. Реляційна модель найкраще відповідає предметній області, де численні сутності пов'язані між собою. Модель даних охоплює користувачів, твори (аніме та мангу), персонажів і осіб, персональні списки та обране, соціальну взаємодію, а також службові підсистеми сповіщень, модерації правок і синхронізації розкладу виходу серій.

Проєктування виконано з урахуванням нормалізації, що дає змогу уникнути дублювання даних, а цілісність зв'язків між таблицями забезпечується зовнішніми ключами та обмеженнями унікальності [23]. Узагальнену ER-діаграму бази даних наведено на рис. 2.2.

Схема бази даних «AniMan»
ER-модель - PostgreSQL / Prisma

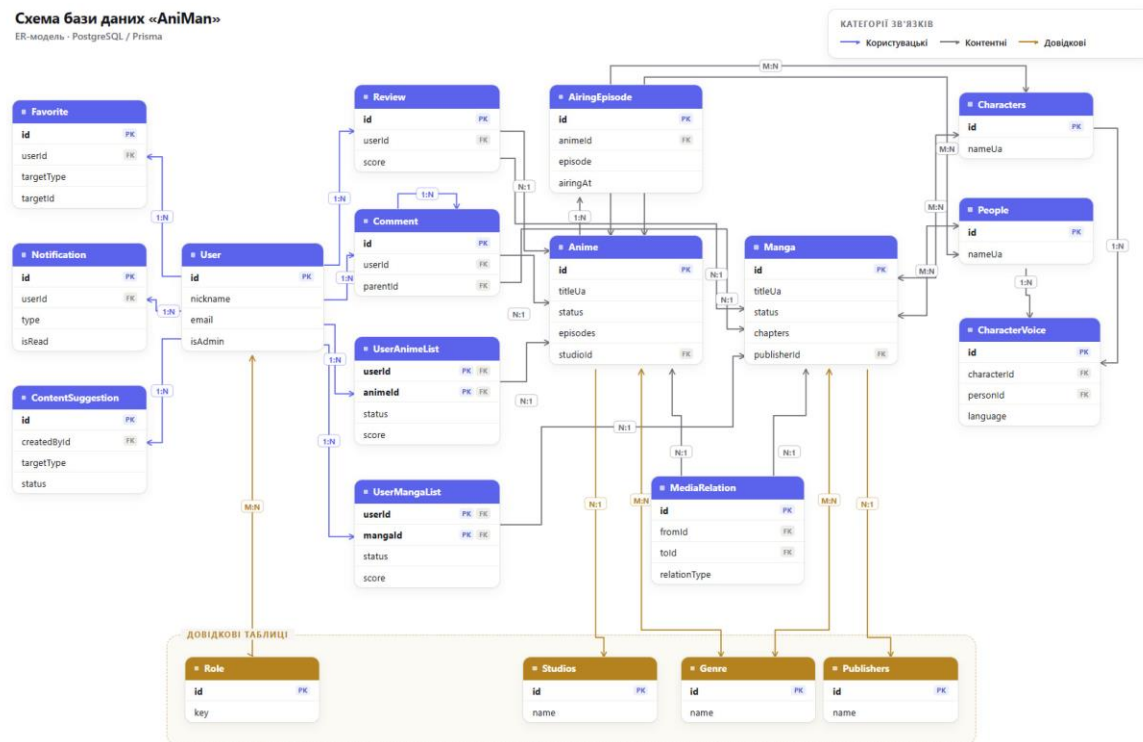


Рис. 2.2. ER-діаграма бази даних вебплатформи аніме та манги

Структуру бази даних утворює сукупність взаємопов'язаних таблиць, кожна з яких відповідає за окрему сутність предметної області. За призначенням вони поділяються на основні, довідкові та проміжні, причому довідкові й проміжні таблиці розглянуто далі в таблиці. Центральними є таблиці користувачів і творів, з якими пов'язані персонажі, особи, жанри, студії та видавництва, а також персональні списки, відгуки, коментарі, сповіщення й запропоновані правки. Саме навколо цих двох сутностей зосереджена більшість зв'язків у системі. Призначення та ключові атрибути основних сутностей наведено в табл. 2.1.

Таблиця 2.1

Основні сутності бази даних

Сутність	Призначення	Ключові атрибути	Унікальні / ключові поля
1.	2.	3.	4.
User	Облікові записи користувачів	id, nickname, email, password, imageUrl, isAdmin	nickname, email
Anime	Аніме-твори	id, titleUa, titleRomaji, titleEn, titleJp, kind, episodes, status, dateRelease, duration, studioId, anilistId	anilistId

1.	2.	3.	4.
Manga	Манга-твори	id, titleUa, titleRomaji, titleEn, titleJp, kind, chapters, volumes, status, dateRelease, publisherId	–
Characters	Персонажі	id, nameUa, nameEn, nameJp, altName, biography, birthday, imageUrl	–
People	Особи (автори, режисери, сейю)	id, nameUa, nameEn, nameJp, altName, biography, other, imageUrl	–
CharacterVoice	Озвучення персонажів	id, characterId, peopleId, language	–
UserAnimeList	Персональні списки аніме	userId, animeId, status, episodes, score, note	(userId, animeId)
UserMangaList	Персональні списки манги	userId, mangaId, status, volumes, chapters, score, note	(userId, mangaId)
MediaRelation	Зв'язки творів (франшизи)	id, fromAnimeId, fromMangaId, toAnimeId, toMangaId, relationType, sortOrder	–
AiringEpisode	Розклад виходу серій	id, animeId, episode, airingAt, notified	(animeId, episode)
Review	Відгуки	id, userId, animeId, mangaId, title, content, score	–
Comment	Коментарі	id, userId, content, reviewId, animeId, mangaId, characterId, peopleId, parentId	–
Favorite	Обране користувача	id, userId, targetType, targetId	(userId, targetType, targetId)
ContentSuggestion	Пропозиції змін контенту (модерація)	id, targetType, targetId, field, oldValue, newValue, status, createdById, reviewedById	–
Notification	Сповіщення	id, userId, type, title, body, targetType, targetId, actorId, commentId, isRead	–

Облікові записи користувачів зберігає таблиця User, що містить нікнейм, електронну пошту, хеш паролю та зображення профілю. Нікнейм і пошта унікальні. Доступ до адміністративних дій визначають ролі користувача. Ролі описують таблиці Role та UserRole, перша з яких зберігає перелік можливих ролей, а друга пов'язує користувачів із ролями зі складеним первинним ключем і дозволяє призначати одному користувачеві кілька ролей. Конкретні

можливості, як-от створення, редагування й видалення контенту чи модерація, виводяться з ролей на рівні застосунку.

Твори зберігають таблиці Anime та Manga зі схожою структурою. Запис аніме містить назви чотирма мовами, основні відомості про твір і посилання на студію, а також ідентифікатор anilistId, потрібний для синхронізації розкладу виходу серій. Таблиця Manga натомість оперує кількістю розділів і томів та посилається на видавництво.

Персонажів описує таблиця Characters з іменами трьома мовами, біографією та зображенням, а осіб, причетних до створення творів, як-от авторів, режисерів і акторів озвучення, зберігає таблиця People. Зв'язок між персонажем і актором, що його озвучує, разом із мовою озвучення фіксує таблиця CharacterVoice.

Ключову функцію платформи реалізують таблиці UserAnimeList та UserMangaList, що ведуть персональні списки. Кожен запис пов'язує користувача з твором і зберігає статус перегляду чи читання, прогрес, особисту оцінку та нотатку, а складений первинний ключ із користувача й твору гарантує, що той самий твір потрапляє до списку лише один раз.

Зв'язки між творами в межах франшизи описує таблиця MediaRelation, кожен запис якої з'єднує вихідний і цільовий твір, визначає тип зв'язку, наприклад приквел, продовження чи адаптацію, та задає порядок відображення пов'язаних творів.

Розклад виходу серій для аніме, що наразі виходять, зберігає таблиця AiringEpisode, синхронізована із зовнішнім сервісом AniList. Кожен її запис містить номер епізоду, час виходу та ознаку того, чи надіслано сповіщення, а обмеження унікальності за парою «аніме й номер епізоду» запобігає дублюванню записів про одну серію.

Відгуки про твори зберігає таблиця Review, тоді як коментарі веде таблиця Comment, записи якої можуть належати до відгуку, твору, персонажа чи особи, а посилання на батьківський коментар забезпечує деревоподібні відповіді. Обране

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	33
		Савіцький Р.С.				

користувача веде таблиця Favorite з універсальним посиланням на твір, персонажа або особу.

Запропоновані зміни до каталогу проходять через таблицю ContentSuggestion, що фіксує об'єкт і поле зміни, попереднє й нове значення, причину, статус розгляду, автора пропозиції та модератора. Тому правки вносяться лише після перевірки.

Внутрішні сповіщення зберігає таблиця Notification, кожен запис якої описує тип події, її зміст, цільовий об'єкт, користувача-ініціатора та ознаку прочитання. Ця таблиця обслуговує повідомлення про нові епізоди, відповіді на коментарі та системні події.

Окрім основних сутностей, база містить службові таблиці. Нормалізовані довідники жанрів, студій, видавництв і ролей зберігають таблиці Genre, Studios, Publishers та Role, на які посилаються основні сутності. Зв'язки «багато-до-багатьох» реалізують проміжні таблиці, серед яких UserRole, AnimeGenreOnAnime, MangaGenreOnManga, CharacterOnAnime, CharacterOnManga, PeopleOnAnime та PeopleOnManga.

Решта службових таблиць виконують вузчі завдання. Альтернативні назви й імена для пошуку зберігають MediaTitleAlias, CharacterAlias та PersonAlias. Реакції на відгуки фіксує ReviewReaction, а зворотний зв'язок і виключення для системи рекомендацій збирають RecommendationFeedback та UserContentExclusion. Дані зовнішньої автентифікації містить UserOAuthAccount, журнал пошукових запитів веде SearchQueryLog.

Між сутностями діють зв'язки різної кратності. До типу «один-до-багатьох» належать зв'язки користувача з його відгуками, коментарями, обраним, сповіщеннями та пропозиціями змін, а також аніме з епізодами розкладу виходу серій, причому відгуки й коментарі можуть стосуватися і аніме, і манги. Кратність «багато-до-багатьох» з'єднує твори з жанрами, персонажами й особами, а користувачів з ролями, і реалізується через проміжні таблиці. Окремий випадок утворюють списки UserAnimeList і UserMangaList, де користувач пов'язаний з багатьма творами, але кожна пара несе власні статуси,

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	34
		Савіцький Р.С.				

оцінку та прогрес, тому її винесено в окрему таблицю. Зв'язок твору з твором у межах франшизи описує таблиця MediaRelation, а самозв'язок таблиці Comment забезпечує деревоподібні відповіді на коментарі.

Таблиці Favorite, RecommendationFeedback та Notification використовують поліморфні зв'язки через перелік ContentType зі значеннями ANIME, MANGA, PEOPLE та CHARACTER, тож одна таблиця посилається на різні типи сутностей і не потребує окремого варіанта для кожного. Складені первинні ключі таблиць UserAnimeList, UserMangaList, UserRole та проміжних таблиць унеможливлюють дублювання зв'язків, а зовнішні ключі з каскадним видаленням прибирають залежні записи разом із батьківськими, чим підтримується загальна цілісність даних. Синхронізацію з AniList забезпечують поле anilistId і таблиця AiringEpisode.

Архітектура передбачає й користувацькі колекції на основі таблиць Collection та CollectionItem, проте їх реалізацію віднесено до напрямів подальшого розвитку платформи.

2.3. Огляд методів рекомендаційних систем

Рекомендаційні системи покликані долати інформаційне перевантаження, допомагаючи користувачеві знаходити релевантний контент серед великої кількості варіантів [11]. Для платформи аніме та манги, де обсяг творів значний і постійно зростає, якість рекомендацій безпосередньо впливає на користувацький досвід та утримання аудиторії. Тому перед проєктуванням власного механізму доцільно розглянути основні підходи до формування рекомендацій, їхні переваги й обмеження. Розрізняють три основні підходи, як-от контентну фільтрацію, колаборативну фільтрацію та гібридний підхід [11, 12].

Контентна фільтрація ґрунтується на ознаках самих об'єктів і рекомендує твори, схожі на ті, що вже сподобалися користувачеві, за такими характеристиками, як жанри, тип чи рейтинг. Для цього кожен твір подають у вигляді набору ознак, а близькість між творами та профілем уподобань користувача обчислюють за мірами подібності, найпоширенішою з яких є

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	35
		Савіцький Р.С.				

косинусна [17]. Щоб зменшити вплив надто поширених ознак, їхню вагу часто коригують за оберненою частотою, завдяки чому рідкісні жанри вважаються інформативнішими за загальнопоширені [18]. Підхід не залежить від даних інших користувачів, може рекомендувати нові або маловідомі твори й дає пояснюваний результат. Водночас контентна фільтрація схильна до «бульбашки фільтрів», коли рекомендації обмежуються вже відомими користувачеві категоріями, потребує якісних метаданих про твори й погано працює для нових користувачів без історії вподобань через проблему холодного старту.

Колаборативна фільтрація ґрунтується на подібності поведінки користувачів і рекомендує твори, які високо оцінили інші користувачі зі схожими смаками. Цей підхід поділяють на дві групи. Методи, що спираються на пам'ять, безпосередньо обчислюють подібність між користувачами або між творами на основі матриці оцінок, застосовуючи косинусну міру чи коефіцієнт кореляції, і поділяються на орієнтовані на користувача та орієнтовані на товар [14]. Методи, що спираються на модель, натомість навчають узагальнену модель уподобань, найвідомішим прикладом якої є матрична факторизація, що подає користувачів і твори у спільному просторі прихованих ознак [15]. Колаборативна фільтрація добре виявляє несподівані, але релевантні твори поза звичним колом інтересів і не потребує детальних метаданих. Стримують її холодний старт для нових користувачів і творів, розрідженість матриці оцінок та схильність надавати перевагу популярному.

Гібридний підхід поєднує контентну та колаборативну фільтрацію, щоб компенсувати слабкі сторони кожного з методів. Існує кілька способів такого поєднання. Зважена схема обчислює підсумкову оцінку як комбінацію оцінок обох методів із певними вагами, схема перемикавання обирає метод залежно від ситуації, наприклад застосовує контентну фільтрацію для нових користувачів, а змішана схема показує результати обох методів одночасно. Поширеним є також комбінування ознак, коли дані про поведінку користувачів додають як додаткові ознаки до контентної моделі. Завдяки поєднанню контентна складова забезпечує рекомендації навіть за браку оцінок і пояснюваність, а колаборативна додає

різноманітності та виявляє неочевидні зв'язки, тому гібридні системи зазвичай досягають вищої точності й стабільності рекомендацій [12].

Окрім трьох основних підходів, у літературі розглядають і допоміжні. Знаннєві системи спираються на явно задані правила та вимоги користувача, а демографічні групують аудиторію за віком, регіоном чи іншими ознаками. Самостійно їх застосовують рідко, проте окремі їхні ідеї входять до складу гібридних рішень. Саме гібридні системи поступово стали галузевим стандартом, бо поєднують сильні сторони різних методів і дають змогу гнучко налаштовувати поведінку під конкретну предметну область.

Незалежно від підходу, на якість рекомендацій впливає кілька наскрізних чинників. Проблема холодного старту, за якої для нових користувачів чи творів бракує даних, частково розв'язується контентною складовою та початковим опитуванням уподобань. Важливе значення має різноманітність видачі, оскільки надмірно однорідні рекомендації знижують свою цінність, тому застосовують методи повторного впорядкування, що балансують релевантність і новизну, як-от принцип максимальної граничної релевантності [16]. Крім того, оцінки творів доцільно нормалізувати з урахуванням кількості голосів, щоб уникнути завищення позицій творів із малою, але високою вибіркою оцінок. Якість роботи рекомендаційних систем оцінюють за такими показниками, як точність, повнота й покриття, що дають змогу об'єктивно порівнювати різні методи між собою.

На роботу будь-якого підходу впливає і характер даних, на яких він навчається. Явний зворотний зв'язок користувач надає свідомо, через оцінки чи додавання творів до обраного, тому він точно відображає вподобання, хоча його зазвичай небагато. Неявний зворотний зв'язок система збирає з поведінки, наприклад з переглядів, додавання до списків і відстеження прогресу, і його значно більше, проте тлумачити його складніше. Через це сучасні системи спираються переважно на неявні сигнали й доповнюють їх явними там, де вони доступні, що дає змогу формувати рекомендації навіть для користувачів, які майже не виставляють оцінок.

Предметна область аніме та манги має особливості, які впливають на вибір методу. Твори часто об'єднані у франшизи з приквелами, продовженнями й адаптаціями, тому пропонувати наступну частину доречно лише після ознайомлення з попередньою. Багато творів виходять сезонно, через що актуальність новинок швидко змінюється, а спільні студії, видавництва та причетні особи слугують додатковими ознаками подібності поза жанрами. Велика кількість структурованих метаданих робить контентну складову особливо придатною для цієї області, тоді як активна спільнота з оцінками й списками дає достатньо даних і для колаборативної складової.

Порівняння розглянутих методів за ключовими критеріями наведено в табл. 2.2. Для вебплатформи аніме та манги обрано гібридний підхід, що поєднує контентну фільтрацію за жанрами та властивостями творів і колаборативну фільтрацію за подібністю вподобань користувачів з можливістю налаштування ваг кожної складової. Це забезпечує релевантні рекомендації як для нових користувачів за рахунок контентної складової, так і для активних учасників за рахунок колаборативної, що відповідає визначеним вимогам до системи.

Таблиця 2.2

Порівняння методів формування рекомендацій

Метод	Переваги	Недоліки
Контентна фільтрація	Незалежність від інших користувачів; пояснюваність; рекомендації нових творів	«Бульбашка фільтрів»; потреба в метаданих; холодний старт
Колаборативна фільтрація	Виявлення несподіваних творів; не потребує метаданих	Холодний старт; розрідженість даних; перебіс до популярного
Гібридний підхід	Вища точність; пом'якшення холодного старту; різноманітність	Складніша реалізація та налаштування

2.4. Проєктування та реалізація алгоритмів роботи

Рекомендаційна система гібридна, бо поєднує контентну фільтрацію, що спирається на властивості творів і профіль уподобань користувача, з колаборативною, яка враховує поведінку схожих користувачів.

Рекомендації спираються на неявні сигнали з поведінки користувача, тобто зі списків аніме й манги зі статусом, прогресом та оцінкою й обраного, і на явні,

які користувач задає сам через оцінки, відгуки та реакції під відгуками. Кожен сигнал дістає вагу за статусом і оцінкою твору, тож завершені та високо оцінені важать найбільше, заплановані помірно, а покинуті отримують від'ємну вагу. Твори з позначками «не цікавить» і «приховати» зникають із видачі зовсім.

Зі зважених сигналів складається профіль уподобань, у якому ваги жанрів обчислюються за оберненою частотою, тож рідкісні жанри впливають сильніше за поширені, а поряд враховуються студії, видавництва, формат і причетні особи. За цим профілем система добирає до тисячі кандидатів і відкидає вже переглянуте, покинуте, попередні частини франшиз та виключені користувачем жанри.

Кожен кандидат проходить контентне оцінювання за зваженою сумою складових, де найбільше важить жанрова спорідненість, а далі відповідність формату, студія чи видавництво, належність до франшизи переглянутих творів, рік виходу, популярність і оцінка якості з поправкою на кількість оцінок. Паралельно виконується колаборативне оцінювання, під час якого система знаходить схожих користувачів за косинусною мірою на векторах уподобань і агрегує їхні позитивні оцінки з вагою, пропорційною схожості. Обидві оцінки нормалізують до спільного діапазону й адаптивно поєднують, причому за замовчуванням переважає контентна складова, а зі зростанням насиченості профілю внесок колаборативної зростає.

Зі сотні найкраще оцінених кандидатів формується підсумкова видача, а оцінки переводяться у відсотки для зручнішого сприйняття. На основі переліку будуються тематичні добірки, персональна «Для вас» із повторним упорядкуванням за різноманітністю жанрів, «Приховані перлини» з якісними, але маловідомими творами, а також «Цього сезону», «Кросмедійні адаптації» та «Профіль смаку».

Видача уточнюється зі зростанням активності користувача, адже кожна нова оцінка, доданий твір чи позначка «не цікавить» змінюють профіль. Новим користувачам без історії пропонується початкове опитування жанрових

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	39
		Савіцький Р.С.				

уподобань, що дає осмислену видачу від першого сеансу й поступово переводить рекомендації від переважно контентних до повноцінно гібридних.

Узагальнений алгоритм формування рекомендацій наведено на рис. 2.3.

