

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Вебплатформа для читання електронних книжок»

Виконав студент групи ІПЗ-22-4

ПЕТРОВ Денис Володимирович

Керівник роботи:

ЄФРЕМОВ Юрій Миколайович

Рецензент:

ВЛАСЕНКО Олег Васильович

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: ПЕТРОВ Денис Володимирович

Керівник роботи: ЄФРЕМОВ Юрій Миколайович

Тема роботи: «Вебплатформа для читання електронних книжок»,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026 р., № 92/с**

Вихідні дані для роботи: предметом дослідження є процес взаємодії користувачів із цифровим контентом у форматі книжок та механізми керування персональними бібліотеками у вебсередовищі;

об'єктом дослідження є методи та інструментальні засоби розробки вебплатформи для публікації, читання та інтерактивної взаємодії з книжками.

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Єфремов Ю. М.	01.03.2026р.	01.03.2026р
2	Єфремов Ю. М.	01.03.2026р	01.03.2026р
3	Єфремов Ю. М.	01.03.2026р	01.03.2026р

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра виконана на тему «Вебплатформа для читання електронних книжок». Робота складається зі вступу, трьох розділів, висновків та переліку використаних джерел. Загальний обсяг записки становить 108 сторінок, робота містить 79 рисунків, 22 таблиць та 30 джерел літератури.

У пояснювальній записці розглянуто предметну область розповсюдження цифрового контенту, зокрема специфіку публікації та споживання книжок у вебсередовищі. Проаналізовано особливості архітектури сучасних сайтів для читання, механізми взаємодії між авторами, перекладачами та читачами, а також виявлено недоліки та обмеження існуючих сервісів для читання.

На підставі проведеного аналізу сформовано концептуальні та функціональні вимоги до програмної вебплатформи, архітектура якої базується на принципах модульного моноліту. Дана вебплатформа буде технічно реалізована в межах випускної кваліфікаційної роботи бакалавра для забезпечення зручного доступу до текстового контенту та оптимізації процесів керування бібліотекою.

Ключові слова: ЕЛЕКТРОННІ КНИЖКИ, ВЕБПЛАТФОРМА, РОЗДІЛИ, ТВОРИ, КОМАНДА ПЕРЕКЛАДУ, ЧИТАЧ.

					ІПЗ.КР.Б – 121 – 26 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа для читання електронних книжок	Літ.	Арк.	Аркушів
Розроб.		Петров Д.В.						
Перевір.		Єфремов Ю.М.					2	124
Керівник						Житомирська політехніка, група ІПЗ-22-4		
Н. контр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ABSTRACT

The bachelor's qualification work was carried out on the topic "Web platform for reading electronic books". The work consists of an introduction, three sections, conclusions and a list of sources used. The total volume of the note is 108 pages, the work contains 79 figures, 22 tables and 30 sources of literature.

The explanatory note considers the subject area of digital content distribution, in particular the specifics of publishing and consuming books in the web environment. The features of the architecture of modern reading sites, mechanisms of interaction between authors, translators and readers are analyzed, and the shortcomings and limitations of existing reading services are identified.

Based on the analysis, conceptual and functional requirements for a software web platform, the architecture of which is based on the principles of a modular monolith, were formed. This web platform will be technically implemented within the framework of the bachelor's qualification work to ensure convenient access to text content and optimize library management processes.

Keywords: E-BOOKS, WEB PLATFORM, CHAPTERS, WORKS, TRANSLATION TEAM, READER.

					ІПЗ.КР.Б – 121 – 26 – ІЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Вебплатформа для читання електронних книжок	Літ.	Арк.	Аркуші
Розроб.		Петров Д.В.						
Перевір.		Єфремов Ю.М.					2	124
Керівник						Житомирська політехніка, група ІПЗ-22-4		
Н. контр.		Тичина О.П.						
Зав. каф.		Вакалюк Т.А.						

ЗМІСТ

ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ	10
1.1. Аналіз аналогів програмного продукту	10
1.2. Постановка завдання вебплатформи	16
1.3. Обґрунтування вибору архітектури системи	19
1.4. Обґрунтування вибору інструментальних засобів та вимоги апаратного забезпечення	25
Висновки до першого розділу.....	28
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИЖОК.....	30
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	30
2.2. Розробка бази даних системи.....	37
2.3. Проєктування та реалізація алгоритмів роботи системи	51
2.4. Реалізація функціоналу вебплатформи для читання книжок	61
Висновки до другого розділу	65
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИЖОК	67
3.1. Визначення варіантів використання структури системи	67
3.2. Структура інтерфейсу вебплатформи	70
3.3. Тестування вебплатформи.....	101
Висновки до третього розділу.....	104
ВИСНОВКИ	105
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	106

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		5

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		6

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - прикладний програмний інтерфейс

БД - база даних

ПЗ - програмне забезпечення

СУБД - система управління базами даних

MVC - схема розділення даних додатка і керуючої логіки на три окремі компоненти: Model-View-Controller