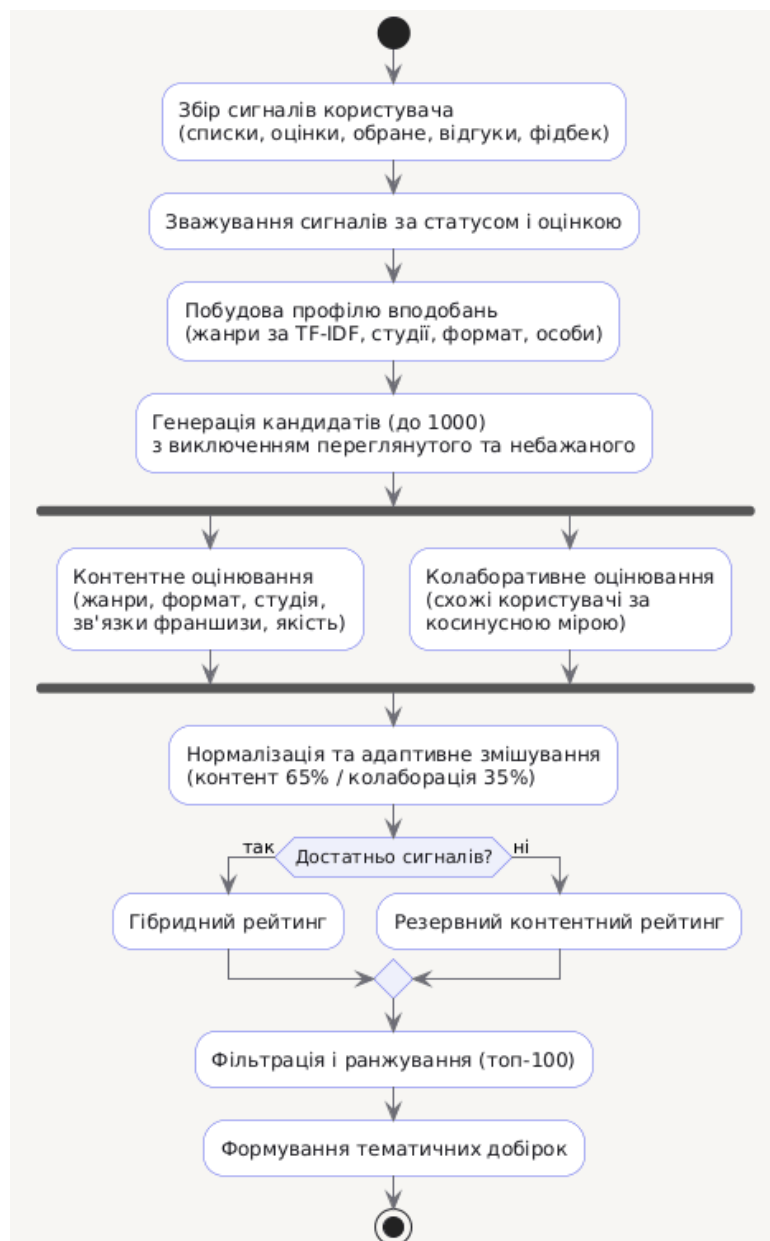


Рис. 2.3. Алгоритм формування персоналізованих рекомендацій

Інтелектуальний пошук перетворює сформульований природною мовою запит на структуровану вибірку творів із каталогу. Працює він у двох режимах. Звичайний виконує запит до бази за нормалізованими назвами творів і їхніми синонімами (MediaTitleAlias), а також за іменами осіб, назвами студій і видавництв із реєстронезалежним частковим збігом. Інтелектуальний дозволяє

формулювати запит звичайною мовою, наприклад «аніме про піратів із гумором», і доручає його тлумачення великій мовній моделі.

Перед зверненням до моделі система перевіряє кеш, і якщо той самий користувач уже надсилав ідентичний запит протягом останньої години, повертає збережений результат із журналу пошукових запитів без повторного виклику моделі. Звернення до моделі обмежені за частотою, тому за перевищення допустимої кількості запитів за хвилину чи добу система робить тимчасову паузу.

Запит обробляє ланцюжок постачальників із резервуванням, де основним є OpenAI, а за його недоступності працює OpenRouter. Системна інструкція скеровує модель розпізнати тип запиту. Концептуальний описує сюжет або настрій, атрибутивний задає жанр, рік, статус, тип або студію, а решта запитів вважаються змішаними чи нерелевантними. У відповідь надходять структуровані дані у форматі JSON із текстовими блоками, посиланнями на конкретні твори та, за потреби, набором фільтрів каталогу.

Назви творів, осіб, студій і жанрів із відповіді моделі зіставляються з реальними записами бази евристикою добору найкращого збігу, після чого виконується відфільтрований запит і формується попередній перегляд результатів. Сам запит, інтерпретовані фільтри, кількість знайдених результатів і назву використаної моделі записують у журнал для подальшого кешування та аналітики.

Модель тут лише тлумачить запит, а остаточну вибірку формує сама система за фактичними записами бази, тому вигадані твори в результати не потрапляють. Кешування ідентичних запитів і обмеження частоти стримують витрати на зовнішні сервіси, а резервування постачальників тримає пошук доступним навіть за відмови основної моделі.

Послідовність роботи інтелектуального пошуку наведено на рис. 2.4.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	41
		Савіцький Р.С.				

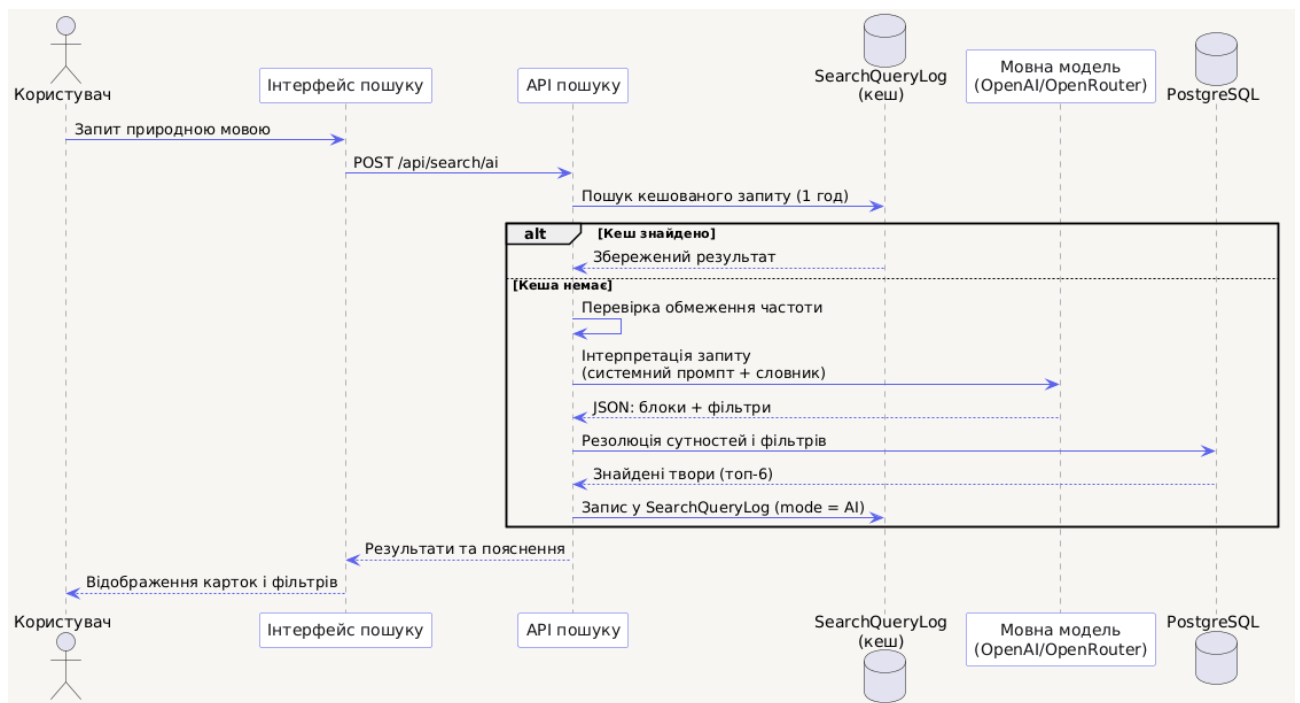


Рис. 2.4. Послідовність роботи інтелектуального пошуку

Автентифікація та авторизація відповідають за безпечний доступ користувачів до персональних можливостей платформи. Автентифікацію побудовано на сесійних токенах. Після успішного входу формується підписаний токен з ідентифікатором користувача, ознакою адміністратора та часом завершення дії. Підпис обчислюється за алгоритмом HMAC-SHA256 [26] із секретним ключем, що зберігається у змінних середовища, а сам токен записується в захищений cookie з ознаками httpOnly, sameSite і secure та діє 30 днів. Паролі система зберігає у вигляді хешу, обчисленого алгоритмом bcrypt [27] з індивідуальною сіллю, а під час входу порівнює хеші в режимі, стійкому до атак за часом (timingSafeEqual).

Передбачено й автентифікацію через зовнішні сервіси Google та X за протоколом OAuth 2.0 [24] із розширенням PKCE [25]. Після підтвердження особи постачальником система перевіряє наявність прив'язаного облікового запису в таблиці UserOAuthAccount і за потреби створює нового користувача або зв'яже зовнішній обліковий запис із наявним за електронною поштою.

Під час кожного захищеного запиту токен зчитується з cookie, перевіряються його підпис і термін дії, після чого з бази завантажуються дані

користувача. Недійсний токен повертає користувача на сторінку входу. Послідовність автентифікації за логіном і паролем наведено на рис. 2.5.

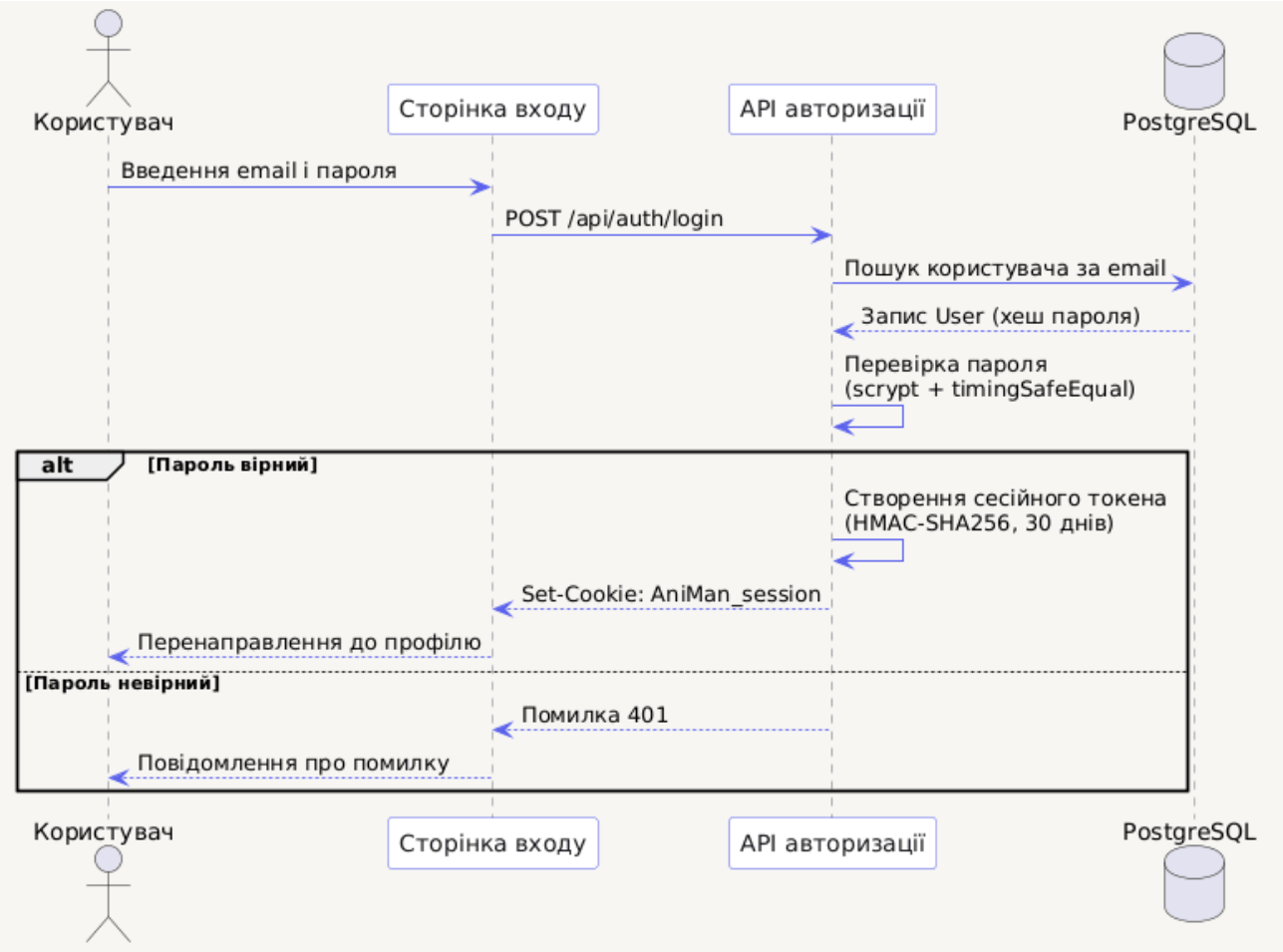


Рис. 2.5. Послідовність автентифікації користувача

Актуальність даних про вихід нових серій підтримує окремий фоновий алгоритм синхронізації із зовнішнім сервісом AniList. Раз на добу система запитує розклад виходу серій і оновлює таблицю AiringEpisode. Щогодини виконується завдання формування сповіщень, яке відбирає серії з уже насталим часом виходу та ще не обробленим прапорцем (notified = false) і для кожного користувача з відповідним аніме у списку створює сповіщення типу ANIME_NEW_EPISODE. Оброблені серії позначаються прапорцем notified, що запобігає повторному надсиланню. Алгоритм наведено на рис. 2.6.

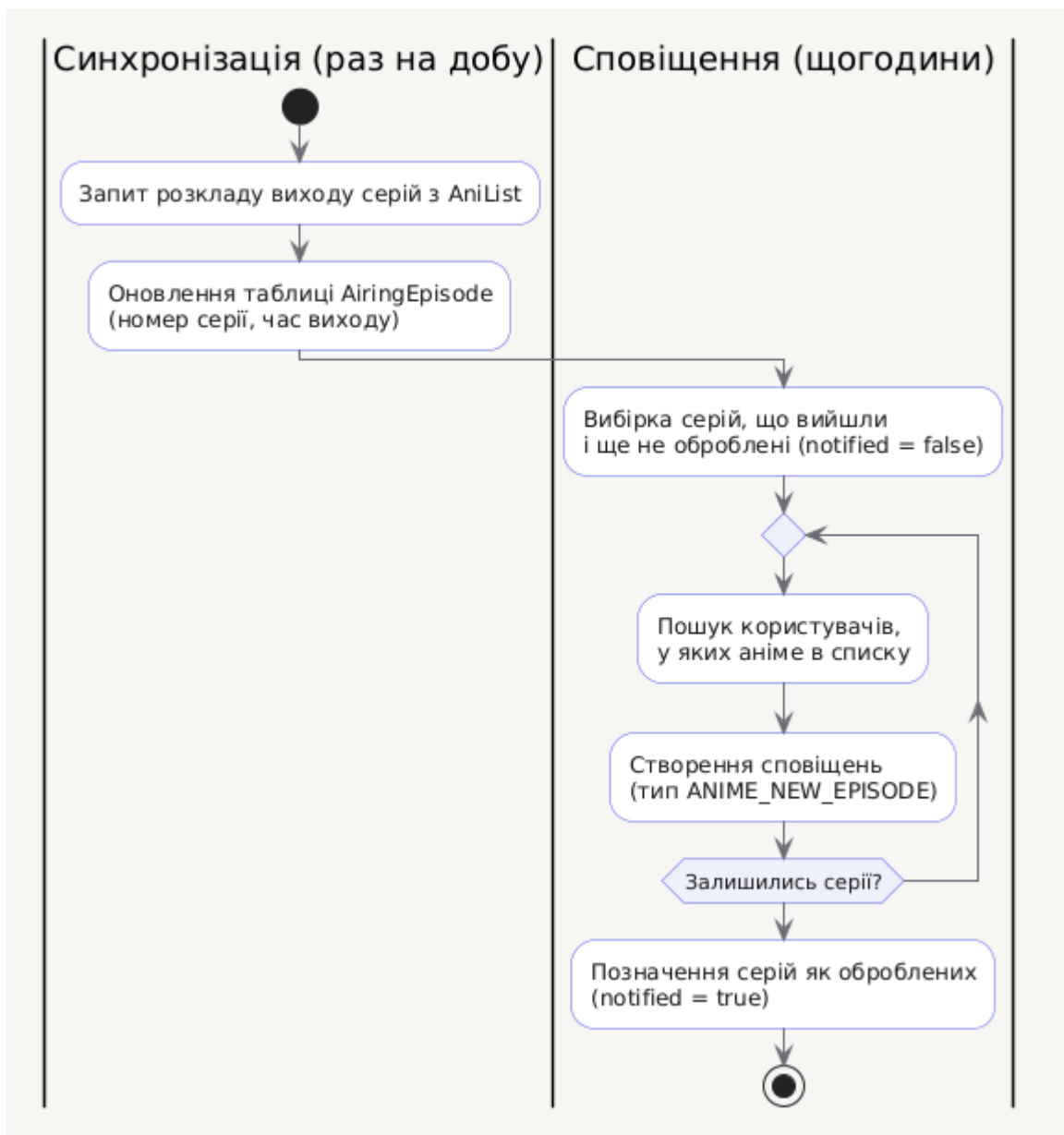


Рис. 2.6. Алгоритм синхронізації розкладу та формування сповіщень

2.5. Реалізація вебплатформи аніме та манги

Вебплатформу реалізовано як повностековий застосунок на основі фреймворку Next.js з App Router і серверним рендерингом. Код організовано за гібридним підходом, що поєднує функціональні теки із шаровою структурою [30]. Кожна функціональна можливість інкапсулює власну логіку доступу до даних, допоміжні функції та компоненти, тоді як спільні утиліти й примітиви винесено в окремі шари. Узагальнену структуру проєкту наведено на рис. 2.7.



Рис. 2.7. Структура проєкту вебплатформи

Сторінки платформи реалізовано як асинхронні серверні компоненти React, які виконують вибірку даних безпосередньо під час рендерингу на сервері. Це зменшує обсяг коду, що передається клієнту, і покращує продуктивність та показники SEO. Незалежні запити виконуються паралельно за допомогою Promise.all.

Усі звернення до бази даних зосереджено в окремому шарі доступу до даних (lib/data/ та features/*/data/), що відокремлює логіку запитів від

представлення. Клієнт Prisma створюється за патерном «singleton» [31], що запобігає вичерпанню пулу з'єднань під час розробки (лістинг A.1). Функції доступу до даних інкапсулюють запити Prisma та повертають уже підготовлені для відображення структури (лістинг A.2).

Операції зміни даних (створення та редагування записів, надсилання форм) реалізовано через обробники маршрутів (`app/api/**/route.ts`) за методами POST і PUT [20]. Операції зміни контенту захищено перевіркою можливості `requireCapability` відповідно до дії (створення, редагування чи видалення). Такий поділ забезпечує чітке розмежування читання (серверні компоненти) і запису (API-маршрути).

Як приклад операції зміни даних, оновлення запису персонального списку виконується методом PUT з перевіркою автентифікації, валідацією вхідних даних та операцією `upsert` за складеним ключем (`userId, animeId`) (лістинг A.3).

Захист доступу реалізовано за принципом багаторівневого захисту [28]. На рівні проміжного програмного забезпечення перехоплюються запити до захищених, адміністративних і авторизаційних маршрутів, де перевіряється підпис сесійного токена (HMAC-SHA256 через Web Crypto API), після чого неавтентифікованих користувачів перенаправляють на сторінку входу зі збереженням цільового шляху, а користувачів без потрібної можливості перенаправляють на головну сторінку (лістинг A.4).

Права згруповано в можливості (`capabilities`), а кожна роль отримує визначений набір можливостей. Перевірки в інтерфейсі та на сервері звертаються саме до можливості, тому додавання нової ролі не змінює наявних перевірок.

Додатково кожен серверний компонент повторно визначає поточного користувача через функцію `getCurrentUser`, яка зчитує токен із `cookie` та завантажує дані з бази. Таким чином перевірка відбувається і на межі застосунку, і безпосередньо під час рендерингу, що унеможливорює доступ до захищеного вмісту в обхід `middleware`.

Інтерактивні елементи (фільтри каталогу, форми, панель інтелектуального пошуку) реалізовано як клієнтські компоненти (директива `"use client"`), тоді як

решта інтерфейсу залишається серверною. Глобальний стан списків користувача передається через контекст React (UserListStatusProvider).

Стилізацію виконано засобами Tailwind CSS 4 із конфігурацією на основі CSS, без окремого файлу налаштувань. Палітру та інші дизайн-токени визначено як CSS-змінні й відображено в утиліти Tailwind через директиву @theme (лістинг А.5), що забезпечує єдиний візуальний стиль платформи.

Бібліотека перевикористовуваних компонентів (components/ui/) містить кнопки, картки, аватари, позначки, хлібні крихти, вкладки та блоки порожніх станів. Інтерфейс адаптивний за підходом «спершу мобільні» (mobile-first) із контрольними точками Tailwind, завдяки чому сітки та елементи керування коректно відображаються на пристроях різного розміру.

Платформа інтегрована з кількома зовнішніми сервісами. Сервіс Cloudinary забезпечує зберігання та оптимізацію зображень творів, персонажів, осіб і аватарів користувачів, а зображення з профілів OAuth також імпортуються до нього. Сервіс AniList є джерелом розкладу виходу серій, а синхронізація виконується скриптом, що звертається до його GraphQL-API й оновлює таблицю AiringEpisode. Мовні моделі OpenAI та OpenRouter використовуються для інтелектуального пошуку через ланцюжок провайдерів із резервуванням та обмеженням частоти запитів. Зовнішню автентифікацію реалізовано через сервіси Google та X за протоколом OAuth 2.0.

Якість коду підтримується статичним аналізатором ESLint та інструментом Husky, який через commitlint забезпечує дотримання угоди про формат повідомлень комітів (Conventional Commits). Коректність ключових модулів (рекомендації, каталог, пошук, календар) перевіряється модульними тестами на Vitest.

Окремо реалізовано набір службових скриптів для супроводу даних і алгоритмів, виявлення дублікатів творів, аудит повноти заповнення записів, синхронізація розкладу з AniList, а також офлайнове оцінювання рекомендаційної системи. Останній скрипт приховує частину взаємодій кожного користувача, формує рекомендації без них і обчислює метрики якості (влучність,

повноту, частку влучань та середній обернений ранг), що дозволяє об'єктивно оцінювати й калібрувати алгоритм рекомендацій.

Рекомендаційна система є найскладнішим модулем платформи. Її реалізовано за принципом єдиної відповідальності, тобто близько двадцяти невеликих модулів у теці `features/recommendations/data/`, кожен з яких розв'язує одну задачу, а оркестратор `getUserRecommendations` поєднує їх у конвеєр.

До основних модулів належать `implicitSignals` та `recommendationWeights` (зважування сигналів), `userPreferenceProfile` (побудова профілю вподобань), `genreAffinity` (жанрова спорідненість), `animeRecommendations` і `mangaRecommendations` (генерація кандидатів та контентне оцінювання), `collaborativeSimilarity` (колаборативна фільтрація), `hybridBlend` і `hybridScore` (змішування каналів), `popularityScore` та `qualityScore` (популярність і якість), `mmrRerank` (різноманіття), `hiddenGems`, `season`, `crossMediaAdaptations`, `coldStart` (тематичні добірки) і `recommendationReasons` (пояснення рекомендацій).

Профіль користувача будується зі зважених взаємодій. Функція `getImplicitListWeight` визначає вагу твору за статусом у списку, оцінкою та часткою перегляду, причому явна оцінка має пріоритет, повторний перегляд вважається найсильнішим позитивним сигналом, а ранню відмову (переглянуто менше 20 %) трактовано як виражений негатив (лістинг А.6).

Підсумковий контентний бал кандидата є зваженою сумою ознак. Вагові коефіцієнти для аніме наведено в таблиці 2.3 (для манги застосовано окремий набір, де замість студії вагомішими є автори, довжина та видавництво).

Таблиця 2.3

Вагові коефіцієнти ознак для аніме-рекомендацій

Ознака	Ваговий коефіцієнт
1.	2.
Жанрова спорідненість	0.30
Сигнал оцінок	0.20
Статус у списку	0.15
Зв'язки франшизи	0.15
Тип твору	0.07
Студія	0.05
Популярність	0.035

1.	2.
Рік виходу	0.03
Якість (з поправкою на впевненість)	0.015

Щоб широкі жанри, наприклад «Екшн», присутній у половині каталогу, не домінували, вагу кожного жанру визначають за оберненою частотою документів

$$idf(g) = \log\left(\frac{N}{df(g)}\right), \quad (2.1)$$

де $idf(g)$ – обернена частота жанру g ;

N – загальна кількість творів у каталозі;

$df(g)$ – кількість творів, що містять жанр g .

Жанрову прихильність кандидата обчислюють як суму ваг його жанрів із логарифмічним демпфуванням, що запобігає частотному домінуванню

$$A(c) = \sum_{g \in G_c} \log(1 + w(g)) \cdot idf(g), \quad (2.2)$$

де $A(c)$ – жанрова прихильність кандидата c ;

G_c – множина жанрів кандидата;

$w(g)$ – невід'ємна вага жанру g у профілі користувача;

$idf(g)$ – за формулою (2.1).

Деталі реалізації наведено в лістингу Б.7

Схожість між користувачами обчислюють як косинусну міру між векторами зважених уподобань, помножену на коефіцієнт довіри, що залежить від кількості спільних творів і знижує вплив випадкових збігів за малого перетину списків

$$sim(u, v) = \frac{\sum_i u_i v_i}{\|u\| \|v\|} \cdot \frac{m}{m+d}, \quad (2.3)$$

де $sim(u, v)$ – схожість користувачів u і v ;

u_i, v_i – ваги i -го твору в їхніх векторах уподобань;

$\|u\|, \|v\|$ – евклідові норми векторів;

m – кількість спільно оцінених творів;

d – коефіцієнт демпфування довіри ($d = 2$).

До агрегації беруться лише 30 найсхожіших користувачів (лістинг А.8).

Частка контентного каналу не фіксована, а залежить від активності користувача, тому новачкам система рекомендує переважно за жанрами, а активним користувачам за подібністю до інших людей. Базова частка спадає з активністю за законом

$$k = \frac{1}{1+n/halfLife}, \quad (2.4)$$

де k – базова частка контентного каналу до обмеження заданими межами;

n – кількість врахованих взаємодій користувача;

$halfLife$ – параметр, що задає темп спадання частки.

Отримане значення обмежують мінімальною та максимальною частками каналу. Нормалізовані контентний і колаборативний канали поєднують у підсумковий бал адаптивними вагами

$$S = \frac{S_c w_c + S_k w_k}{w_c + w_k}, \quad (2.5)$$

де S – підсумковий гібридний бал;

S_c, S_k – нормалізовані контентний і колаборативний бали;

w_c, w_k – ваги відповідних каналів.

Для аніме ввімкнено резервування на контент, а для манги його вимкнено (емпірично визначено калібратором) (лістинг А.9).

Функція `getUserRecommendations` спочатку паралельно (через `Promise.all`) зчитує всі сигнали користувача, списки, відгуки, улюблене, реакції, явний зворотний зв'язок та зв'язки творів. На їх основі будуються карти ваг жанрів, студій, осіб і окремих творів. Далі генеруються кандидати з виключенням уже відомих, кинутих творів і попередніх частин франшиз, після чого кожен кандидат оцінюється функціями `rankAnime` / `rankManga` (контентний бал, колаборативний бал і пояснення). Підсумкові бали нормалізуються у відсотки відносно найкращого результату видачі.

Щоб у видачі поруч не стояли однотипні твори, застосовано переранжування за принципом максимальної граничної релевантності. На кожному кроці обирається твір із найбільшим значенням

$$MMR(c) = \lambda \frac{rel_c}{rel_{max}} - (1 - \lambda) \max_{d \in s} J(G_c, G_d), \quad (2.6)$$

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	50
		Савіцький Р.С.				

де $MMR(c)$ – оцінка кандидата c під час переранжування;

λ – баланс релевантності й різноманіття ($\lambda = 0,7$);

rel_c – релевантність кандидата, rel_{max} – найбільша серед кандидатів;

S – множина вже відібраних творів;

$J(G_c, G_d)$ – коефіцієнт Жаккара між наборами жанрів, $J(A, B) = |A \cap B| / |A \cup B|$.

Додатково формуються добірки «Приховані перлини» (висока оцінка якості за низької популярності), «Цей сезон для тебе», «Кросмедійні адаптації» та «Профіль смаку».

Оркестратор приймає набори прихованих ідентифікаторів творів, що використовуються скриптом офлайнного оцінювання^б за якого частина взаємодій тимчасово вилучається з профілю, після чого перевіряється, чи рекомендує система приховані твори. Це дозволяє вимірювати якість алгоритму метриками влучності, повноти та середнього оберненого рангу.

Інтелектуальний пошук дозволяє формувати запити природною мовою. Модуль складається з двох частин, теки `features/search` (оркестрація `aiSearch`, розбір відповіді `aiParse`, запити до бази `searchSources`, словник допустимих значень `vocabulary`) та теки `lib/ai`, що абстрагує роботу з мовними моделями (інтерфейс провайдера, реалізації для `OpenAI` та `OpenRouter`, ланцюг резервування, обмеження частоти й конфігурація).

Модель отримує системний промпт, який класифікує запит як нерелевантний, концептуальний (за сюжетом чи описом), атрибутивний (жанри, рік, статус, тип, студія) або змішаний, і зобов'язує повернути відповідь у форматі JSON згідно зі строгою схемою. Схема (`RESPONSE_SCHEMA`) задає поля `relevant`, `message`, `blocks` і `filters` та контролюється на рівні моделі, що унеможливорює довільний формат відповіді. У промпт вбудовується словник допустимих значень, тож модель добирає лише наявні в каталозі жанри й статуси (лістинг А.10).

Виклик моделі виконується через ланцюг провайдерів із резервуванням, де спершу йде платний сервіс `OpenAI` (за наявності ключа), далі безкоштовні моделі `OpenRouter`. Тимчасові помилки (код 429 або 5xx, перевищення часу очікування)

спричиняють один повтор із затримкою; після помилки 429 провайдер тимчасово вимикається на 30 секунд; постійні помилки одразу передають керування наступному провайдеру (лістинг А.11). Це забезпечує стійкість пошуку до збоїв окремого сервісу.

Оскільки звернення до мовних моделей є платними й обмеженими квотою, реалізовано кешування та контроль частоти. Якщо той самий користувач робив ідентичний запит протягом останньої години, повертається збережений результат без виклику моделі. Обмеження частоти ґрунтується на підрахунку записів у таблиці SearchQueryLog, тому коректно працює в безсерверному середовищі з кількома екземплярами застосунку. Передбачено три пороги (за хвилину, за добу на користувача та глобальний за добу), що налаштовуються через змінні середовища. Слот резервується (запис створюється) ще до виклику моделі, що усуває стан гонки за паралельних запитів (лістинг А.12).

Назви творів, студій, видавництв та осіб, повернуті моделлю, не використовуються напряму, а зіставляються з реальними записами бази даних. Спочатку виконується точний пошук, а за відсутності результату пошук за окремими словами, після чого евристика добору обирає найкращий збіг (лістинг А.13). Атрибутивні фільтри так само перетворюються на ідентифікатори й формують посилання на відфільтрований каталог із попереднім переглядом. Тому система не показує неіснуючих творів, навіть якщо модель помилилася в назві.

Висновки до другого розділу

У другому розділі виконано проєктування та програмну реалізацію вебплатформи аніме та манги. На основі сформованих у першому розділі вимог визначено акторів системи, побудовано модель варіантів використання й деталізовано ключові сценарії взаємодії, а також окреслено узагальнену функціональну структуру платформи з поділом на взаємодійні модулі.

Спроектовано базу даних як реляційну структуру на основі СУБД PostgreSQL із використанням ORM Prisma. Визначено основні сутності

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	52
		Савіцький Р.С.				

предметної області, як-от користувачів, твори, персонажів, осіб, персональні списки, відгуки, зв'язки франшиз, розклад виходу серій і пропозиції модерації – та зв'язки між ними. Структуру побудовано з урахуванням нормалізації, цілісності даних і розширюваності, наприклад через застосування проміжних таблиць і поліморфних зв'язків.

Спроектовано та реалізовано ключові алгоритми платформи. Гібридна рекомендаційна система поєднує контентну та колаборативну фільтрацію з адаптивним змішуванням каналів, IDF-зваженою жанровою спорідненістю, косинусною мірою схожості користувачів і переранжуванням за різноманіттям. Інтелектуальний пошук інтерпретує запити природною мовою за допомогою мовної моделі зі структурованим виводом, прив'язкою результатів до реального каталогу, кешуванням і обмеженням частоти. Окремо реалізовано багаторівневу автентифікацію та механізм синхронізації розкладу із формуванням сповіщень.

Реалізацію виконано на сучасному стеку технологій: Next.js (App Router) із серверними компонентами React, TypeScript, Prisma та Tailwind CSS. Застосовано гібридну архітектуру, що поєднує функціональні модулі із шаровим доступом до даних, реалізовано багаторівневий захист маршрутів, інтегровано зовнішні сервіси (Cloudinary, AniList, мовні моделі, OAuth) та впроваджено засоби забезпечення якості коду – статичний аналіз, контроль формату комітів, модульне тестування й офлайнне оцінювання рекомендаційної системи.

РОЗДІЛ 3. ТЕСТУВАННЯ ТА РОЗГОРТАННЯ ВЕБПЛАТФОРМИ

3.1. Розгортання та безперервна інтеграція

Щоб зробити процес оновлення платформи передбачуваним і безпечним, його автоматизовано за методологією безперервної інтеграції та доставки [29], яка усуває ручні дії, забезпечує однакове середовище для кожного запуску й дає змогу виявляти помилки ще до того, як вони потраплять у робочу версію. Конвеєр реалізовано на платформі GitLab CI/CD, тісно інтегрований із системою контролю версій проєкту.

Опис конвеєра винесено у файл `.gitlab-ci.yml`, а окремі завдання згруповано в каталозі `.gitlab/ci` за призначенням, що спрощує супровід і дає змогу змінювати окрему стадію, не зачіпаючи решти. Усі завдання виконуються в наперед визначеному середовищі на основі Node.js, а встановлені залежності кешуються між запусками за контрольною сумою файлу `package-lock.json`. Кешування усуває повторне встановлення пакетів і помітно пришвидшує проходження конвеєра.

Основний конвеєр запускається автоматично під час надсилання змін до головної гілки або створення запиту на злиття й складається з п'яти послідовних стадій. На першій стадії виконується статичний аналіз коду лінтером ESLint, що виявляє стилістичні й потенційно небезпечні конструкції ще до запуску тестів. Якщо аналіз не проходить, подальші стадії не виконуються, тож проблемний код не потрапляє до збірки.

На стадії тестування спершу перевіряються типи компілятором TypeScript [21, 22] у режимі без генерації коду, що гарантує узгодженість типів у всьому проєкті. Після цього запускаються модульні тести з вимірюванням покриття, відсоток якого автоматично зчитується з виводу й відображається індикатором покриття безпосередньо в репозиторії. Інтеграційні тести, що потребують підключення до бази даних, винесено в окреме завдання з ручним запуском, оскільки вони залежать від зовнішнього середовища й не повинні блокувати основний конвеєр.

На стадії збірки виконуються генерація клієнта Prisma та промислова збірка застосунку Next.js. Успішна збірка слугує контрольним бар'єром, адже якщо проєкт не збирається, розгортання не відбувається. Окрему стадію відведено для роботи з базою даних, на якій зміни схеми застосовуються до бази Neon командою `prisma db push`. Цю стадію свідомо зроблено ручною та без примусового застосування руйнівних змін, щоб виключити випадкову втрату даних під час оновлення структури бази.

Завершальна стадія розгортання публікує застосунок у хмарному середовищі Vercel і виконується лише для головної гілки після успішного проходження всіх попередніх стадій. Розгортання здійснюється через офіційний інструмент Vercel, який отримує доступ за захищеним токеном зі сховища захищених змінних конвеєра. Тому в робочу версію потрапляє лише код, який пройшов перевірку, тестування та збірку.

Конфіденційні параметри, потрібні для роботи конвеєра, рядок підключення до бази даних і токен розгортання, зберігаються не у вихідному коді, а серед захищених змінних середовища GitLab із позначками маскування та захисту. Це унеможливлює їх потрапляння до журналів виконання чи до сторонніх осіб.

Окрім основного конвеєра, налаштовано заплановані конвеєри, що запускаються за розкладом незалежно від внесення змін у код. Один із них щодоби синхронізує розклад виходу серій із зовнішнім сервісом AniList та оновлює відповідні записи в базі даних, а інший щогодини ініціює розсилання сповіщень про нові епізоди, звертаючись до службового маршруту застосунку із захищеним секретом. Винесення цих регламентних задач в окремі заплановані конвеєри відокремлює фонові операції від процесу доставки коду й дає змогу керувати їхнім розкладом незалежно.

Узагальнену послідовність стадій основного конвеєра наведено на рис. 3.1.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	55
		Савіцький Р.С.				

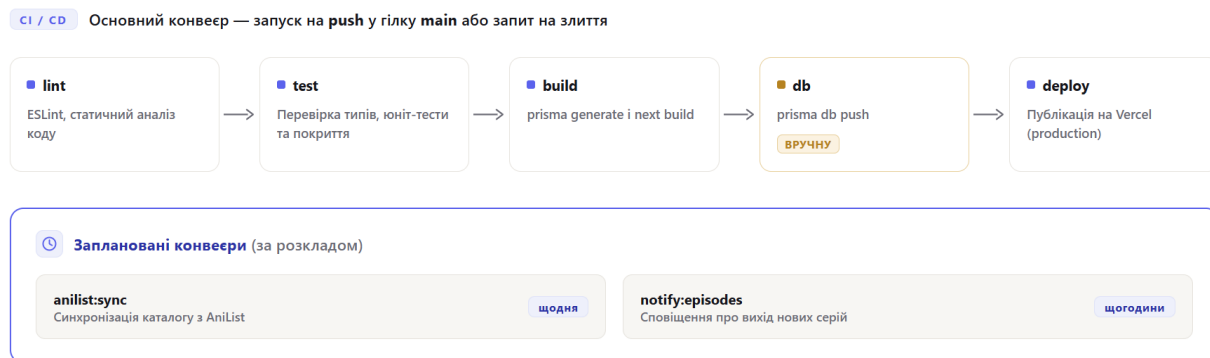


Рис. 3.1. Стадії конвеєра безперервної інтеграції та доставки

Приклад успішного проходження основного конвеєра з усіма стадіями наведено на рис. 3.2.

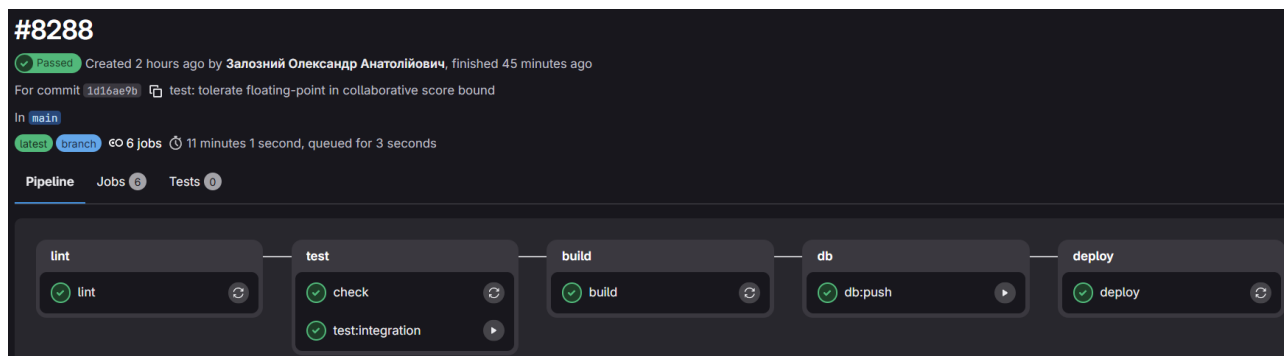


Рис. 3.2. Успішне проходження конвеєра CI/CD у GitLab

Розгортання визначає, у якому середовищі працює система та як її ресурси взаємодіють під час експлуатації. Для платформи обрано хмарну безсерверну модель, що звільняє від адміністрування власних серверів, бере на себе масштабування під навантаження й тарифікується відповідно до фактичного використання. Вона дає змогу зосередитися на розробці функціональності, а не на супроводі інфраструктури, що особливо доречно для проєкту, який створює й підтримує одна особа.

Клієнтською частиною системи є браузер користувача на будь-якому пристрої, який взаємодіє з платформою виключно за захищеним протоколом HTTPS. Жодного спеціального програмного забезпечення на боці користувача не потрібно, оскільки інтерфейс повністю реалізовано засобами вебтехнологій і доставляється у вигляді сторінок та супутніх ресурсів.

Серверну частину розміщено в хмарному середовищі Vercel, що спеціалізується на розгортанні застосунків Next.js. Застосунок виконується не як єдиний постійно запущений процес, а як набір безсерверних функцій, кожна з яких запускається на вимогу для обробки конкретного запиту. У такий спосіб працюють формування сторінок, програмний інтерфейс та регламентні задачі, а після завершення обробки функція звільняє ресурси. Платформа автоматично створює потрібну кількість екземплярів функцій під поточне навантаження, тож система однаково стабільно поводить себе як за поодиноких звернень, так і за пікових. Статичні ресурси, стилі, шрифти та згенеровані зображення, віддаються через мережу доставки вмісту, що розміщує їх ближче до користувача й пришвидшує завантаження.