CRUD - основні операції управління даними: Create, Read, Update, Delete

UI/UX - інтерфейс користувача та досвід взаємодії з користувачем

JSON - текстовий формат обміну даними, заснований на JavaScript

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		7

ВСТУП

Актуальність роботи. Сьогодні можна побачити, як електронні книжки поступово, але дуже впевнено витісняють традиційні паперову літературу. Люди звикли читати з гаджетів вдома, або в дорозі тому кількість читачів у мережі збільшується. У людей виникає цілком зрозуміла потреба в якісних, та надійних сайтів для читання. Але насправді виходить по іншому. Коли бібліотека розгортається до великої кількості файлів для зберігання, більшість вебплатформ цього не витримують. У результаті з'являються різні проблеми з функціями вебресурсу. Після аналізу було помічено спільну проблему, велика кількість вебплатформ мають незручний інтерфейс, який ускладнює навігацію. Через це, розробка абсолютно нової сучасної системи на базі архітектури модульного моноліту - це реальна необхідність. Такий підхід дасть змогу розбити логіку програми на різні блоки. А це у свою чергу, допоможе зробити сайт максимально швидким і зручним для користувача.

Мета роботи. Глибоко проаналізувати сферу розповсюдження електронних книг та створити чіткі вимоги для розробки власної вебплатформи. Щоб так трапилося потрібно провести порівняння вже наявних сервісів, щоб виявити їхні плюси та мінуси. Врешті-решт на основі проаналізованих даних, будуть сформовані загальні вимоги до майбутньої системи, ядром якої стане архітектура модульного моноліту.

Завдання роботи. Полягає у проведенні аналізу існуючих рішень у сфері електронних бібліотек та виявити їхні основні недоліки; сформулювати функціональні та нефункціональні вимоги до вебплатформи, що розробляється; спроектувати архітектуру програмного забезпечення та логічну структуру бази даних; розробити серверну та клієнтську частини системи; провести тестування розробленого програмного продукту та розгорнути його в робочому середовищі.

Об'єктом дослідження виступають механізми публікації, зберігання та саме головне читання електронних книжок на спеціальних сервісах.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

Предмет дослідження - це інструменти та методики, за допомогою яких можна створити комфортну вебплатформу для читання електронних книжок.

Методи дослідження. Для розв'язання поставлених завдань у роботі використано: методи системного та порівняльного аналізу; принципи об'єктно-орієнтованого проєктування; методи візуального моделювання; методи автоматизованого та ручного тестування.

За темою випускної кваліфікаційної роботи бакалавра було опубліковано тези XVI Міжнародної науково-технічної конференції Інформаційно-комп'ютерні технології - 2026, 02-03 квітня 2026 року. Житомир: «Житомирська політехніка», 2026. [1, 2].

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		9

РОЗДІЛ 1. АНАЛІЗ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

1.1. Аналіз аналогів програмного продукту

Сьогодні сайти для читання ранобе перетворилися на основний доступ цифрової літератури для багатьох користувачів молодшого віку. Ці не просто зберігають тексти, а ще й формують навколо себе активні спільноти читачів. Щоб правильно реалізувати власний сервіс потрібно проаналізувати вже існуючі українські аналоги. Для цього необхідні вебресурси, які мають велику кількість творів, мають змогу зручно відстежувати прогрес читання та створювати власні віртуальні бібліотеки. Для такого порівняння було обрано три вебплатформи: «Бака», «UAuthor» та «hikka».

Інтерфейс вебплатформи «Бака» (рис 1.1). функціонує як інформаційний каталог. Головна ціль цієї вебплатформи збирати, упорядковувати та збільшувати популярність аніме та ранобе в українському інтернеті. Цей вебсервіс створений як єдине місце, де приділяється увага до детального опису творів. Інакше кажучи користувачі бачать тут не просто назву. Вебплатформа пропонує класифікацію: від жанрів і тематичних тегів до вікових рейтингів та актуального статусу перекладу.

Робота з вебплатформою зроблена на створенні та налаштуванні власної бібліотеки. Читачі можуть виставляти оцінки творам за п'ятибальною шкалою, і так вираховується рейтинг на сайті. Також є можливість вести зручні списки читання для сортування творів. Користувачі можуть легко розмістити ранобе по спискам, «читаю», «прочитано», «відкладено» або «покинуто». Цей функціонал важливий, адже він допомагає утримувати увагу аудиторії на своєму вебресурсі.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		10

Таблиця 1.1.