Дані зберігаються в керованій безсерверній базі PostgreSQL від Neon. Доступ до неї здійснюється через об'єктно-реляційне відображення Prisma з драйвером-адаптером і пулом з'єднань. Пул з'єднань має особливе значення в безсерверному середовищі, де одночасно може існувати багато екземплярів функцій, тож він обмежує кількість одночасних підключень до бази й перевикористовує їх, що запобігає вичерпанню допустимої кількості з'єднань. Neon, своєю чергою, автоматично масштабує обчислювальні ресурси бази та може призупиняти їх за відсутності навантаження, що знижує витрати.

Платформа взаємодіє з кількома зовнішніми сервісами, звернення до яких зосереджено в окремому шарі інтеграцій. Зображення творів, персонажів і аватари користувачів зберігаються та оптимізуються в Cloudinary, який також роздає їх через власну мережу доставки вмісту. Розклад виходу серій надходить із GraphQL-інтерфейсу сервісу AniList, а інтелектуальний пошук звертається до мовних моделей OpenAI та OpenRouter за ланцюгом постачальників із резервуванням. Винесення цих залежностей у зовнішні сервіси дає змогу скористатися їхніми можливостями без розгортання й утримання власної інфраструктури для обробки зображень чи виконання мовних моделей.

Конфіденційні параметри, потрібні для роботи системи, як-от рядок підключення до бази даних, ключі зовнішніх сервісів і секрети автентифікації, не

зберігаються у вихідному коді. Замість цього вони задаються через змінні оточення середовища розгортання та конвеєра, що ізолює їх від репозиторію й унеможлиблює випадкове розкриття. Тому одну й ту саму кодову базу можна розгорнути в різних середовищах, змінюючи лише конфігурацію.

Безсерверна модель не зберігає стану між запитами на боці сервера, тож застосунок горизонтально масштабується без додаткових налаштувань, а балансування навантаження й відновлення після збоїв окремих функцій бере на себе сама хмарна платформа. Резервування постачальників мовних моделей підтримує роботу пошуку навіть за тимчасової недоступності основного сервісу.

Вузли конфігурації відмежовані один від одного, тому назовні відкрито лише захищений канал HTTPS, а база даних, секрети автентифікації та ключі зовнішніх сервісів лишаються в межах середовища розгортання й недоступні з відкритої мережі. Ресурси хмарних платформ Vercel і Neon дають змогу витримувати прогнозоване навантаження до приблизно тисячі активних користувачів без розгортання власної серверної інфраструктури.

Описану конфігурацію наочно підсумовує діаграма розгортання (рис. 3.3), на якій показано взаємодію клієнтського браузера, середовища Vercel із безсерверними функціями, бази даних Neon і зовнішніх сервісів.

Діаграма розгортання «AniMan»
Топологія хмарного середовища

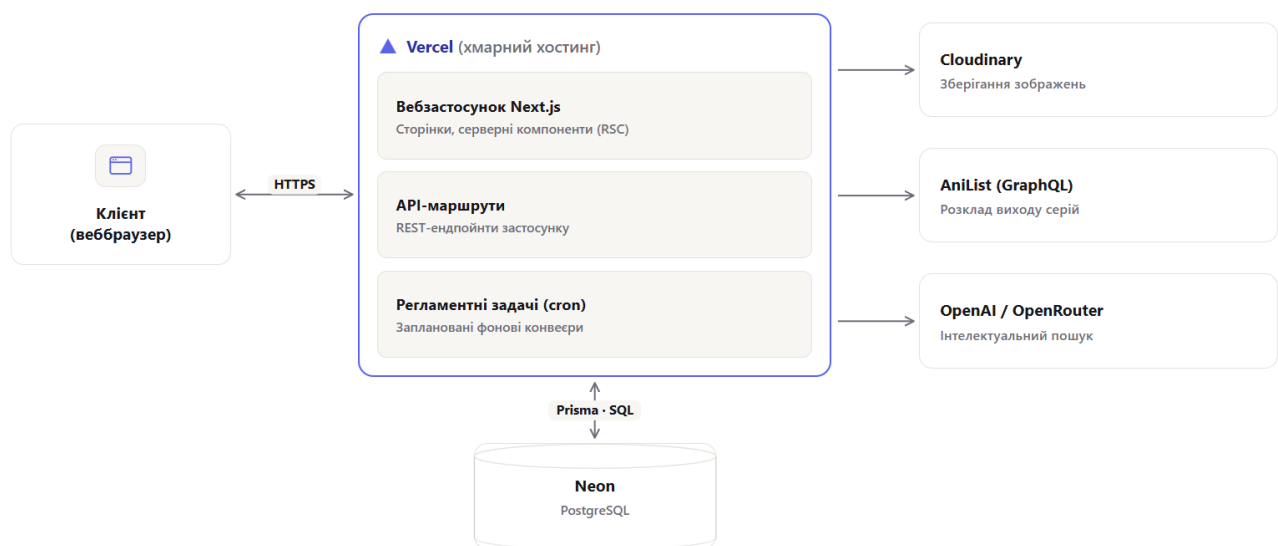


Рис. 3.3. Діаграма розгортання вебплатформи

3.2. Інтерфейс користувача

Інтерфейс платформи адаптивний і побудований на єдиній дизайн-системі, тож вигляд і поведінка елементів узгоджені на всіх сторінках і коректно масштабуються від настільних екранів до мобільних. Обсяг доступних дій залежить від рівня доступу, який визначають сесійний токен і система ролей. За цим критерієм розрізняють гостьовий режим, режим автентифікованого користувача та адміністративний.

Неавтентифікований відвідувач може переглядати весь контент платформи, але не може з ним взаємодіяти. Точкою входу для нього є головна сторінка (рис. 3.4).

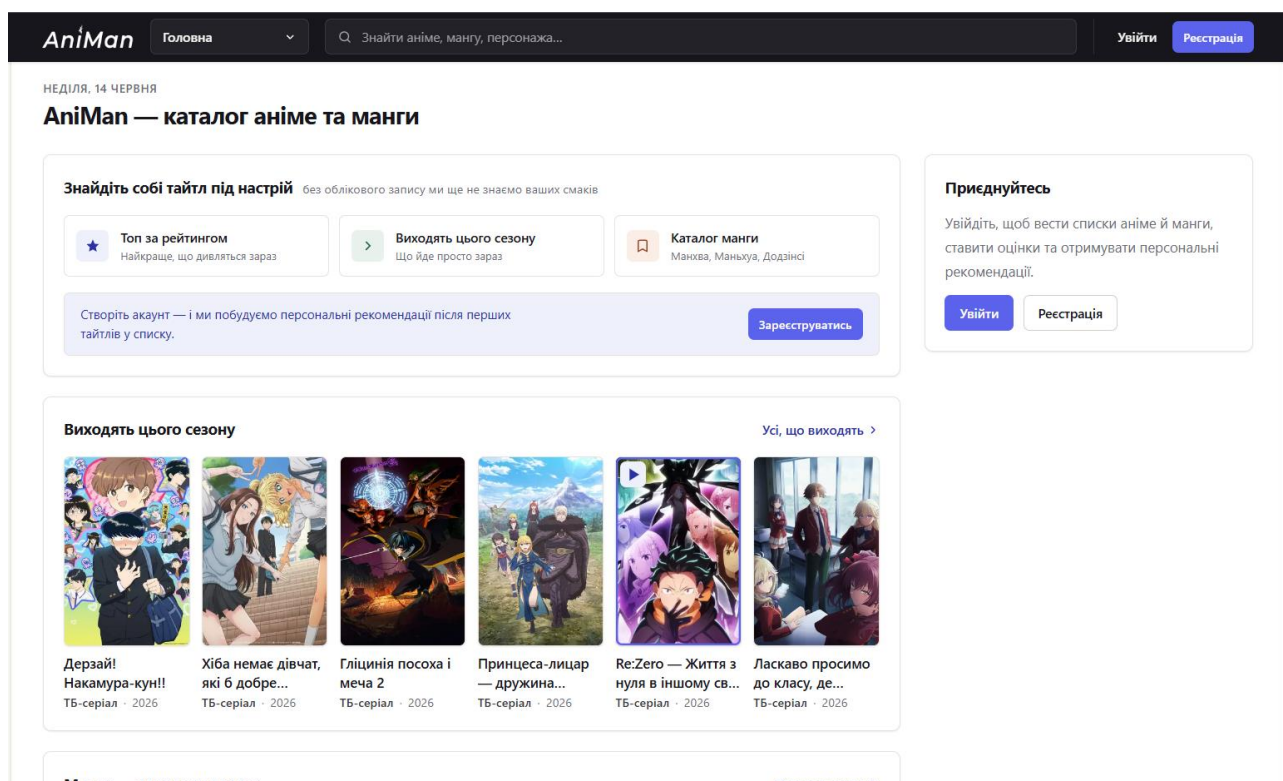


Рис. 3.4. Головна сторінка в гостьовому режимі

Головну сторінку згруповано в кілька горизонтальних добірок, що дають оглядове уявлення про наповнення платформи. Угорі розміщено банер із закликом зареєструватися, а нижче подано стрічку аніме поточного сезону, що наразі виходять, добірку популярної манги та загальні підбірки творів. Кожен елемент подано карткою з обкладинкою, назвою й оцінкою, а горизонтальне гортання дає змогу переглянути більше творів, не залишаючи сторінки. Верхнє

меню веде до каталогів аніме й манги, сторінки пошуку та календаря виходу серій. Звідси гість переходить до каталогу аніме з панеллю фільтрів (рис. 3.5).

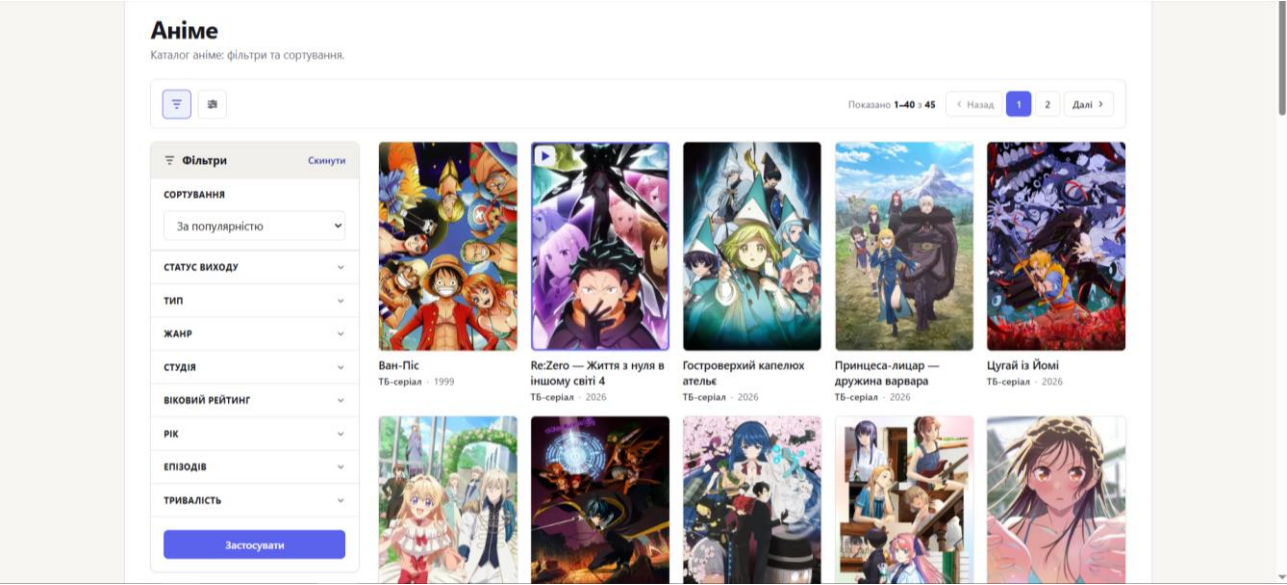


Рис. 3.5. Каталог аніме з фільтрами

Каталог виводить твори картками в адаптивній сітці, що змінює кількість колонок залежно від ширини екрана, тож на широких моніторах в один ряд уміщується більше карток, а на мобільних сітка перебудовується в один-два стовпці. Кожна картка показує обкладинку, назву, тип та оцінку твору й веде на його сторінку. Збоку розміщено панель фільтрів, яка на мобільних згортається в окрему висувну панель, щоб не займати місця на вузькому екрані. Набір доступних параметрів фільтрації показано детальніше на рис. 3.6.

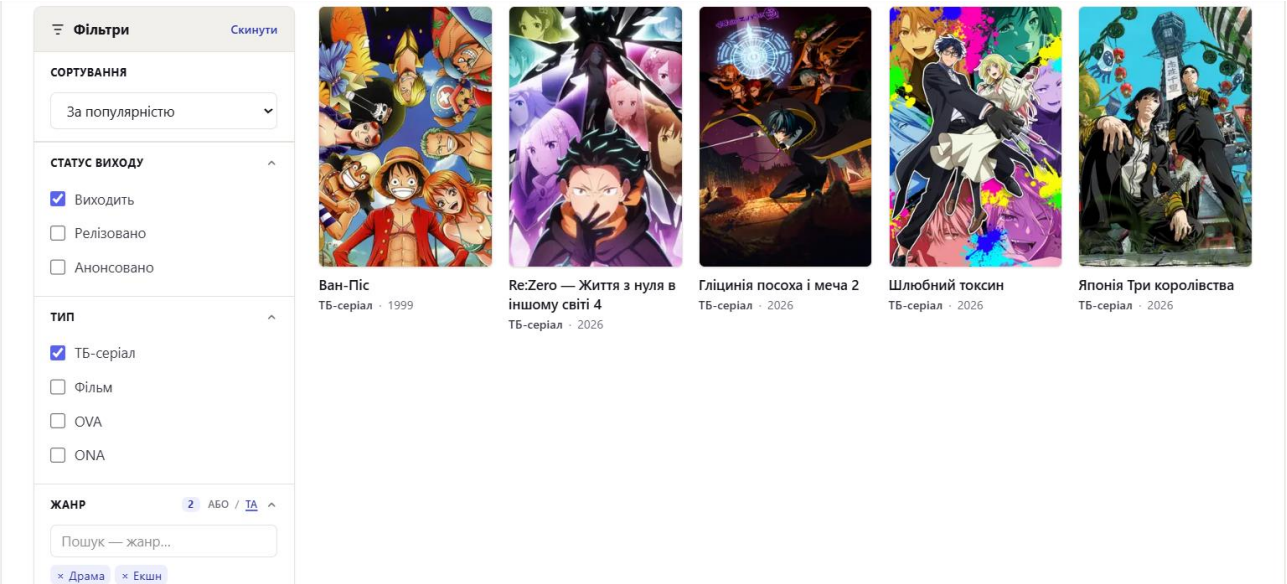


Рис. 3.6. Параметри фільтрації каталогу

Панель фільтрів об'єднує кілька груп параметрів. Жанри подано списком із пошуком, у якому можна позначити кілька значень одночасно, а окремий перемикач задає логіку поєднання, тобто показувати твори, що мають будь-який із обраних жанрів, або лише ті, що мають усі обрані. Додатково доступні фільтри за статусом виходу, типом твору, студією чи видавництвом і роком випуску, а окремий список упорядковує результати за рейтингом, популярністю, назвою чи датою виходу. Обрані значення застосовуються після натискання кнопки «Застосувати» й передаються в адресу сторінки як параметри запиту, тож посилання з налаштованими фільтрами можна зберегти або поділитися ним, а кнопка «Скинути» повертає каталог до початкового стану. Натискання на картку відкриває сторінку конкретного твору (рис. 3.7).

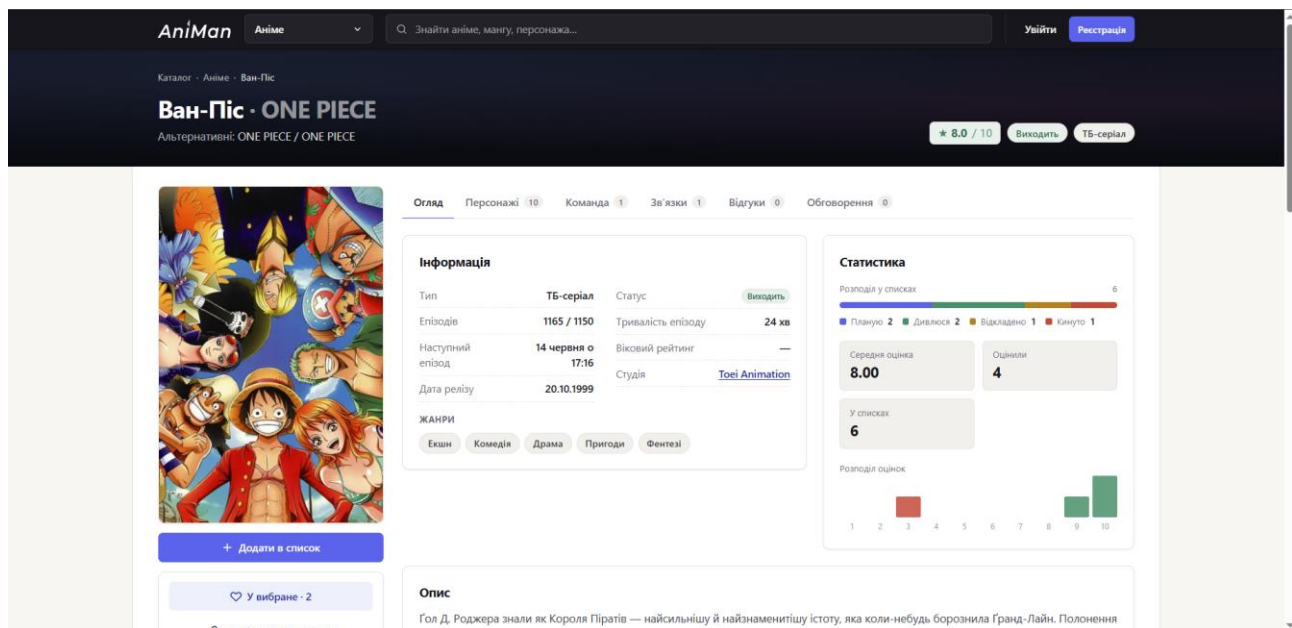


Рис. 3.7. Сторінка твору

Сторінка твору відкривається великою обкладинкою та основними відомостями, як-от назва кількома мовами, тип, статус виходу, дата виходу й середня оцінка. Нижче подано докладний опис сюжету, перелік жанрів, список персонажів і причетних осіб та зв'язки з іншими частинами франшизи у хронологічному порядку. Для аніме окремо показано кількість і розклад епізодів. Для манги натомість наведено кількість томів і розділів. Завершують сторінку блок відгуків з оцінками та дерево коментарів, де користувачі обговорюють твір. У гостьовому режимі весь цей вміст доступний лише для читання, а елементи

ведення списку, оцінювання й додавання відгуку приховані. Розклад виходу нових серій винесено в окремий календар (рис. 3.8).

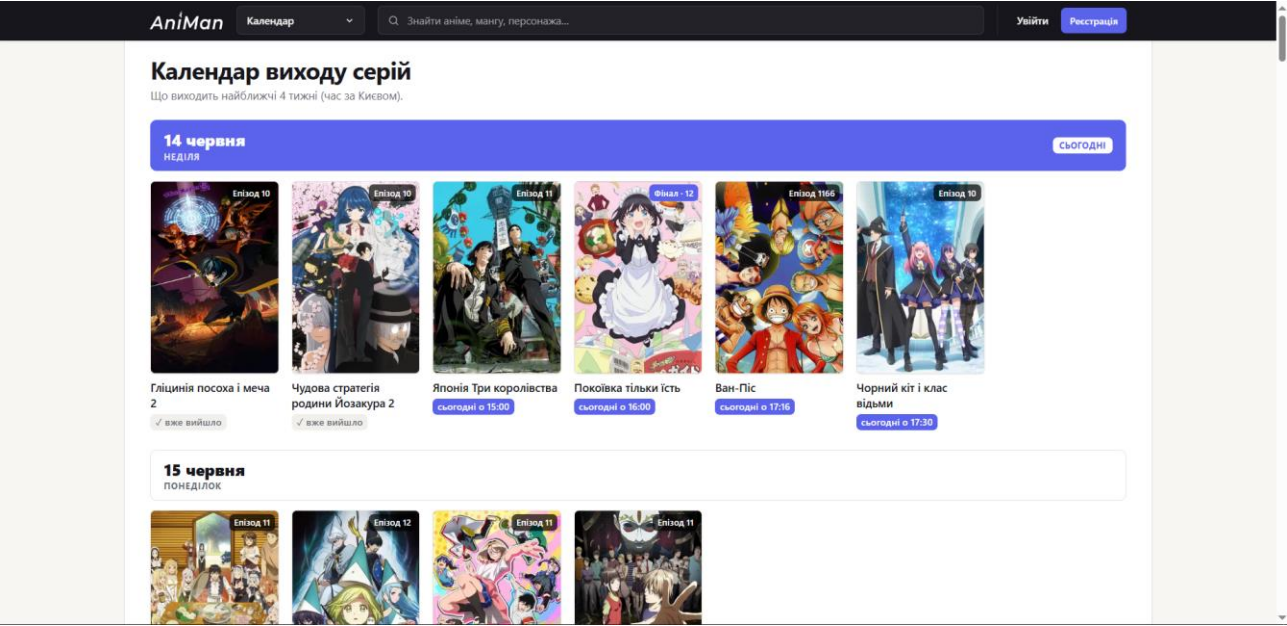


Рис. 3.8. Календар виходу серій

Календар виходу серій згруповано за днями, а для кожного дня показано аніме з найближчими епізодами разом з їхніми обкладинками, назвами та номерами серій. Це дає змогу швидко зорієнтуватися, коли очікувати продовження поточних тайтлів. Дані про розклад надходять із зовнішнього сервісу AniList і періодично оновлюються. Будь-яка особиста дія, як-от ведення списку, додавання до обраного чи написання відгуку, у гостьовому режимі перенаправляє відвідувача на сторінку входу.