Аналіз переваг та недоліків вебплатформи «Бака»

Переваги	Недоліки
Повна орієнтація на українську аудиторію та контент.	Відсутність інструментарію для команд перекладачів.
Детальна система фільтрації за тегами та рейтингами.	Низька якість модерації
Зручне структурування списків.	Зниження швидкодії при пікових навантаженнях.
Єдина екосистема з базою ранобе	Відсутність вбудованої гнучкої читалки контенту.
Наявність великої бази відгуків та рецензій від україномовної спільноти	Складність оновлення інформації про нові розділи в реальному часі
Гарні алгоритми персоналізованих рекомендацій.	Залежність користувача від стабільності роботи сторонніх сайтів-джерел

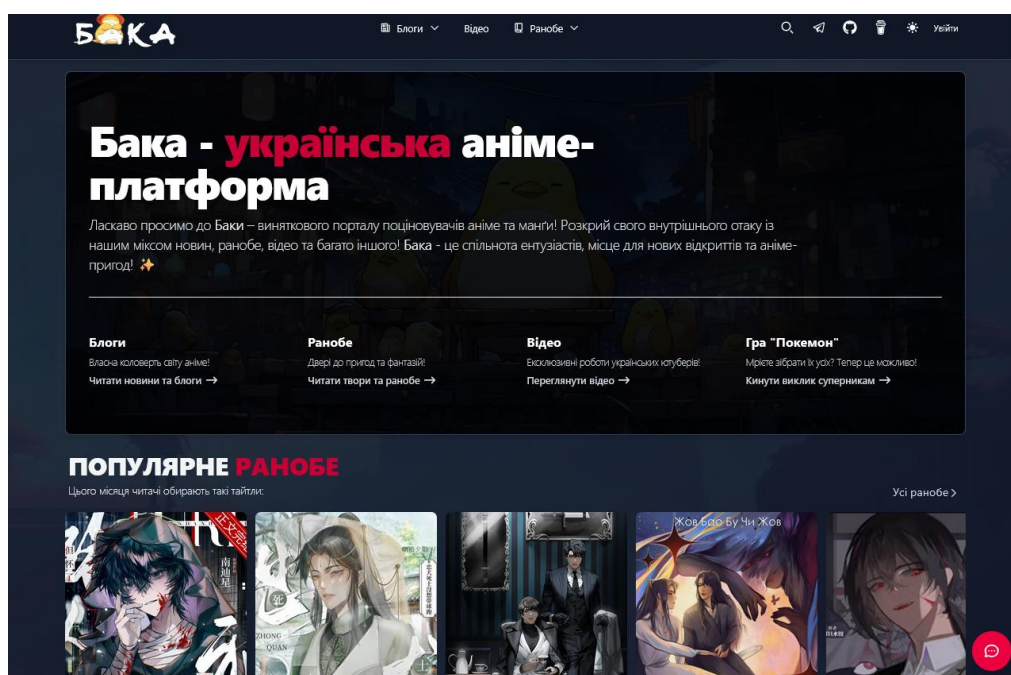


Рис.1.1. Інтерфейс сайту Бака

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Вебплатформа «UAuthor» (рис. 1.2) працює як збирач інформації, а не вебплатформи для публікації творів. На сайті не можна читати тексти розділів. Замість цього вебресурс надає детальну картку твору з його описом, жанрами, тегами. Головна функція вебплатформи - це зручна навігація. Зі сторінки ранобе користувач може перейти за зовнішніми посиланнями на інші сайти, де вже розміщено сам переклад. Також тут реалізована можливість ставити оцінки та залишати коментарі, що допомагає створювати незалежний рейтинг творів.

Інтерфейс вебплатформи зосереджений на пошуку та навігації, а не на публікації текстів. Оскільки самі розділи тут не розміщуються, розробники зробили головний акцент на інформативності сторінки твору. Читач одразу бачить детальну інформацію про книжку і може в один клік перейти за посиланням на зовнішній вебресурс для читання. Користувачі обговорюють твори в коментарях, а популярність творів легко відстежити через відкриту систему метрик, кількість переглядів, загальний рейтинг та частість додавання в обране.

Таблиця 1.2.

Аналіз переваг та недоліків вебплатформи «UAuthor»

Переваги	Недоліки
Зручний формат збирача інформації з детальними інформаційними картками творів.	Обмежені налаштування відображення тексту (шрифти, фони).
Швидка навігація та наявність прямих посилань на зовнішні сайти, де розміщено сам переклад.	Слабка система пошуку серед великих масивів даних.
Активна соціальна складова, що дозволяє формувати незалежні рейтинги через оцінки та коментарі.	Відсутність синхронізації прогресу між пристроями.

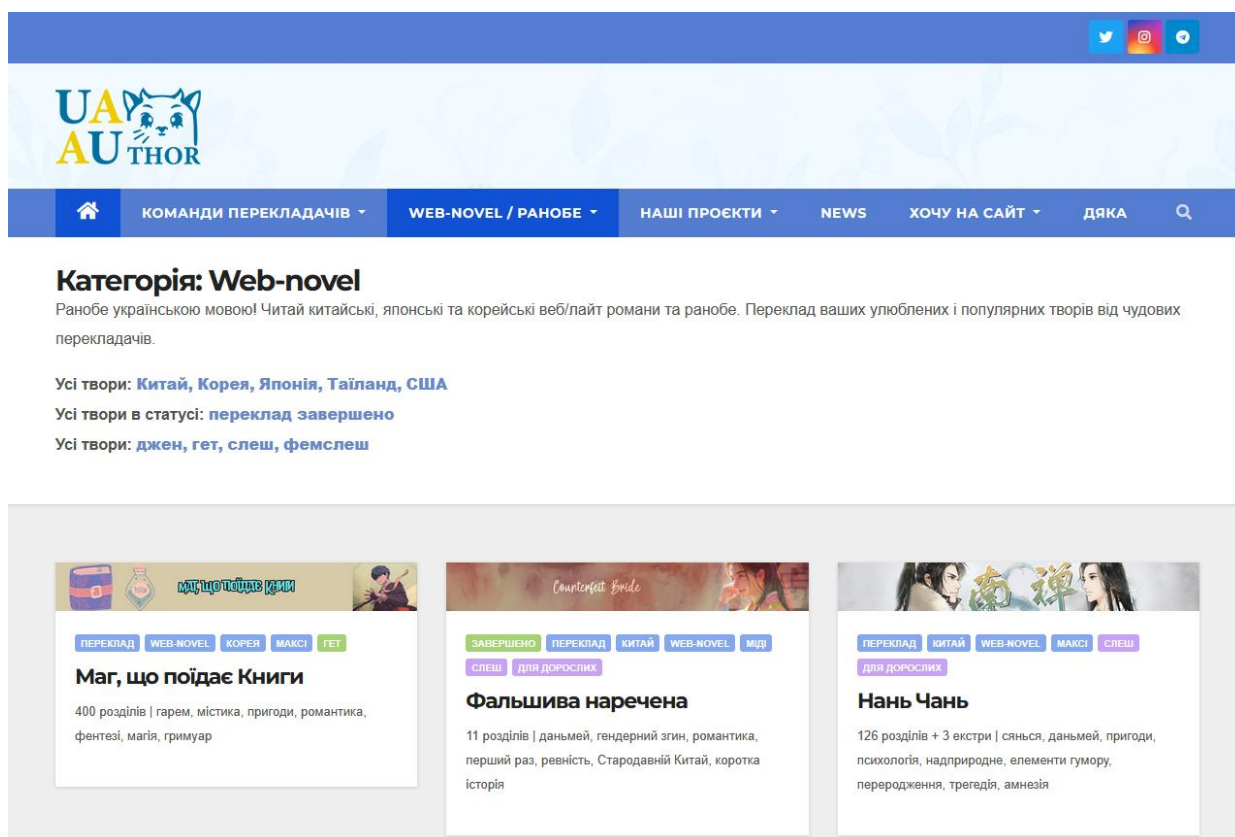


Рис.1.2. Інтерфейс сайту UAauthor

Вебпортал «hikka» (рис. 1.3) працює не просто як база даних, а як соціальна мережа для читачів. Розробники визначили пріоритетом саме активне спілкування аудиторії. Вебсервіс вміє збирати інформацію з багатьох різних джерел, тому сторінка кожного твору виходить інформативною. Також кожен читач має розгорнутий особистий профіль, де можна стежити за оновленнями друзів, гортати стрічку подій та брати участь у живих дискусіях.

Візуальний стиль «hikka» виглядає сучасно, а інтерфейс активно використовує медіаелементи для вигляду творів. Окрема перевага - це продумана темна тема, завдяки якій сайтом комфортно користуватися у вечірній час. Окрім стандартного додавання ранобе до списків, користувачі мають багато корисних функцій. Наприклад, тут можна налаштувати миттєві сповіщення про появу нових розділів, підключити синхронізацію зі сторонніми ресурсами чи вивчати детальну статистику своєї читачької активності.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 1.3.

Аналіз переваг та недоліків вебплатформи «hikka»

Переваги	Недоліки
Розвинена соціальна складова (можливість додавати друзів, стрічка активності та зручні коментарі).	Перевантаженість інтерфейсу. Через велику кількість функцій дизайн іноді здається занадто складним.
Детальна система тегів та фільтрів, що значно полегшує пошук потрібних творів у каталозі.	Брак інструментів для команд. Перекладачам не вистачає зручних рішень для спільної роботи над релізами.
Персоналізована стрічка новин, яка автоматично підлаштовується під вподобання читача.	Високий поріг входження. Новачкам часто буває важко одразу розібратися з навігацією на сайті.
Надійна система сповіщень, яка вчасно нагадує про вихід нових розділів для збережених творів.	Технічні збої. Під час синхронізації дуже великих списків іноді трапляються системні помилки.