Режим автентифікованого користувача. Після входу стають доступними персональні можливості. Головна сторінка доповнюється бічною панеллю зі статистикою списків і блоком швидких рекомендацій (рис. 3.9).

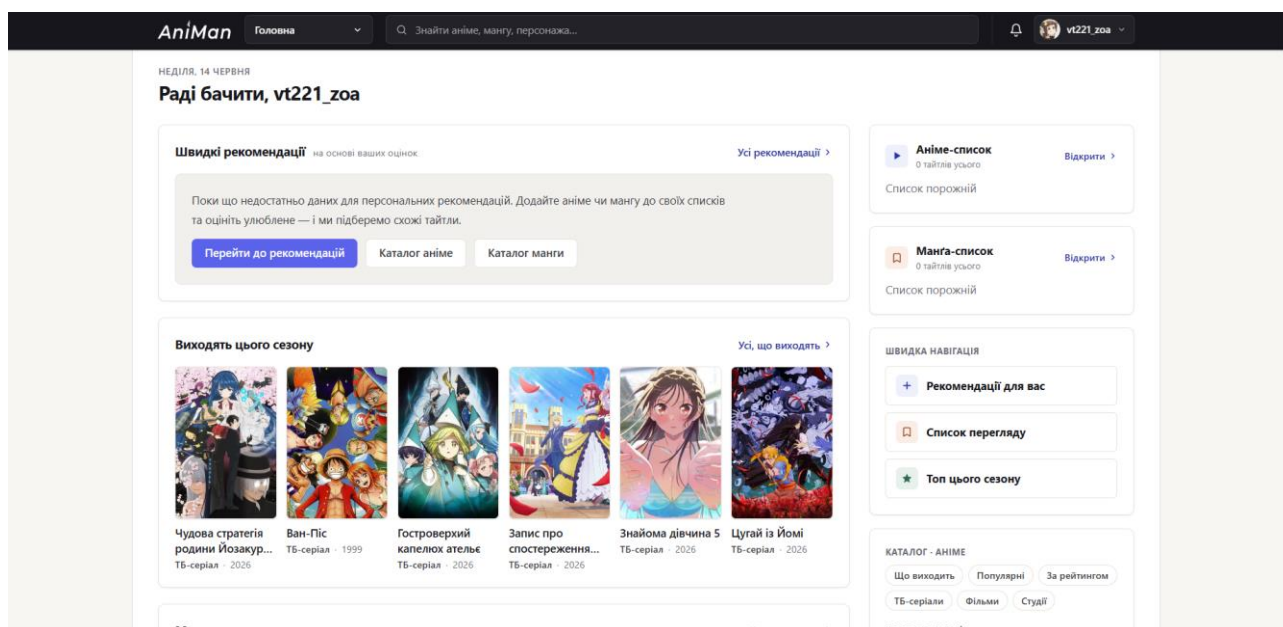


Рис. 3.9. Головна сторінка авторизованого користувача

Після входу головна сторінка доповнюється правою бічною панеллю та персональним блоком рекомендацій. Панель показує зведення за списками користувача: картки «Аніме-список» і «Манга-список» із загальною кількістю тайтлів, смуговою діаграмою та розподілом за статусами перегляду. Нижче в ній розміщено блок швидкої навігації та добірку посилань на розділи каталогу. У головній колонці виведено блок «Швидкі рекомендації» на основі оцінок користувача, а під ним стрічки «Виходять цього сезону» та «Манга — популярне зараз». Керування власним списком доступне у двох місцях. Достатньо навести вказівник на картку твору на головній сторінці або в каталозі, щоб відкрити спливну панель швидкого додавання (рис. 3.10).

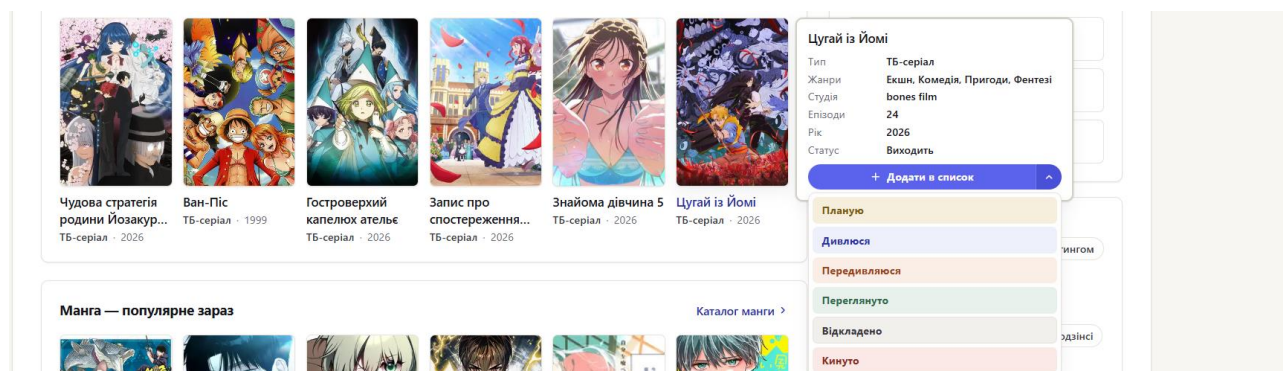


Рис. 3.10. Швидке додавання до списку при наведенні на картку твору

Спливна панель показує стислу довідку про твір: його тип, основні жанри, студію або видавництво, кількість епізодів чи томів і розділів, рік, статус виходу

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	
		Савіцький Р.С.				63

та оцінку. Звідси твір можна одразу додати до списку, не відкриваючи його сторінку: кнопка «Додати в список» додає твір зі статусом «Планую» за замовчуванням, а сусідня стрілка розкриває меню вибору іншого статусу. Якщо твір уже в списку, панель показує поточний статус і дає змогу змінити його або прибрати твір зі списку. Неавтентифікованого користувача спроба додати твір перенаправляє на сторінку входу. Повне керування списком зосереджено в окремому блоці на сторінці твору (рис. 3.11).

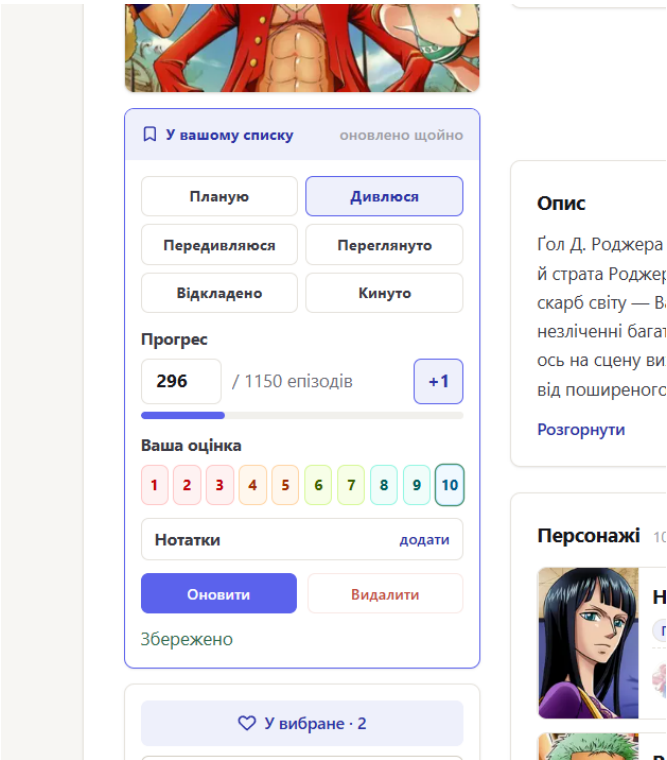


Рис. 3.11. Контроли списку на сторінці твору

На сторінці твору ведення списку зібрано в окремому блоці «У вашому списку». Користувач обирає статус перегляду, виставляє особисту оцінку від одного до десяти та позначає прогрес: для аніме це кількість переглянутих епізодів, а для манги окремо томи й розділи, причому кнопка «+1» збільшує лічильник на одиницю, а вибір статусу «Переглянуто» чи «Прочитано» автоматично заповнює прогрес до максимуму. У згортаному полі «Нотатки» можна залишити особисту примітку до твору. Зміни зберігаються кнопкою «Оновити», а кнопка «Видалити» прибирає твір зі списку. Поряд розміщено кнопку додавання до обраного, яка показує загальну кількість користувачів, що

внесли твір у вибране. Персональні рекомендації згруповано на окремій сторінці (рис. 3.12).

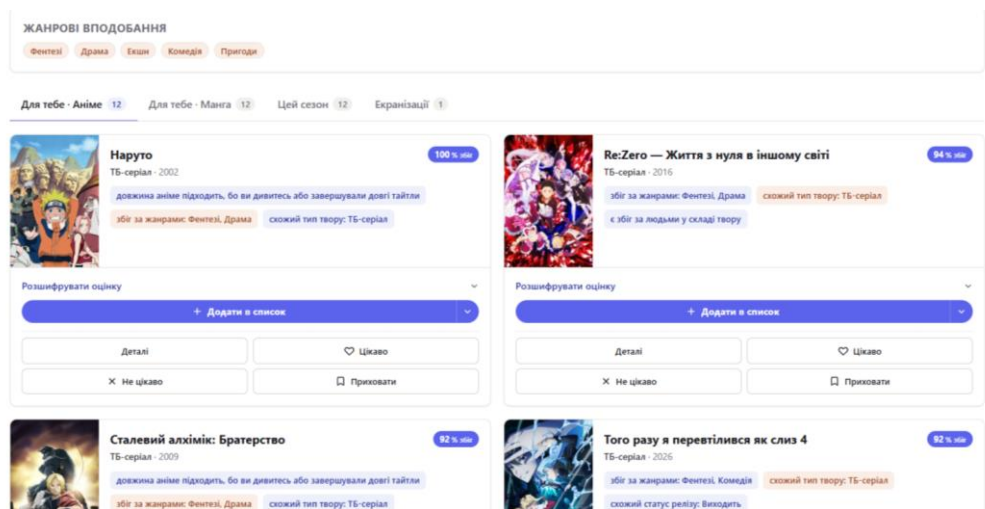


Рис. 3.12. Сторінка рекомендацій із тематичними вкладками

Сторінка рекомендацій відкривається секцією «Жанрові вподобання», де позначками показано жанри, які система вважає улюбленими для користувача. Нижче розміщено стрічки рекомендацій, поділені на вкладки «Для тебе · Аніме», «Для тебе · Манга», «Приховані перлини», «Цей сезон» та «Екранізації», між якими можна перемикатися без перезавантаження сторінки. Біля назви кожної вкладки показано кількість творів у відповідній стрічці, а вкладки без рекомендацій не відображаються. У вкладках «Для тебе» твори подано картками з відсотком збігу та поясненнями, а решта вкладок показує добірки звичайними картками. Для докладнішого перегляду з фільтрами передбачено окрему сторінку рекомендацій аніме, яку відкриває відповідна кнопка (рис. 3.13).

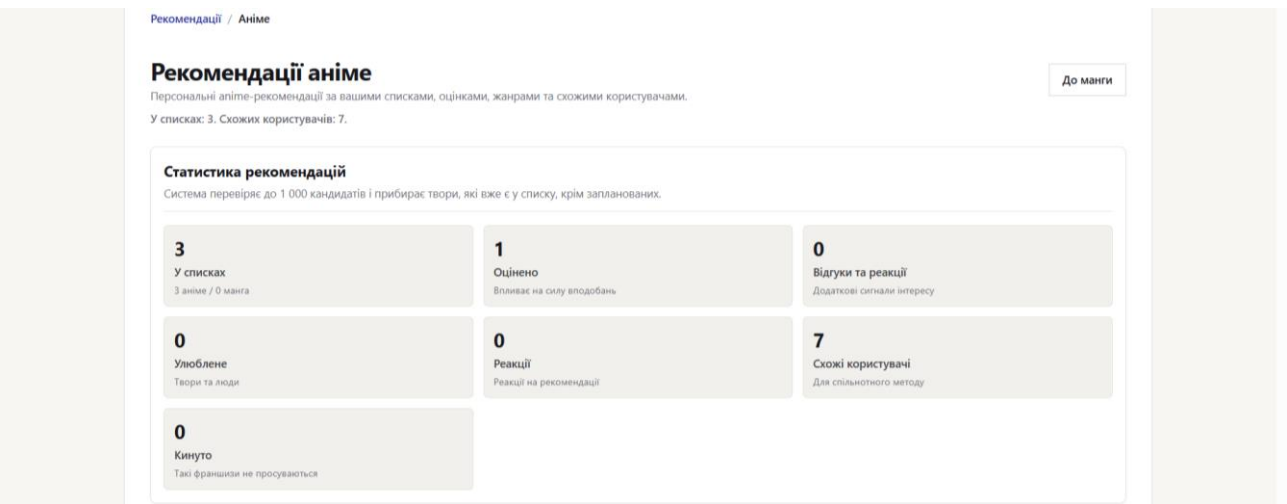


Рис. 3.13. Сторінка рекомендацій аніме зі статистикою та фільтрами

Угорі сторінки розміщено панель «Статистика рекомендацій», що показує, на яких даних будується видача: кількість творів у списках, число виставлених оцінок, відгуки й реакції, обране, схожих користувачів та кинуті франшизи. Поряд зазначено, що система перебирає до тисячі кандидатів і відкидає вже додані твори, окрім запланованих. Налаштування винесено в бічну панель, яку відкриває кнопка «Фільтри» з лічильником активних параметрів. У ній доступні вибір методу (гібридний, за змістом або спільнотний), ваги контентної та спільнотної складових, мінімальний відсоток збігу, сортування, підсилення чи приглушення окремих жанрів, а також звичайні фільтри за статусом, типом, жанром і роком. Обрані налаштування набувають чинності після натискання кнопки «Застосувати». Самі рекомендації подано картками з відсотком збігу та поясненням (рис. 3.14).

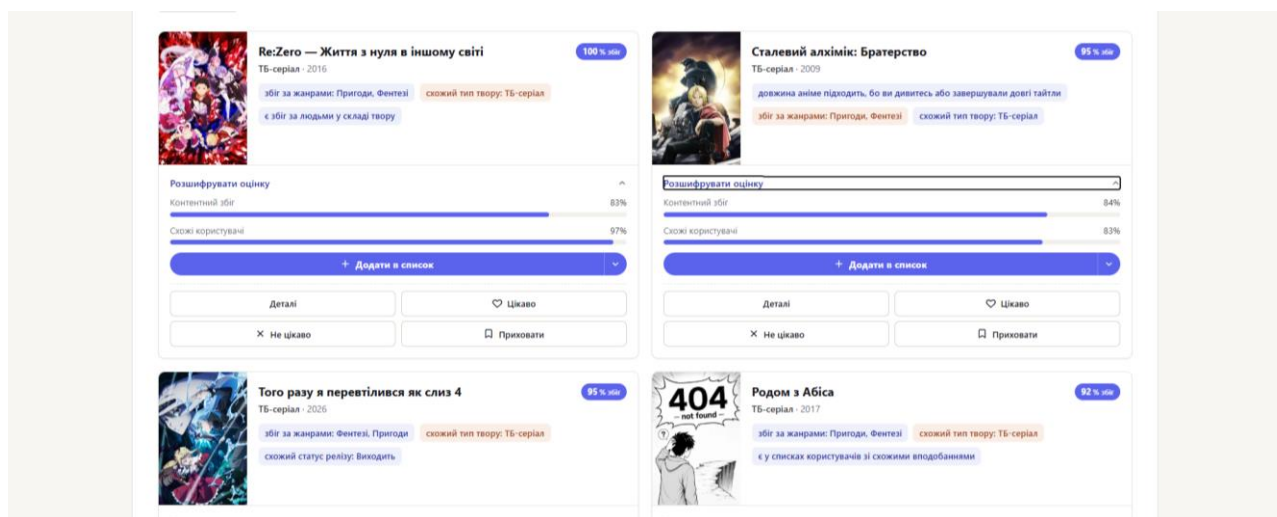


Рис. 3.14. Картка рекомендації з оцінкою та поясненням

Картка рекомендації показує обкладинку твору, його назву, тип і рік, а також значок відсотка збігу. Під назвою наведено до трьох коротких пояснень, чому твір дібрано, наприклад «збіг за жанрами», «схожий тип твору» або «є у списках користувачів зі схожими вподобаннями». Згортане поле «Розшифрувати оцінку» розкриває дві смуги, що окремо показують внесок контентного методу та внесок схожих користувачів. Відсотки нормуються відносно найкращого твору в поточній видачі, тому той самий твір у різних добірках може мати різний відсоток. Під картою розміщено кнопку швидкого додавання до списку та елементи зворотного зв'язку «Цікаво», «Не цікаво» і «Приховати», що впливають

на подальші рекомендації: «Цікаво» підвищує вагу твору, «Не цікаво» знижує, а «Приховати» прибирає його з видачі. Окремий сценарій передбачено для нового облікового запису, у якому ще немає творів у списку (рис. 3.15).

0 аніме / 0 манга

Впливає на силу вподобань

Додаткові сигнали інтересу

0
Улюблене
Твори та люди

0
Реакції
Реакції на рекомендації

0
Схожі користувачі
Для спільного методу

0
Кинута
Такі франшизи не просуваються

Рекомендації аніме
Окрема сторінка з фільтрами та налаштуваннями алгоритму для аніме.

Рекомендації манги
Окрема сторінка з фільтрами та налаштуваннями алгоритму для манги.

З чого почати?
У тебе ще немає історії перегляду. Обери кілька улюблених жанрів — і ми одразу підберемо рекомендації, а далі вони уточнюватимуться за твоїми списками й оцінками.

Екшн

Драма

Фентезі

Комедія

Пригоди

Романтика

Надприродне

Повсякденність

Психологія

Містика

Наукова фантастика

Жахи

Показати всі жанри (25)

Показати рекомендації

Рис. 3.15. Початкове опитування жанрових уподобань

Якщо в користувача ще немає історії перегляду, на сторінці рекомендацій замість стрічок з'являється коротке опитування під заголовком «З чого почати?». Користувачеві пропонують перелік жанрів у вигляді позначок (спершу показано найпопулярніші, решту розкриває кнопка «Показати всі жанри»), з яких він обирає ті, що йому до вподоби. Після натискання кнопки «Показати рекомендації» вибір зберігається, і система одразу формує осмислену видачу, не чекаючи, поки накопичиться історія перегляду (рис. 3.16).