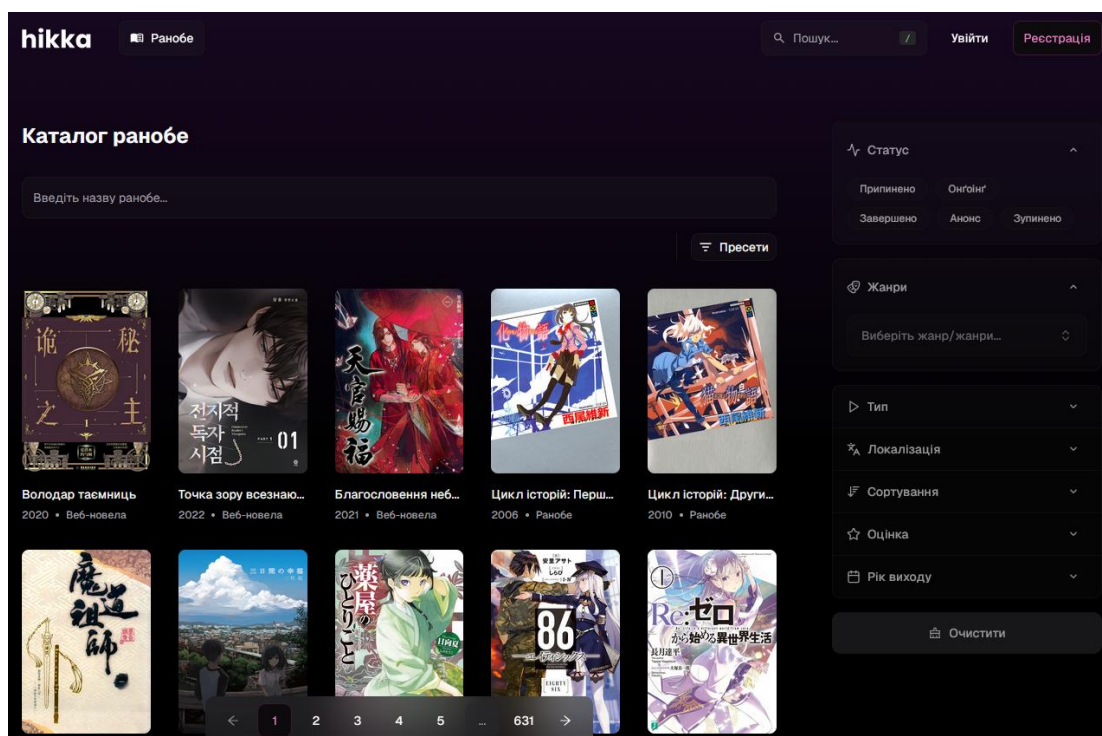


Рис.1.3. Інтерфейс сайту hikka

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

На основі виконаного аналізу існуючих програмних рішень було укладено порівняльну таблицю ключових характеристик досліджуваних вебплатформ: Бака, UAuthor та hikka.

Таблиця 1.4.

Порівняльна характеристика досліджуваних вебплатформ

Характеристика	Бака	UAuthor	hikka	SekaiLib
Тип вебплатформи	Інформаційна база та каталог	вебплатформа для самопублікації	Соціальний медіапортал	Модульна екосистема для читання
Вбудована система читання	Так	Ні	Обмежено	Так
Система оцінювання	Рейтинг (1-10)	Зіркова система	Зіркова система та статуси	Рейтинг за рецензіями
Формування списків	Так	Ні	Так	Так (персоналізовано)
Соціальна взаємодія	Середня	Середня	Висока	Висока
Персоналізовані рекомендації	Обмежено	Ні	Частково	Частково
Інструменти для авторів/перекладачів	Розширені	Відсутні	Обмежені	Розширені
Синхронізація прогресу читання	Так	Ні	Ні	Так
Модерація контенту	Централізована	Авторська та адмінська	Спільнотою та адмінська	Модерська та адмінська
Інтерфейс користувача	Класичний каталог	Спрощений, текстоцентричний	Сучасний, медіанасичений	Сучасний

Характеристика	Бака	UAuthor	hikka	SekaiLib
Архітектура системи	Монолітна	Монолітна	Закрита	Модульний моноліт
Орієнтація на спільноту	Середня	Низька	Висока	Висока

Якщо підсумувати дані з таблиці 1.4, стає зрозуміло, що українським вебплатформам бракує комплексного підходу. Проаналізовані ресурси зазвичай обирають один підхід: вони працюють або як інформаційні каталоги, або зосереджують увагу виключно на спілкуванні читачів. При цьому розробники не приділяють увагу зручному інтерфейсу для самого процесу читання, а також про створення корисних інструментів для команд перекладачів.

Розроблена вебплатформа SekaiLib, створюється саме для того, щоб закрити ці мінуси на ринку. В вебплатформі об'єднується зручна адаптивна вебплатформа для читання, сучасну систему персоналізації та спеціальні інструменти для команд перекладачів. Технічною базою для вебсервісу було обрано архітектуру модульного моноліту. У поєднанні з хмарною синхронізацією прогресу, такий підхід робить вебплатформу корисною для всіх учасників процесу. Тому розробка вебплатформи є цілком виправданою, в результаті виходить швидкий і конкурентний продукт, який легко підлаштовується під кожного користувача.

1.2. Постановка завдання вебплатформи

У межах випускної кваліфікаційної роботи бакалавра потрібно дослідити сферу вебплатформ для читання електронних книжок. Головна мета цього етапу сформулювати завдання для майбутньої програми. Визначаються цілі вебплатформи, його базовий функціонал, а також необхідність зрозуміти потреби майбутньої аудиторії. Оскільки на таких сайтах зазвичай взаємодіють кілька різних груп людей (читачі, перекладачі та адміністратори), важливо зробити цю взаємодію комфортною. На сьогоднішній день більшість процесів

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		16

- від пошуку потрібної книги до завантаження нових сторінок - часто працюють на застарілих технологіях, тому потрібно розробляти програму на сучасних інновацій.

Враховуючи вищезазначене, головне завдання випускної кваліфікаційної роботи бакалавра зводиться до правильного проєктування архітектури [26]. Було обрано підхід модульного моноліту. Це рішення дозволяє логічно розділити систему на незалежні домени (наприклад, управління користувачами, каталог творів та модуль читання). Наступний важливий крок - організація централізованої бази даних. Потрібно спроектувати структуру, яка б ефективно зберігала самі тексти, розбиті на розділи, та надійно зв'язувала їх із профілями користувачів. Також клієнтська частина має бути повністю адаптивною, щоб процес читання був комфортним як на смартфонах, так і на широких моніторах. Ще необхідно реалізувати персоналізацію користувачів. Йдеться про створення власних списків для читання та механізмів, які автоматично запам'ятовують, на якій сторінці зупинився читач.

Окрім зберігання та відображення тексту, окремим завданням є проєктування системи управління ролями та доступом. Тому вебплатформа передбачає взаємодію різних категорій користувачів, необхідно розмежувати їхні права та можливості. Звичайні читачі повинні мати доступ до каталогу, персональних списків та функцій чату та форуму. Натомість для команд перекладачів необхідно розробити панель управління, яка дозволить зручно завантажувати нові розділи, та керувати складом власної команди. Для адміністраторів та модераторів слід передбачити панель управління контентом для спостереження порушень та контролю якості.

Ще одним важливим завданням, яке впливає на утримання аудиторії на сайті, є соціальна частина вебплатформи. Сучасний користувач очікує не лише можливості користуватися матеріалами, але й обговорювати їх з іншими читачами. Тому технічне завдання має включати реалізацію багаторівневої системи коментарів, механізмів оцінювання.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

Через те, що вебплатформи для читання часто потрапляють у ситуацію з проблемою сильного навантаження, важливим завданням є забезпечення швидкодії програми. Для цього необхідно налаштувати багаторівневе кешування. Це дозволить значно зменшити навантаження на реляційну базу даних під час запитів до сторінок каталогу. Крім того, важливо спроектувати швидку систему пошуку з розширеною фільтрацією за жанрами, форматами та статусом перекладу.

Особливу увагу при складанні завдань слід приділити самому модулю читання, який є основою системи. Завдання полягає не лише у виведенні тексту на екран пристрою, а й у наданні користувачеві інструментів для глибокої кастомізації, зміни розміру шрифту, перемикання між фонами (світла, темна, сепія).

Через те що є необхідність підтримки роботи команд перекладачів та забезпечення загальної життєздатності вебплатформи, окремим завданням є додавання оплати. Розробка системи фінансових операцій відкриє доступ для реалізації функціоналу "раннього" доступу до нових розділів. Це вимагає створення надійної логіки обробки транзакцій, синхронізації статусів покупок.

Не менш важливим моментом роботи сучасної системи є забезпечення миттєвої комунікації з користувачами, для вебплатформи, де постійно публікуються нові розділи, оновлюються переклади та ведеться активне обговорення, необхідною є реалізація системи сповіщень у реальному часі. Використання сучасних вебпротоколів та технологій двостороннього зв'язку дозволить інформувати читачів про вихід нових розділів улюблених творів, відповіді на їхні коментарі або важливі системні події без необхідності перезавантаження сторінки.

У результаті можна сказати, розробка сучасної вебплатформи для електронних книг охоплює проєктування вірної бекенд-архітектури, створення нормалізованої бази даних, гарантування безпеки даних, додавання механік взаємодії між користувачами та розробку високоякісного клієнтського інтерфейсу. Вирішення цих завдань дозволить досягти глобальної мети -

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

створити продукт, що повністю задовольняє актуальні потреби сучасних читачів та авторів контенту.

1.3. Обґрунтування вибору архітектури системи

Вибір архітектури для SekaiLib став чи не найважливішим кроком на старті. Від цього рішення залежить, як система витримає раптовий наплив користувачів і наскільки складно буде її потім масштабувати. Було перебрано три основні підходи, спочатку розглянуто класичний моноліт, уся програма збирається як один великий блок, тому бізнес-логіка, база даних та інтерфейс міцно з'єднані між собою.

Якщо детальніше розглянути недоліки класичного моноліту в контексті випускної кваліфікаційної роботи, то головною загрозою стає відсутність ізоляції помилок. Наприклад, критичний збій у системі обробки платежів або коментарів може призвести до падіння всього вебсервера. Це означає, що читачі миттєво втратять доступ до самого процесу читання книг, хоча ці функції логічно ніяк не пов'язані. Крім того, при класичному підході неможливо масштабувати лише ту частину, яка дійсно цього потребує.

Далі був розглянутий варіант із мікросервісами [10]. Суть у тому, щоб розбити вебплатформу на купу дрібних незалежних сервісів, які спілкуються між собою по мережі, це дає максимальну гнучкість і стійкість. Але для старту SekaiLib такий підхід виявився занадто перевантаженим, головна проблема ховається у дуже складній інфраструктурі. Якщо постійно оперувати такими масивами даних через мережеві запити, інтерфейс почне помітно гальмувати.