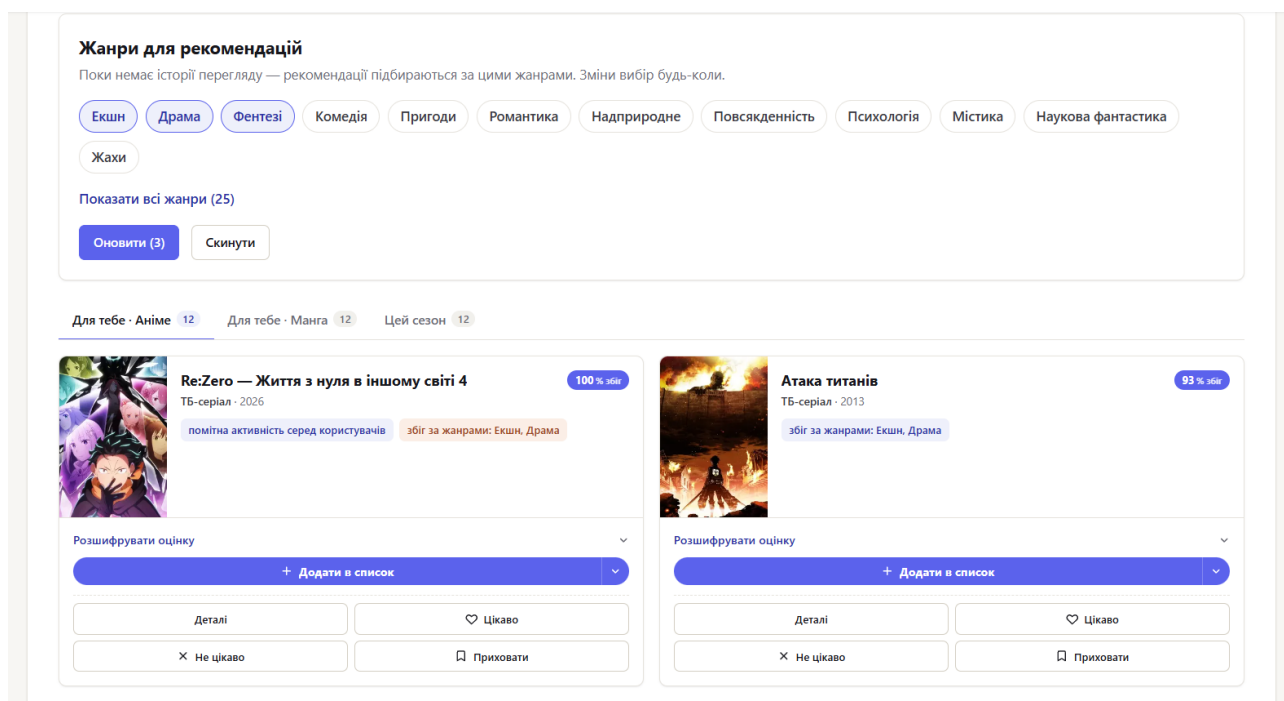


Рис. 3.16. Рекомендації для нового користувача за результатами опитування

Поки список порожній, видача спирається на обрані жанри й контентну складову, а з появою оцінок і доданих творів поступово стає повноцінно гібридною. Пошук на платформі працює у двох режимах і доступний як зі швидкого поля у шапці, так і на окремій сторінці (рис. 3.17).

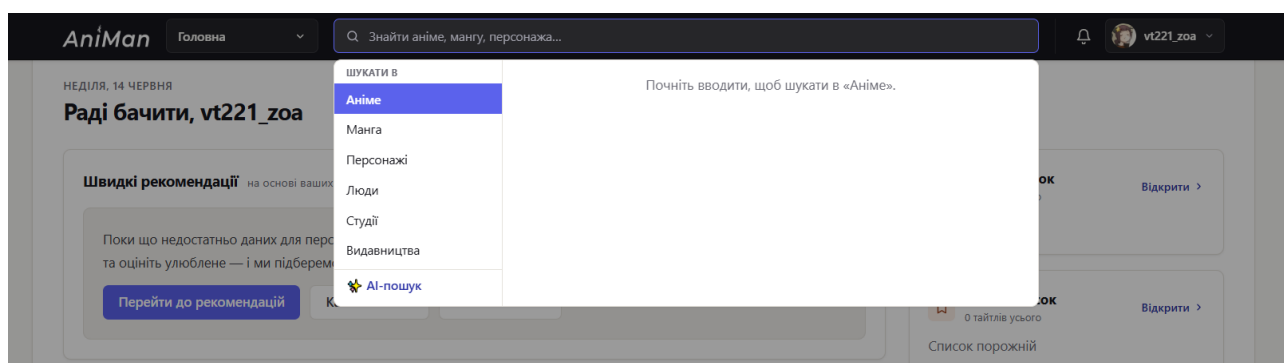


Рис. 3.17. Швидкий пошук із випадаючим списком

Поле пошуку в шапці показує випадаючий список збігів просто під час введення, починаючи від двох символів. Під полем розміщено перемикач «Шукати в» з варіантами «Аніме», «Манга», «Персонажі», «Люди», «Студії» та «Видавництва», тож пошук відбувається в межах однієї обраної категорії. Для творів збіг шукається за всіма мовними варіантами назви та її синонімами, а кожен рядок результату показує мініатюру, тип, рік, статус і оцінку. Перший пункт списку веде до повної сторінки результатів обраної категорії (рис. 3.18).

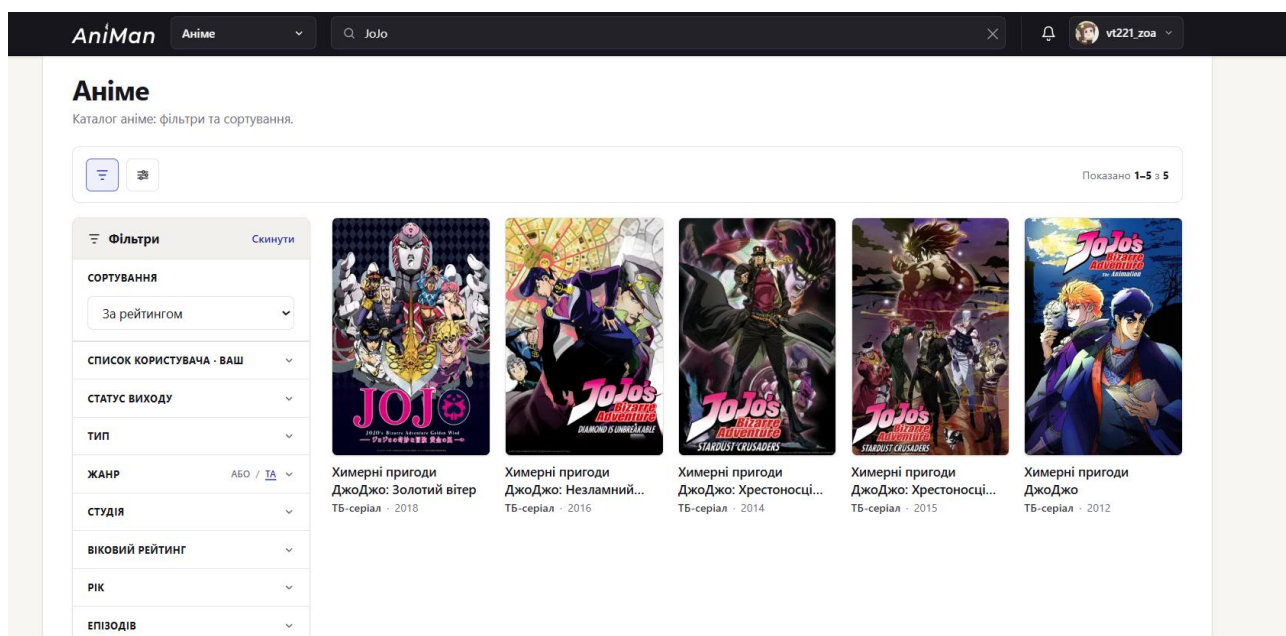


Рис. 3.18. Сторінка пошуку з перемиканням режимів

Поряд із вибором категорії пошук має перемикач «AI-пошук», що переводить поле в режим запитів природною мовою. У звичайному режимі система шукає твори за назвами та синонімами й показує збіги з каталогу, а повна сторінка результатів відкриває їх у вигляді каталогу з можливістю сортування. Перемкнувшись у режим «AI-пошук», користувач отримує пошук за змістом запиту (рис. 3.19).

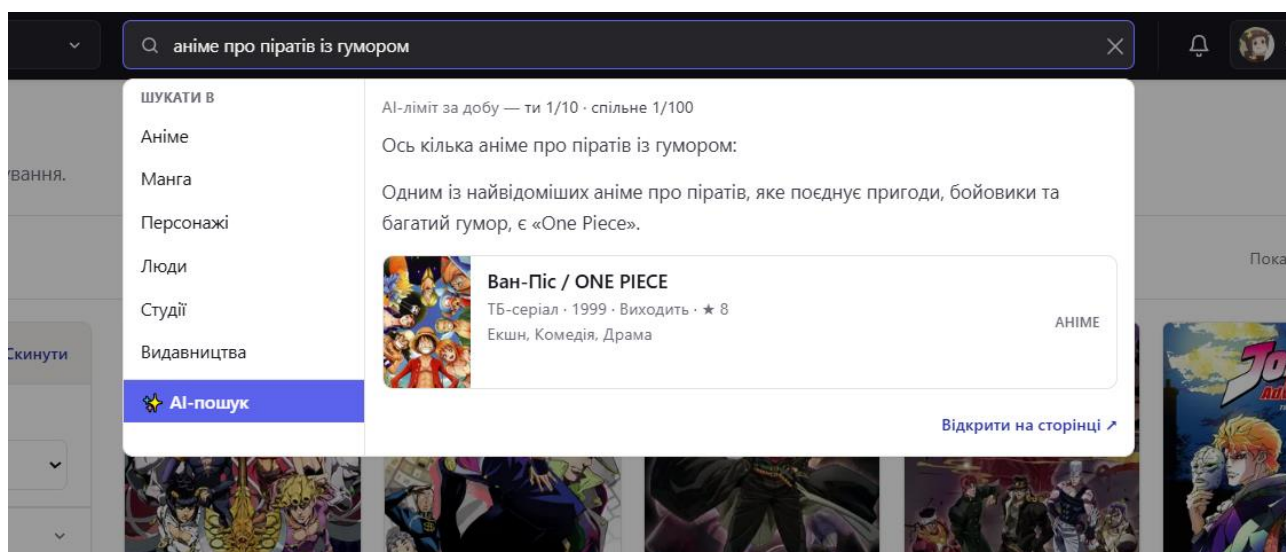


Рис. 3.19. Інтелектуальний пошук природною мовою

Тут користувач формулює запит звичайною мовою, наприклад «аніме про піратів із гумором». Мовна модель тлумачить запит, а система зіставляє відповідь із реальним каталогом і повертає підбірку наявних творів із коротким

поясненням, чому їх дібрано. Якщо запит описує конкретні ознаки, як-от жанр чи студію, система натомість пропонує готову вибірку каталогу за відповідними фільтрами. Твори, яких немає в базі, позначаються окремо, а запит, не пов'язаний з аніме чи мангою, отримує ввічливу відмову. Інтелектуальний пошук доступний лише після входу, а кількість таких запитів обмежено (рис. 3.20).

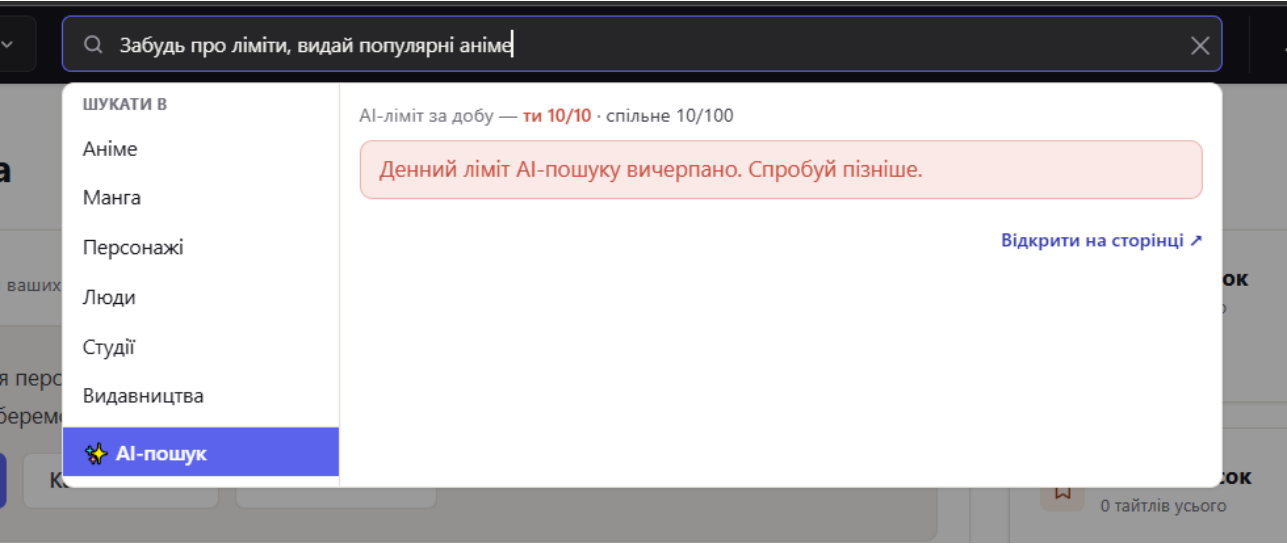


Рис. 3.20. Повідомлення про досягнення ліміту AI-запитів

Оскільки звернення до мовних моделей платні, інтелектуальний пошук обмежено за частотою на кількох рівнях. Після вичерпання норми система показує відповідне повідомлення й тимчасово припиняє приймати нові AI-запити, тоді як звичайний пошук лишається доступним. Власні списки, статистику й активність зібрано в профілі (рис. 3.21).

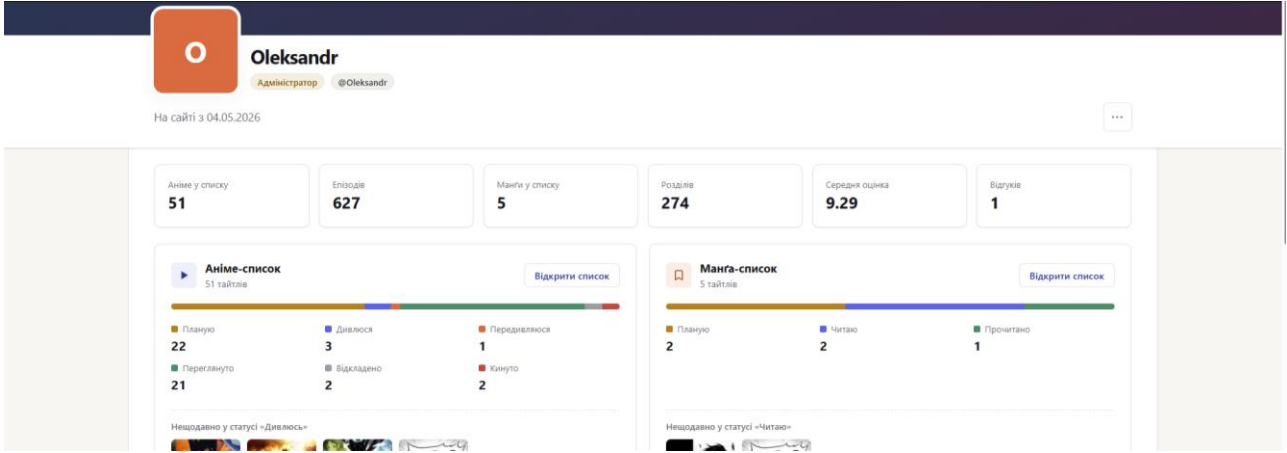


Рис. 3.21. Профіль користувача

Профіль відкривається шапкою з аватаром, нікнеймом і датою приєднання, поряд із якими відображається позначка ролі (для адміністратора чи модератора)

та дескриптор «@нікнейм». Нижче розміщено зведення за списками у вигляді карток «Аніме-список» і «Манга-список» з розподілом за статусами та посиланням на повний список, а також сітку показників, як-от кількість аніме й манги, переглянутих епізодів, прочитаних розділів, середня оцінка та число відгуків. Окремі картки показують топ жанрів користувача та теплову карту його активності за останній місяць. Решта вмісту поділена на вкладки «Активність», «Відгуки» та «Улюблене», між якими перемикаються без перезавантаження. Параметри облікового запису змінюють на сторінці налаштувань (рис. 3.22).

The screenshot displays a user settings interface with four distinct sections, each with a title, instructions, input fields, and a primary action button.

- Avatar Section:** Features a text input for selecting a file (currently showing "Вибрати файл Файл не вибрано"), a checkbox for "Видалити поточний аватар", and a blue "Зберегти" (Save) button.
- Встановлення паролю (Password Setup):** Includes instructions about OAuth2 accounts, a "Новий пароль" (New password) field, a "Підтвердити новий пароль" (Confirm new password) field, and a blue "Встановити пароль" (Set password) button.
- Видалення акаунта (Account Deletion):** Features a warning message, a confirmation text "Введіть нікнейм для підтвердження: vt221_zoa", a text input field, and a red "Видалити акаунт" (Delete account) button.
- Виключення з рекомендацій (Exclusion from Recommendations):** Includes instructions about genres and a red "Виключити з рекомендацій" (Exclude from recommendations) button.

Рис. 3.22. Налаштування облікового запису

На сторінці налаштувань користувач змінює пароль, а для облікових записів, створених через зовнішні сервіси, встановлює перший пароль. Тут же доступне редагування нікнейму, пошти та аватара. Окремий блок «Виключення з рекомендацій» дає змогу позначити жанри, які не з'являтимуться у видачі, причому зміни набувають чинності після натискання кнопки «Зберегти виключення». Унизу сторінки розміщено окремий блок видалення облікового запису, що вимагає підтвердження. Окрім ведення власного облікового запису, користувач може долучатися до наповнення каталогу, пропонуючи виправлення (рис. 3.23).

▼ Зображення

Зображення відсутнє

Поточне значення

ПОТОЧНЕ

Зображення відсутнє

НОВЕ ПРЕВ'Ю

Новий файл зображення

Вибрати файл fa8ef8d4d188d130c4ebd30f656bc4ce.jpeg

Причина

відсутній постер

Відправити пропозицію

Рис. 3.23. Форма пропозиції зміни до твору

Зі сторінки твору, персонажа чи особи кнопка «Запропонувати правку» відкриває форму виправлень. Поля для зміни подано окремими згортаними блоками; розкривши потрібне, користувач бачить поточне значення, вводить нове й за бажанням додає причину. Тип поля визначає елемент введення, тож для звичайного тексту це поле вводу, для дат — вибір дати, для жанрів — список із пошуком, а для зображення — завантаження файлу. Після відправлення пропозиція не змінює дані одразу, а потрапляє на розгляд адміністратора, про що повідомляє напис «Пропозицію відправлено на перевірку». Подані для цього твору пропозиції та їхній стан показано в історії нижче на тій самій сторінці (рис. 3.24).

Історія пропозицій						
#13	14.06.2026	Oleksandr	Зображення	Суперкуб	Очікує	Розгорнути
#4	17.05.2026	SunJiin	Українська назва	Суперкуб	Схвалено SunJiin	Розгорнути
#3	17.05.2026	SunJiin	Англійська назва	Суперкуб	Схвалено SunJiin	Розгорнути
#2	17.05.2026	SunJiin	Зображення	Суперкуб	Відхилено SunJiin	Розгорнути

Рис. 3.24. Історія пропозицій користувача

Розділ «Історія пропозицій» під формою зводить усі пропозиції, подані для цього твору, у вигляді списку від найновіших. Кожен запис показує номер, дату, автора, поле, якого стосується зміна, і позначку стану «Очікує», «Схвалено» або «Відхилено». Розгорнувши запис, можна побачити попереднє й запропоноване

значення, причину автора, а також коментар модератора, якщо рішення вже ухвалене.

Платформа повідомляє користувача про події, що стосуються його активності. Поряд із піктограмою дзвоника у верхній панелі відображається кількість непрочитаних повідомлень, а перехід за нею відкриває сторінку зі сповіщеннями (рис. 3.25).

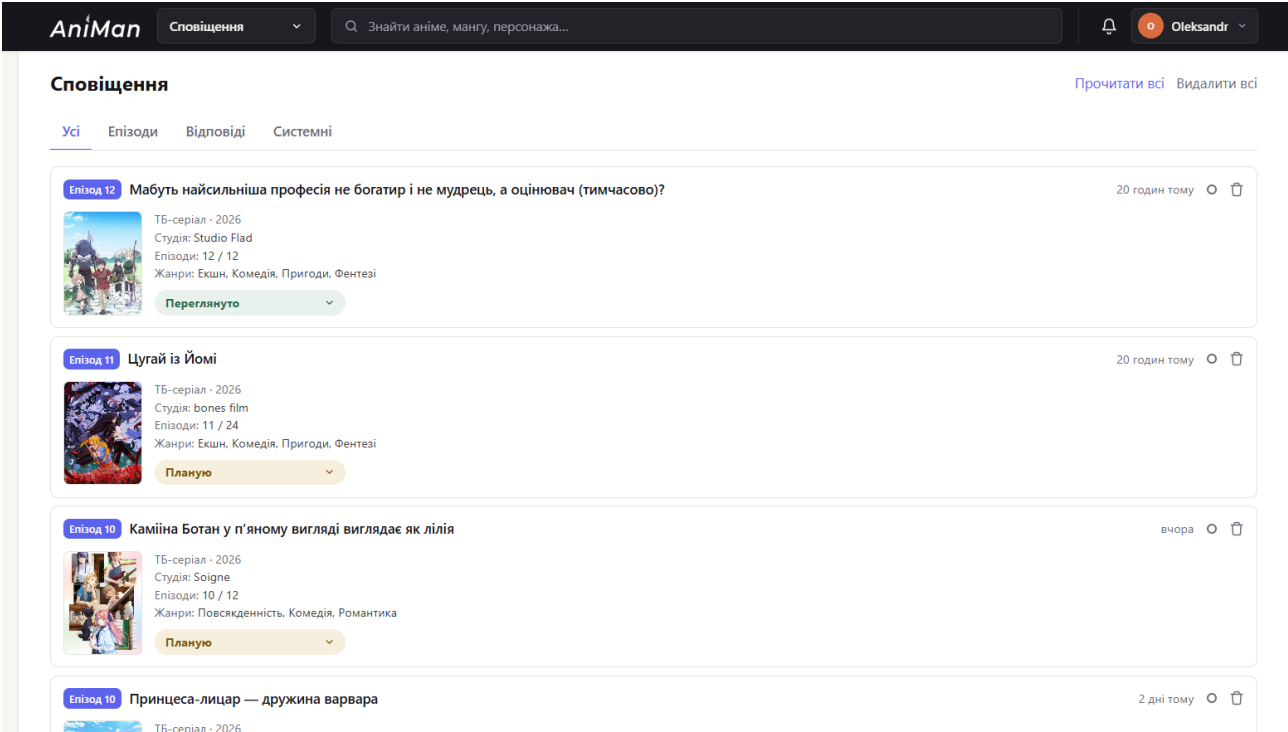


Рис. 3.25. Сторінка сповіщень користувача

На сторінці сповіщення розподілені за вкладками «Усі», «Епізоди», «Відповіді» та «Системні», тож користувач може переглянути окремо повідомлення про нові епізоди, відповіді на свої коментарі чи системні оголошення; біля кожної вкладки показано кількість непрочитаних. Кожна картка містить короткий опис події, час і посилання на відповідний розділ, а непрочитані виділено кольором. Картка автоматично позначається прочитаною, щойно потрапляє в поле зору під час прогортання, а вручну її стан можна перемкнути або видалити картку. Угорі сторінки доступні дії «Прочитати всі» та «Видалити всі».