Окрім мережевих затримок, мікросервісний підхід створює проблеми на рівні баз даних. Оскільки кожен сервіс повинен мати власну незалежну базу, виникає проблема розподілених транзакцій. Для збереження цілісності даних довелося б реалізовувати складні патерни. Для вебплатформи на стадії запуску це означає невиправдано високі витрати часу на написання інфраструктурного коду.

Тому було обрано модульний моноліт. Тут зберігається головний плюс класичного підходу, проте сама система всередині суворо поділена на

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		19

ізолювані частини. Ці модулі спілкуються виключно через прописані інтерфейси. У результаті маємо чистий код і можливість спокійно оновлювати окремий функціонал без страху випадково зламати всю програму.

Архітектура модульного моноліту (рис. 1.4) в SekaiLib опирається на принципи предметно-орієнтованого проєктування. Уся система логічно розбивається на так звані "обмежені контексти". Наприклад, виділяються межі між модулями "Каталог", "Читач", "Соціальна взаємодія" та "Команди перекладачів". Кожен із цих модулів має власну бізнес-логіку та використовує свої дані. При цьому, на відміну від мікросервісів, усі вони використовують спільну реляційну базу даних, але звертаються лише до своїх логічних схем або таблиць.

Завдяки такому підходу залишається висока швидкість роботи вебплатформи. Найбільша перевага тут стратегічна, якщо в майбутньому вебплатформа виросте настільки, що мікросервіси стануть необхідністю, не доведеться переписувати весь код з нуля. Ми зможемо просто від'єднати потрібні модулі. Наразі ж така архітектура дає змогу паралельно і незалежно оновлювати систему обробки електронних книжок та соціальні функції [4]. Для вебплатформи SekaiLib це ідеальний варіант, який дозволяє плавно масштабуватися разом зі збільшенням кількості активних читачів.

Мікросервісна архітектура (рис. 1.5) працює по іншому, вся система розбивається на дрібні незалежні сервіси. Кожен сервіс відповідає за свою частину роботи і має власну базу даних. Наприклад, модулі авторизації, каталогу електронних книжок, читання та сповіщень працюють окремо, а спілкуються між собою через загальний шлюз (API Gateway). Головний плюс у тому, що можна посилювати сервери тільки для тих частин, які мають найбільше навантаження. Але ця ж схема показує головну проблему - занадто складну інфраструктуру. Довелося б постійно налаштовувати мережеві запити між сервісами та стежити, щоб дані в різних базах не конфліктували.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