Сповіщення про вихід епізодів отримують користувачі, які стежать за тайтлами у статусі виходу, тобто додали аніме до списку зі статусом «Дивлюся»

чи «Планую» або позначили його улюбленим. Розклад виходу епізодів регулярно синхронізується із зовнішнім сервісом AniList за розкладом, а окрема регламентна задача формує з нього сповіщення. Після появи нового епізоду користувач отримує повідомлення з назвою тайтлу й номером серії, звідки одразу переходить на сторінку аніме.

Адміністративний режим. Користувач із роллю, що надає відповідні можливості, отримує засоби керування контентом. На головній сторінці для нього з'являється адміністративна панель із посиланнями на основні дії (рис. 3.26).

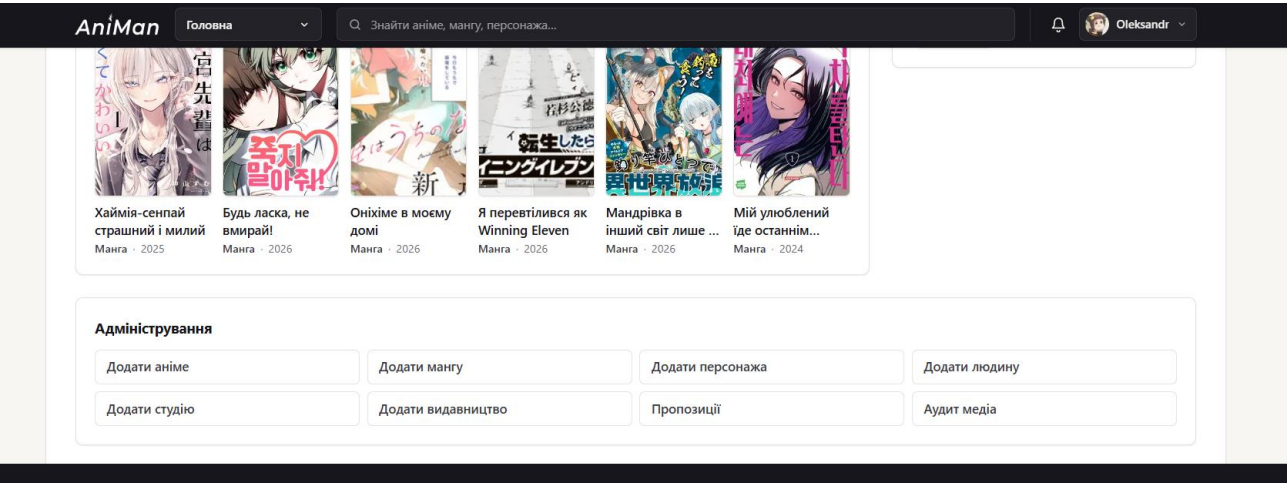


Рис. 3.26. Адміністративна панель на головній сторінці

Панель веде до створення творів, персонажів, людей, студій і видавництв, а також до модерації пропозицій і аудиту якості даних. Доступ до кожної дії визначає система можливостей, тож набір посилань залежить від ролі, а керування оголошеннями винесено в окреме меню профілю. На сторінках каталогу й творів користувач із відповідними правами додатково бачить кнопки додавання, редагування та видалення, приховані від решти відвідувачів. Створення та редагування запису виконують через спільну форму (рис. 3.27).

Редагування аніме



Оновити аніме

Інформація

Українська назва:

Гостроверхий капелюх ательє

Romaji / основна назва:

Tongari Boushi no Atelier

Англійська назва:

Witch Hat Atelier

Японська назва:

とんがり帽子のアトリエ

Альтернативні назви:

Скорочені та альтернативні назви
MHA

Тип:

ТБ-серіал

Кількість епізодів:

13

Студія

BUG FILMS

Пов'язані твори

Пошук пов'язаного твору:

Naruto, JoJo, One Piece...

Пов'язані твори ще не вибрані.

Рис. 3.27. Форма створення та редагування твору

Та сама форма використовується і для створення, і для редагування твору, змінюючи лише підпис кнопки та спосіб збереження. Вона охоплює назви кількома мовами (українською, ромадзі, англійською та японською) і альтернативні назви, опис, тип, статус, дату виходу, віковий рейтинг, для аніме кількість і тривалість епізодів та студію, а для манги томи, розділи й видавництво. Окремими блоками задають жанри, персонажів із їхніми ролями, причетних осіб за категоріями та зв'язки з іншими творами. Введені дані перевіряються на обов'язкові поля та на збіг назви з уже наявним твором того ж типу й року. Окремий блок форми відповідає за уніфікацію назв (рис. 3.28).

Сторінка «Пропозиції контенту» зводить усі подані користувачами зміни до творів, персонажів і людей у спільну чергу. Кожен запис показує номер, дату, автора, поле, цільовий твір і позначку стану, а позначки-фільтри вгорі («Усі», «Очікує», «Схвалено», «Відхилено») звужують перелік за станом розгляду. Щоб ухвалити рішення, адміністратор розгортає окрему пропозицію (рис. 3.30).

Запропоновано:



ПРИЧИНА

відсутній постер

Коментар модератора

Схвалити

Відхилити

Рис. 3.30. Розгляд окремої пропозиції

Розгорнувши запис прямо в переліку, адміністратор бачить поле, що змінюється, попереднє й запропоноване значення та причину автора. Для пропозицій щодо назви додатково показується попередження про можливі дублікати, щоб модератор звірив дані перед рішенням. Адміністратор може схвалити пропозицію, після чого зміна застосовується до твору автоматично, або відхилити її; до обох рішень за бажанням додають коментар модератора. Стан повноти й коректності даних відстежують на сторінці аудиту (рис. 3.31).

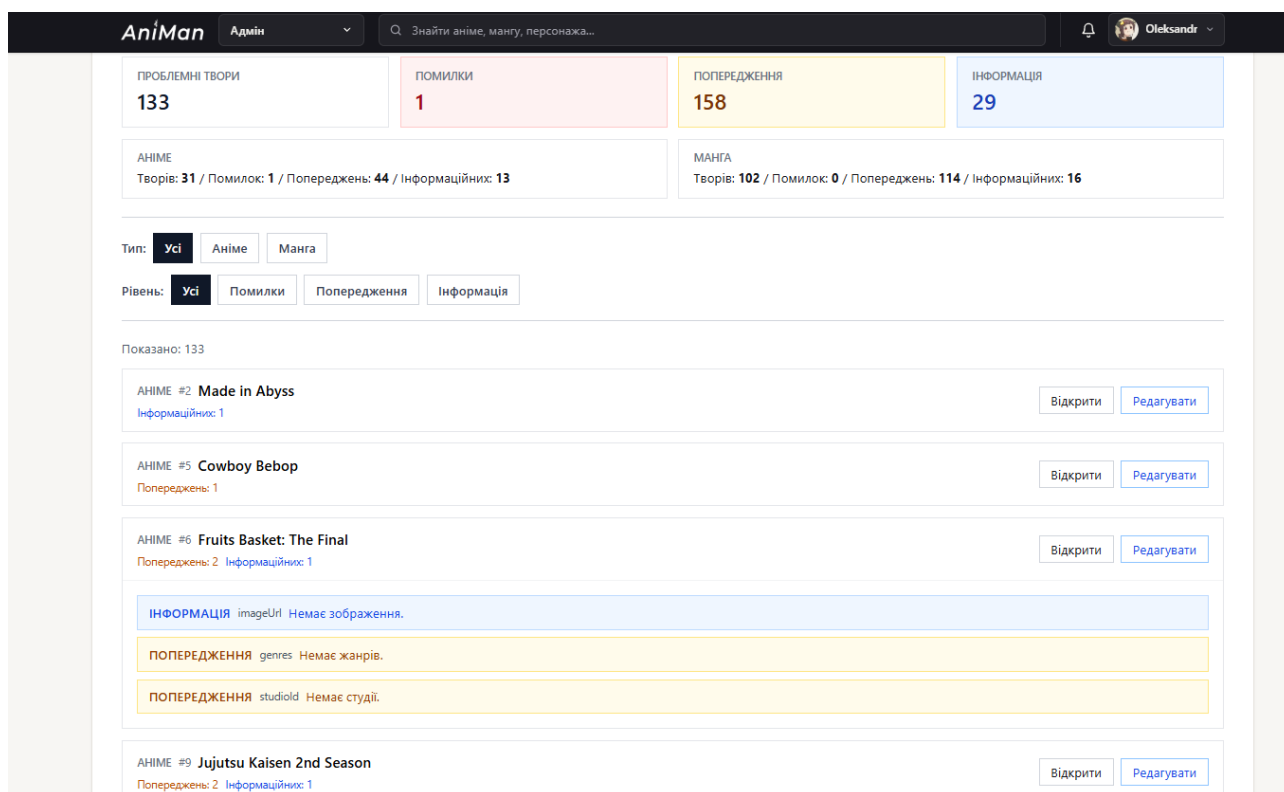


Рис. 3.31. Аудит якості даних

Сторінка «Аудит якості медіа» показує виявлені в каталозі помилки, попередження та інформаційні зауваги, як-от відсутні обов'язкові поля, некоректні числові значення, назви в неправильній системі письма чи збіг назви з alias іншого твору. Зведені лічильники та фільтри за типом і рівнем серйозності допомагають швидко відібрати проблемні твори, кожен з яких можна одразу відкрити чи відредагувати. Звіт призначений лише для перегляду, а розсилання системних оголошень усім користувачам винесено на окрему сторінку. Розмежування всіх адміністративних дій спирається на ту саму систему можливостей, що й серверні перевірки, тож інтерфейс показує лише доступне конкретному користувачеві.

3.3. Тестування програмного забезпечення

Для перевірки коректності та надійності платформи застосовано кілька рівнів тестування, що доповнюють один одного. Модульні тести перевіряють окремі функції бізнес-логіки у відриві від зовнішніх залежностей, інтеграційні підтверджують правильність взаємодії з базою даних, а навантажувальні

оцінюють поведінку системи за умов інтенсивного використання. Такий поділ дає змогу швидко локалізувати помилку, адже збій модульного тесту вказує на конкретну функцію, тоді як інтеграційний чи навантажувальний виявляють проблеми на рівні взаємодії компонентів або продуктивності. Усі автоматизовані тести інтегровано в конвеєр безперервної інтеграції, тож вони виконуються під час кожного оновлення коду й не допускають потрапляння помилок у робочу версію.

Модульне тестування реалізовано засобами фреймворку Vitest і охоплює чисту логіку системи, незалежну від бази даних та зовнішніх сервісів. Тестами покрито найскладніші обчислювальні частини рекомендаційної системи, як-от зважування сигналів користувача, обчислення жанрової спорідненості за оберненою частотою, адаптивне змішування контентного й колаборативного каналів та переранжування за різноманіттям, а також розбір структурованих відповідей інтелектуального пошуку, обчислення розкладу виходу серій і формування сповіщень та правила роботи зі списками й обраним. Кожна функція перевіряється на типових і граничних значеннях, що дає змогу впевнено змінювати реалізацію без ризику порушити наявну поведінку.

Якість покриття контролюється автоматично під час проходження тестів. Вимірювання охоплює саме логічний шар системи, тоді як шари, що залежать від бази даних і зовнішніх сервісів, перевіряються інтеграційними та навантажувальними тестами, тому їх свідомо виключено з підрахунку покриття. За такого підходу покриття логічного шару становить близько 96 % за рядками, а звіт автоматично формується конвеєром і відображається індикатором у репозиторії. Зведений звіт про покриття наведено на рис. 3.32.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	79
		Савіцький Р.С.				

File	% Stmts	% Branch	% Funcs	% Lines	Uncovered Line #s

All files	95.94	89.85	96.59	95.83	
auth/helpers	100	100	100	100	
passwordChange.ts	100	100	100	100	
permissions.ts	100	100	100	100	
calendar/helpers	96.66	88.88	94.44	96.49	
airingStatus.ts	100	100	100	100	
kyivTime.ts	96	80	93.75	95.74	24,110
lifecycleStatus.ts	100	100	100	100	
favorites/helpers	100	100	100	100	
favorites.ts	100	100	100	100	
notifications/helpers	97.87	97.56	100	97.87	
notificationTabs.ts	100	100	100	100	
notificationView.ts	91.66	90.9	100	91.66	42
relativeTime.ts	100	100	100	100	
recommendations/data	96.11	89.39	98.95	96.19	
coldStart.ts	88.88	75	100	88.88	23
collaborativeSimilarity.ts	97.43	93.75	100	97.43	120
crossMediaAdaptations.ts	94.44	81.81	100	94.44	24,33
genreAffinity.ts	100	80	100	100	19,48
hiddenGems.ts	96.55	90	100	96	35
hybridBlend.ts	100	100	100	100	
hybridScore.ts	100	100	100	100	
implicitSignals.ts	96.55	96.55	100	96.42	115
mmrRerank.ts	96.22	80	100	95.83	17,42
popularityScore.ts	100	100	100	100	
qualityScore.ts	100	100	100	100	
recommendationExclusions.ts	100	100	100	100	
recommendationPreferences.ts	92.68	84.37	100	92.68	85,93,97
recommendationWeights.ts	93.42	90.14	100	94.52	63,142,205,209
season.ts	96.77	82.75	90	96.42	32
recommendations/helpers	92.68	79.06	93.33	92.5	
recommendationFeedback.ts	100	100	100	100	
recommendationFilters.ts	89.65	72.72	90.9	89.28	20,33,61
search	96.33	93.15	96.55	95.74	
aiParse.ts	95.58	90	100	95.38	82,94,106
entityFacts.ts	100	100	100	100	
types.ts	90.9	100	80	87.5	46
search/data	85.71	83.33	60	80	
aiExamples.ts	85.71	83.33	60	80	29-34

===== Coverage summary =====					
Statements : 95.94% (709/739)					
Branches : 89.85% (496/552)					
Functions : 96.59% (170/176)					
Lines : 95.83% (667/696)					
=====					

Рис. 3.32. Звіт про покриття

Інтеграційне тестування перевіряє роботу системи разом із реальною базою даних. Перед кожним запуском тести наповнюють тимчасову базу узгодженим набором даних, виконують прикладні сценарії та звіряють отримані результати з очікуваними, після чого очищають базу, щоб запуски не впливали один на одного. Перевіряються каталогізація творів, персонажів і причетних осіб, формування рекомендацій за реальними сигналами користувачів, робота

інтелектуального пошуку та обробка коментарів. Такі тести підтверджують коректність запитів, узгодженість зв'язків між сутностями й цілісність даних. Оскільки вони потребують підключення до бази, їх винесено в окреме завдання конвеєра з ручним запуском, щоб залежність від зовнішнього середовища не блокувала основний процес доставки.

Навантажувальне тестування виконано інструментом Apache JMeter для оцінювання продуктивності системи під одночасним навантаженням. План тестування відтворює звернення багатьох користувачів до ключових сторінок платформи, каталогу, сторінок творів і пошуку, із поступовим зростанням кількості одночасних потоків. Для кожного рівня навантаження фіксувалися середній час відповіді, пропускна здатність і частка помилкових запитів.

Результати тестування показали, що за навантаження до ста одночасних користувачів середній час відповіді не перевищує однієї секунди, а частка помилкових запитів залишається близькою до нуля, що відповідає нефункціональним вимогам. Зі зростанням кількості користувачів час відповіді очікувано збільшується, проте система зберігає працездатність завдяки автоматичному масштабуванню безсерверного середовища.

Навантажувальне тестування виконано засобом Apache JMeter для оцінювання продуктивності системи під одночасним навантаженням. План відтворює типовий сценарій роботи через п'ять ключових HTTP-ендпоінтів, серед яких пошук у каталозі, вхід, ведення списку, подання пропозиції зміни та службова синхронізація розкладу. Сценарій запускали з 30 одночасними потоками по 10 ітерацій, що дало 1500 запитів; для кожного ендпоінта фіксували час відповіді, пропускну здатність і частку помилкових запитів. Зведені результати наведено в таблиці 3.1.

Усі 1500 запитів виконано без помилок. Найшвидше відповідає ендпоінт сповіщень (у середньому 436 мс), тоді як створення пропозиції зміни найповільніше (1425 мс) через запис до бази даних. Загальна пропускна здатність склала близько 32 запитів за секунду, а середній час відповіді по всіх запитах не перевищив однієї секунди (861 мс), що відповідає нефункціональним вимогам.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	81
		Савіцький Р.С.				

Таблиця 3.1

Результати навантажувального тестування

Запит	Зразків	Середній, мс	90-й перц., мс	Макс, мс	Пропускн а, зап/с	Помилки, %
GET /api/search	300	664	807	1012	6.5	0
POST /api/auth/login	300	682	844	996	6.6	0
PUT /api/user-list/anime/{id}	300	1099	1310	1477	6.6	0
POST /api/content-suggestions	300	1425	1717	1885	6.6	0
GET /api/cron/notify-episodes	300	436	598	801	6.6	0
Усього	1500	861	1489	1885	32.0	0

Розподіл часу відповіді за перцентилями для кожного ендпоінта показано на рис. 3.33.

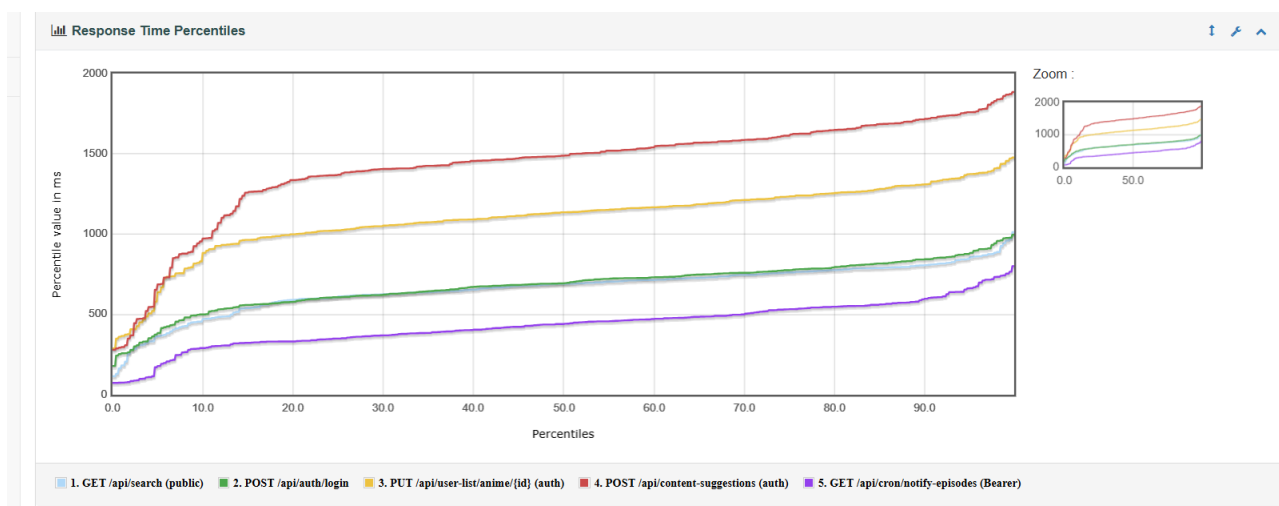


Рис. 3.33. Графік JMeter

Графік підтверджує, що для переважної частини запитів час відповіді лишається помірним, а помітне зростання у верхніх перцентилях стосується операцій запису до бази даних, насамперед створення пропозиції зміни.

Усі модульні та інтеграційні тести інтегровано в конвеєр безперервної інтеграції. Модульні тести разом із перевіркою типів і контролем покриття

виконуються автоматично під час кожного оновлення коду, тож регресії виявляються одразу, ще до збірки й розгортання. Контроль покриття слугує додатковим запобіжником, не допускаючи зниження якості тестування під час подальшого розвитку платформи.

Поєднання модульного, інтеграційного та навантажувального тестування забезпечує контроль як коректності окремих компонентів, так і поведінки системи загалом. Автоматичне виконання тестів у конвеєрі перетворює тестування з разової перевірки на постійний процес, що супроводжує кожну зміну й підтримує надійність платформи на сталому рівні.

Висновки до третього розділу

У третьому розділі описано розгортання, інтерфейс і тестування платформи. Серверну та клієнтську частини розгорнуто у хмарному середовищі Vercel із базою даних PostgreSQL у сервісі Neon, а оновлення коду проходять через конвеєр безперервної інтеграції в GitLab, який автоматично виконує перевірки й доставку. Така організація забезпечує відтворюване розгортання та зменшує ручну працю під час випуску нових версій.

Інтерфейс реалізовано як адаптивний вебзастосунок із трьома режимами роботи (гостьовим, режимом автентифікованого користувача та адміністративним), обсяг доступних дій у яких визначає система ролей. Описано основні сторінки кожного режиму, ведення списків, персональні рекомендації, звичайний та інтелектуальний пошук, сповіщення про вихід серій і засоби адміністрування контенту.

Працездатність платформи підтверджено кількома рівнями тестування. Модульні тести охоплюють чисту логіку системи з покриттям близько 96% за основними метриками, інтеграційні перевіряють взаємодію з базою даних, а навантажувальне тестування засобом Apache JMeter показало стабільну роботу ключових ендпоінтів без помилок за одночасного звернення багатьох користувачів. Окреслено також напрями подальшого розвитку платформи.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	83
		Савіцький Р.С.				

ВИСНОВКИ

У кваліфікаційній роботі розв'язано задачу створення україномовної вебплатформи аніме та манги з рекомендаційною системою на основі користувацьких списків. Поставлену мету досягнуто, а всі визначені завдання виконано.

У результаті аналізу предметної області та порівняння чотирьох провідних платформ (MyAnimeList, Anime News Network, AniList і Kitsu) з'ясовано, що кожна з них сильна лише в окремому напрямі, а жодна не поєднує структурований каталог, персоналізовані рекомендації та інтелектуальний пошук у межах єдиного рішення. Усі вони лишаються іншомовними, тому якісного україномовного сервісу досі не існувало. На основі цього аналізу сформовано функціональні та нефункціональні вимоги й обґрунтовано вибір технологій і архітектури.

Спроектовано та реалізовано клієнт-серверний вебзастосунок із гібридним підходом до рендерингу на основі Next.js і реляційної бази даних PostgreSQL. Побудовано модель предметної області з основними сутностями, як-от твори, персонажі, причетні особи, персональні списки, відгуки, зв'язки франшиз і розклад виходу серій, із дотриманням нормалізації та цілісності даних. Реалізовано каталогізацію, ведення персональних списків, систему відгуків і коментарів, а також розмежування доступу за ролями на основі набору можливостей.

Розроблено ключові алгоритмічні складові платформи. Гібридна рекомендаційна система поєднує контентну фільтрацію з ваговим оцінюванням жанрів за оберненою частотою та колаборативну фільтрацію за косинусною подібністю користувачів, доповнені переранжуванням за різноманіттям і початковим опитуванням для пом'якшення холодного старту. Інтелектуальний пошук тлумачить запити природною мовою з кешуванням, обмеженням частоти та резервуванням постачальників, а підсистема синхронізації відстежує вихід нових серій і формує сповіщення. Автентифікацію побудовано на підписаних сесійних токенах, хешуванні паролів та вході через зовнішні сервіси.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	
		Савіцький Р.С.				
						84

Платформу розгорнуто у хмарному середовищі з автоматизованим конвеєром безперервної інтеграції та доставки, що виконує статичний аналіз, перевірку типів, тестування, збірку й розгортання. Працездатність підтверджено модульним, інтеграційним і навантажувальним тестуванням. Модульні тести охоплюють основну логіку з покриттям близько 96%, а навантажувальне тестування засобом Apache JMeter підтвердило стабільну роботу ключових ендпоінтів без помилок із пропускнуою здатністю близько 32 запитів за секунду.

Практична цінність роботи полягає у створенні готового до використання україномовного сервісу, що закриває потребу вітчизняної аудиторії в єдиному інструменті для каталогізації, персоналізованих рекомендацій та відстеження аніме й манги. Закладена архітектура й розроблені алгоритми придатні для подальшого розвитку платформи та можуть слугувати прикладом побудови подібних веборієнтованих інформаційних систем.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Залозний О. А., Савіцький Р. С. Розробка вебплатформи для каталогізації та аналізу аніме та манги. Тези XIX Всеукраїнської науково-практичної web-конференції аспірантів, студентів та молодих учених «Комп'ютерні інтелектуальні системи та мережі», 25–27 березня 2026 р. Кривий Ріг : Криворізький національний університет, 2026. С. 112.
2. Залозний О. А., Савіцький Р. С. Організація зберігання повідомлень, сповіщень та дій користувачів у вебплатформі аніме й манги. Тези XI Міжнародної науково-технічної конференції «Інформаційно-комп'ютерні технології», 02–03 квітня 2026 р. Житомир : Житомирська політехніка, 2026.
3. MyAnimeList. URL: <https://myanimelist.net> (дата звернення: 14.02.2026).
4. Anime News Network. URL: <https://www.animenewsnetwork.com> (дата звернення: 26.02.2026).
5. AniList. URL: <https://anilist.co> (дата звернення: 05.03.2026).
6. Kitsu. URL: <https://kitsu.io> (дата звернення: 18.03.2026).
7. Next.js Documentation. URL: <https://nextjs.org/docs> (дата звернення: 30.03.2026).
8. React Documentation. URL: <https://react.dev> (дата звернення: 08.04.2026).
9. PostgreSQL Documentation. URL: <https://www.postgresql.org/docs/> (дата звернення: 16.04.2026).
10. Prisma ORM Documentation. URL: <https://www.prisma.io/docs> (дата звернення: 24.04.2026).
11. REST API Tutorial. URL: <https://restfulapi.net/> (дата звернення: 07.05.2026).
12. Aggarwal C. C. Recommender Systems: The Textbook. Cham : Springer, 2016. 498 с.
13. Ricci F., Rokach L., Shapira B. Recommender Systems Handbook. 2nd ed. New York : Springer, 2015. 1003 с.

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	
		Савіцький Р.С.				
						86

14. Sarwar B., Karypis G., Konstan J., Riedl J. Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th International Conference on World Wide Web. 2001. С. 285–295.
15. Koren Y., Bell R., Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009. Т. 42, вип. 8. С. 30–37.
16. Carbonell J., Goldstein J. The use of MMR, diversity-based reranking for reordering documents and producing summaries. Proceedings of the 21st ACM SIGIR Conference. 1998. С. 335–336.
17. Salton G., Wong A., Yang C. S. A vector space model for automatic indexing. Communications of the ACM. 1975. Т. 18, вип. 11. С. 613–620.
18. Salton G., Wong A., Yang C. S. A vector space model for automatic indexing. Communications of the ACM. 1975. Т. 18, вип. 11. С. 613–620.
19. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation / University of California, Irvine. 2000. URL: <https://roy.gbiv.com/pubs/dissertation/top.htm> (дата звернення: 11.02.2026).
20. Fielding R., Nottingham M., Reschke J. HTTP Semantics : RFC 9110 / IETF. 2022. URL: <https://www.rfc-editor.org/rfc/rfc9110> (дата звернення: 20.02.2026).
21. ECMAScript 2025 Language Specification (ECMA-262). 16th ed. Geneva : Ecma International, 2025. URL: <https://ecma-international.org/publications-and-standards/standards/ecma-262/> (дата звернення: 03.03.2026).
22. TypeScript Documentation / Microsoft. URL: <https://www.typescriptlang.org/docs/> (дата звернення: 23.03.2026).
23. Codd E. F. A relational model of data for large shared data banks. Communications of the ACM. 1970. Т. 13, вип. 6. С. 377–387.
24. The OAuth 2.0 Authorization Framework : RFC 6749 / IETF ; ed. D. Hardt. 2012. URL: <https://www.rfc-editor.org/rfc/rfc6749> (дата звернення: 04.04.2026).

25. Proof Key for Code Exchange by OAuth Public Clients : RFC 7636 / IETF ; eds. N. Sakimura, J. Bradley, N. Agarwal. 2015. URL: <https://www.rfc-editor.org/rfc/rfc7636> (дата звернення: 14.04.2026).
26. Krawczyk H., Bellare M., Canetti R. HMAC: Keyed-Hashing for Message Authentication : RFC 2104 / IETF. 1997. URL: <https://www.rfc-editor.org/rfc/rfc2104> (дата звернення: 28.04.2026).
27. Percival C., Josefsson S. The scrypt Password-Based Key Derivation Function : RFC 7914 / IETF. 2016. URL: <https://www.rfc-editor.org/rfc/rfc7914> (дата звернення: 09.05.2026).
28. OWASP Top 10:2021 / Open Worldwide Application Security Project. 2021. URL: <https://owasp.org/Top10/> (дата звернення: 19.05.2026).
29. Humble J., Farley D. Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation. Boston : Addison-Wesley, 2010. 512 с.
30. Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Boston : Prentice Hall, 2017. 432 с.
31. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Reading : Addison-Wesley, 1994. 395 с.

ДОДАТКИ

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	
		Савіцький Р.С.				
						89

Лістинги програмного коду:

Лістинг А.1 – Ініціалізація клієнта Prisma з адаптером Neon

```
// src/lib/prisma.ts
import {PrismaClient} from "@generated/prisma/client";
import {PrismaPg} from "@prisma/adapter-pg";

const globalForPrisma = globalThis as unknown as {
  prisma: PrismaClient | undefined;
};

const adapter = new PrismaPg({connectionString: process.env.DATABASE_URL!});

export const prisma =
  globalForPrisma.prisma ?? new PrismaClient({adapter});

if (process.env.NODE_ENV !== "production") {
  globalForPrisma.prisma = prisma;
}
```

Лістинг А.2 – Приклад функції шару доступу до даних

```
// src/lib/data/home.ts
export async function getHomeOngoingAnime() {
  const anime = await prisma.anime.findMany({
    where: {status: "Виходить"},
    select: animeCardSelect,
    orderBy: [
      {userAnimeLists: {_count: "desc"}},
      {dateRelease: "desc"},
    ],
    take: HOME_POOL_LIMIT,
  });
  return pickRandomItems(anime);
}
```

Лістинг А.3 – Обробник оновлення запису персонального списку

```
// src/app/api/user-list/anime/[id]/route.ts (фрагмент)
export async function PUT(request: Request, {params}: IdPromise) {
  const user = await getCurrentUser();
  if (!user) {
```

```

        return NextResponse.json({error: "Потрібно увійти"}, {status: 401});
    }

    const {id} = await params;
    const animeId = Number(id);
    const body = await request.json();

    if (!isAnimeListStatus(body.status)) {
        return NextResponse.json({error: "Некоректний статус"}, {status: 400});
    }
    const score = validateScore(body.score);

    const listItem = await prisma.userAnimeList.upsert({
        where: {userId_animeId: {userId: user.id, animeId}},
        create: {userId: user.id, animeId, status: body.status, episodes, score,
note},
        update: {status: body.status, episodes, score, note},
    });

    return NextResponse.json({success: true, listItem});
}

```

Лістинг А.4 – Проміжне ПЗ захисту маршрутів

```

// src/proxy.ts (фрагмент)
const protectedRoutes = ["/profile", "/profile/settings"];
const adminPrefixes = ["/admin", "/anime/update", "/manga/update"];
export async function proxy(request: NextRequest) {
    const {pathname} = request.nextUrl;
    const token = request.cookies.get(SESSION_COOKIE)?.value;
    const session = token ? await verifySessionToken(token) : null;
    if (!session && (isProtectedRoute(pathname) || isAdminRoute(pathname))) {
        const loginUrl = new URL("/auth/login", request.url);
        loginUrl.searchParams.set("next", pathname);
        return NextResponse.redirect(loginUrl);
    }
    if (session && isAdminRoute(pathname)) {
        const capabilities = getCapabilities(session.roles ?? []);
        if (!can(capabilities, requiredCapability(pathname))) {
            return NextResponse.redirect(new URL("/", request.url));
        }
    }
}

```

```

}
return NextResponse.next();
}

```

Лістинг А.5 – Визначення дизайн-токенів

```

/* src/styles/globals.css (фрагмент) */
@import "tailwindcss";

```

```

:root {
  --am-bg: #f7f6f3;
  --am-surface: #ffffff;
  --am-ink: #16161c;
  --am-primary: #5b62ec;
  --am-success: #4a8f6a;
  --am-danger: #c44a3b;
}

```

```

@theme inline {
  --color-am-bg: var(--am-bg);
  --color-am-primary: var(--am-primary);
}

```

Лістинг А.6 – Обчислення ваги твору за неявними сигналами

```

// src/features/recommendations/data/implicitSignals.ts (фрагмент)
export function getImplicitListWeight(status, score, progressRatio = null) {
  const config = IMPLICIT_SIGNAL_CONFIG;

  if (isRewatchListStatus(status)) { // повторний перегляд
    const bonus = typeof score === "number"
      ? Math.max(0, getRatingWeight(score)) * config.rewatchScoredBonus : 0;
    return clamp(config.rewatchWeight + bonus, config.minWeight,
config.maxWeight);
  }
  if (typeof score === "number") { // є явна оцінка
    return clamp(getRatingWeight(score), config.minWeight, config.maxWeight);
  }
  if (isDroppedListStatus(status)) { // кинуто рано / пізно
    return progressRatio !== null && progressRatio < config.earlyDropThreshold
      ? config.earlyDropWeight : config.lateDropWeight;
  }
  if (isCompletedListStatus(status)) return config.completedNoScoreWeight;
}

```

		Залозний О.А.			ІПЗ.КР.Б-121-26-ПЗ	92
		Савіцький Р.С.				

```

    if (isActiveListStatus(status)) return config.activeWeight;
    if (isPlannedListStatus(status)) return config.plannedIntentWeight;
    return config.defaultWeight;
}

```

Лістинг А.7 – IDF-зважена жанрова спорідненість

```

// src/features/recommendations/data/genreAffinity.ts
export function getGenreIdf(genreDocFrequency, totalDocuments) {
    const idf = new Map();
    for (const [genreId, df] of genreDocFrequency) {
        if (df > 0) idf.set(genreId, Math.log(totalDocuments / df));
    }
    return idf;
}

//  $\sum \log(1 + \text{var}_a(g)) \times \text{idf}(g)$ 
export function getGenreAffinity(candidateGenreIds, userGenreWeights, genreIdf) {
    let sum = 0;
    for (const genreId of candidateGenreIds) {
        const idf = genreIdf.get(genreId) ?? 0;
        if (idf <= 0) continue;
        const weight = Math.max(0, userGenreWeights.get(genreId) ?? 0);
        sum += Math.log1p(weight) * idf;
    }
    return sum;
}

```

Лістинг А.8 – Косинусна схожість користувачів з урахуванням довіри

```

// src/features/recommendations/data/collaborativeSimilarity.ts (фрагмент)
// similarity = cosine × confidence; confidence = common / (common + damping)
export function getWeightedSimilarityDetails(leftVector, rightVector, damping = 2)
{
    let dotProduct = 0, commonCount = 0;
    const leftNorm = getVectorNorm(leftVector);
    const rightNorm = getVectorNorm(rightVector);

    for (const [key, leftWeight] of leftVector) {
        const rightWeight = rightVector.get(key) ?? 0;
        if (leftWeight <= 0 || rightWeight <= 0) continue;
        dotProduct += leftWeight * rightWeight;
        commonCount += 1;
    }
}

```

```

    }
    const cosine = leftNorm > 0 && rightNorm > 0
      ? Math.max(0, Math.min(1, dotProduct / (leftNorm * rightNorm))) : 0;
    const confidence = commonCount / (commonCount + damping);

    return {cosine, confidence, similarity: cosine * confidence};
  }

```

Лістинг А.9 – Адаптивне змішування контентного та колаборативного каналів

```

// src/features/recommendations/data/hybridBlend.ts + hybridScore.ts (фрагменти)
export function getAdaptiveContentShare(interactionCount, type) {
  const config = getAdaptiveBlendConfig(type);
  const raw = 1 / (1 + Math.max(0, interactionCount) / config.halfLife);
  return clamp(raw, config.minContentShare, config.maxContentShare);
}

export function blendChannels(content, collaborative, contentWeight,
collaborativeWeight, contentFallback = true) {
  if (contentFallback && collaborative <= 0) return content;
  const total = contentWeight + collaborativeWeight;
  return (content * contentWeight + collaborative * collaborativeWeight) / total;
}

```

Лістинг А.10 – Виклик мовної моделі зі структурованою схемою

```

// src/features/search/data/aiSearch.ts (фрагмент)
export async function runAiSearch(query: string): Promise<AiSearchResult> {
  const genres = await getGenreVocabulary();
  const systemPrompt = buildSystemPrompt(formatFilterVocabulary(genres));

  const result = await chatWithFallback(
    [
      {role: "system", content: systemPrompt},
      {role: "user", content: query},
    ],
    {responseSchema: RESPONSE_SCHEMA, temperature: 0.3, maxTokens: 1600},
  );

  const raw = safeParseJson(result.text);
  if (!raw) {

```

```

        return {relevant: true, message: "Не вдалося розібрати відповідь AI.",
blocks: []};
    }
    if (!raw.relevant) {
        return {relevant: false, message: raw.message?.trim() || "...", blocks:
[]};
    }
    // ... резолюція карток і фільтрів у реальні записи бази даних
}

```

Лістинг А.11 – Ланцюг провайдерів із резервуванням і повторами

// src/lib/ai/fallback.ts (фрагмент)

```

export async function chatWithFallback(messages, opts, models): Promise<LlmResult>
{
    const chain = buildChain(models); // OpenAI → безкоштовні моделі OpenRouter

    for (const entry of chain) {
        if (isCoolingDown(entry.label)) continue;
        try {
            return await entry.provider.chat(messages, opts);
        } catch (firstError) {
            const first = classify(firstError);
            if (first.status === 429) cooldowns.set(entry.label, Date.now() +
COOLDOWN_MS);

            if (!first.transient) continue; // постійна помилка – наступний
провайдер

            await delay(RETRY_BACKOFF_MS); // тимчасова – один повтор
            try {
                return await entry.provider.chat(messages, opts);
            } catch { /* перехід до наступного провайдера */ }
        }
    }
    throw new Error("chatWithFallback exhausted all providers");
}

```

Лістинг А.12 – Кешування та обмеження частоти AI-пошуку

// src/app/api/search/ai/route.ts (фрагмент)

```

export async function POST(request: Request) {
    const user = await getCurrentUser();
    if (!user || !canUseAiSearch(user)) {
        return NextResponse.json({error: "AI-пошук доступний лише після входу."},
{status: 401});
    }
}

```

```

    }

    const query = /* валідація: від 2 до 300 символів */;

    // Кеш: той самий запит за останню годину – без виклику LLM.
    const cached = await prisma.searchQueryLog.findFirst({
        where: {userId: user.id, mode: "AI", query, createdAt: {gte: new
Date(Date.now() - CACHE_WINDOW_MS)}}},
        orderBy: {createdAt: "desc"},
    });
    if (cached?.interpretedFilters) {
        return NextResponse.json({...cached.interpretedFilters, cached: true});
    }

    const limit = await checkAiRateLimit(user.id);    // ліміт лише для реальних
запитів
    if (!limit.ok) {
        return NextResponse.json({error: "Ліміт вичерпано."}, {status: 429});
    }

    // Резервуємо слот ДО виклику моделі (закриває гонку «вийшов за ліміт»).
    const reserved = await prisma.searchQueryLog.create({data: {userId: user.id,
mode: "AI", query}});
    const result = await runAiSearch(query);
    await prisma.searchQueryLog.update({
        where: {id: reserved.id},
        data: {interpretedFilters: JSON.parse(JSON.stringify(result)), aiModel:
result.model},
    });

    return NextResponse.json(result);
}

```

Лістинг А.13 – Резолюція назв від моделі в записи бази даних

```

// src/features/search/data/aiSearch.ts (фрагмент)
async function resolveEntities(type, names): Promise<{id: number; title:
string}[]> {
    const out = [];
    for (const name of names.slice(0, MAX_FILTER_ENTITIES)) {
        const trimmed = name.trim();
        if (trimmed.length < 2) continue;

```

```

        let matches = await searchByType(type, trimmed, 8);      // точний пошук
        if (matches.length === 0) {
            matches = await searchByTokens(type, trimmed, 8);    // пошук за словами
        }
        const best = pickBestMatch(trimmed, matches);            // евристика
добору
        if (best) out.push({id: best.id, title: best.title});
    }
    return out;
}

```