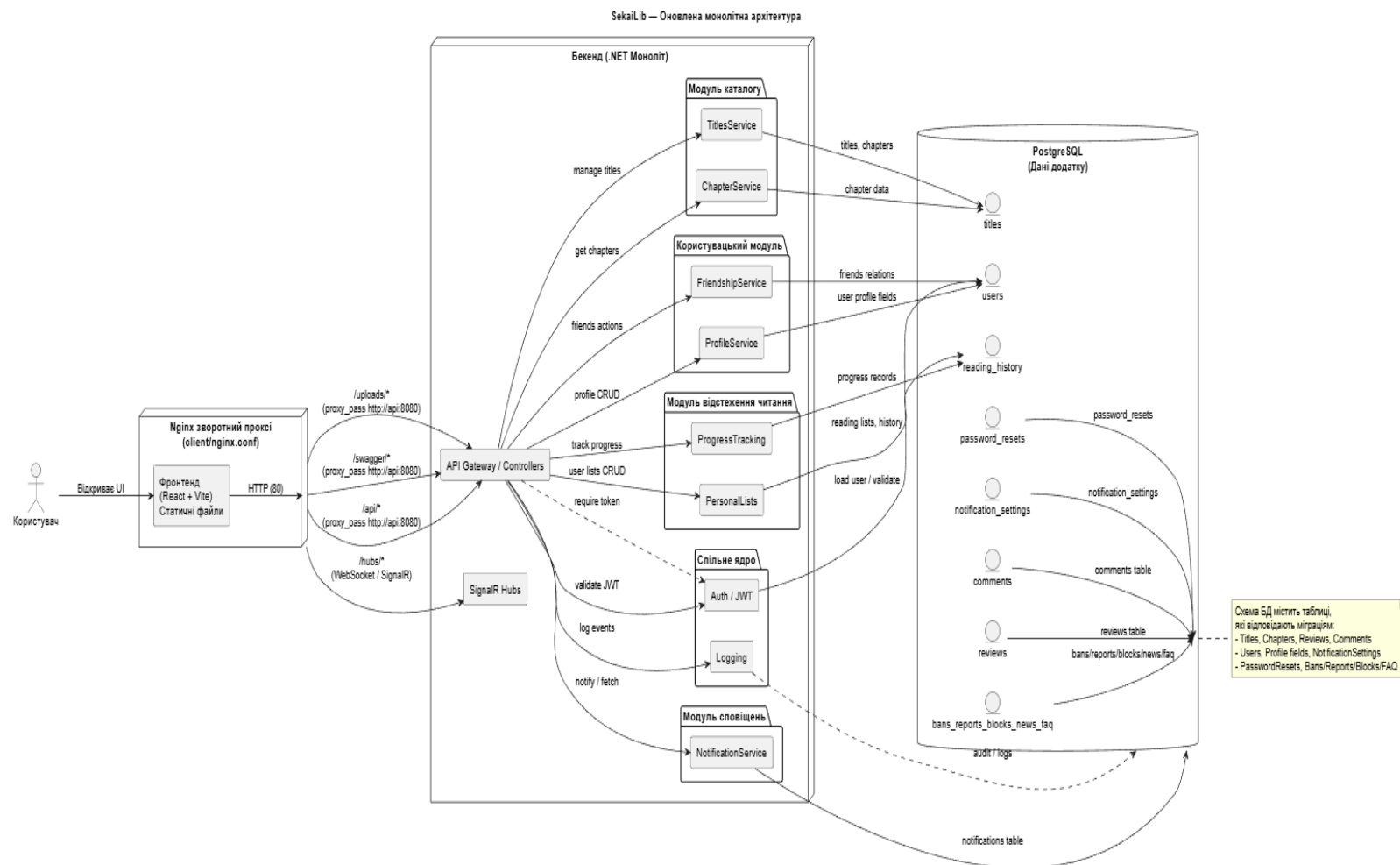


Рис. 1.4. Схематичне зображення монолітної архітектури

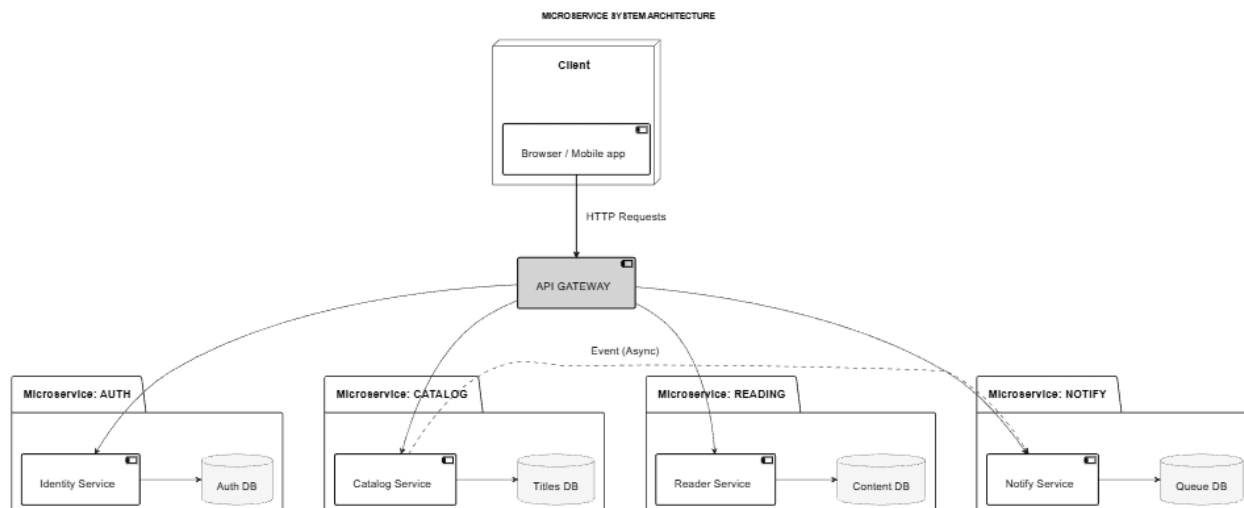


Рис. 1.5. Схематичне зображення мікросервісної архітектури

Щоб реалізувати вебплатформу SekaiLib, необхідно зупинитися на багат шаровій модульній архітектурі. Фактично, це логічне продовження і покращення концепції модульного моноліту. Різниця така, що система не просто ділиться на незалежні функціональні блоки. У середині кожного модуля код додатково структурований за суворими правилами "Clean Architecture" [29]. Це дозволяє підтримувати порядок у вебплатформі та гарантує, що бізнес-логіка не буде переплітатися з базою даних чи зовнішніми інтерфейсами.

На діаграмі (рис. 1.6). детально зображено, як працює ця архітектура у вебплатформі. Головне правило - суворе дотримання принципу слабе зв'язування. Якщо подивитися на схему, видно, що в самому центрі знаходиться ядро системи (Domain Layer). Воно містить базові сутності електронних книжок, бізнес-правила та інтерфейси. Це ядро є повністю автономним. Навколо нього побудований рівень застосунку (Application Layer), де зосереджені всі сервіси (Use Cases), валідація та логіка роботи з DTO. Вся технічна робота винесена на зовнішній рівень (Infrastructure Layer). Саме там реалізовані адаптери для підключення до PostgreSQL через Entity Framework Core [6], робота з файловою системою та кешування через Redis.

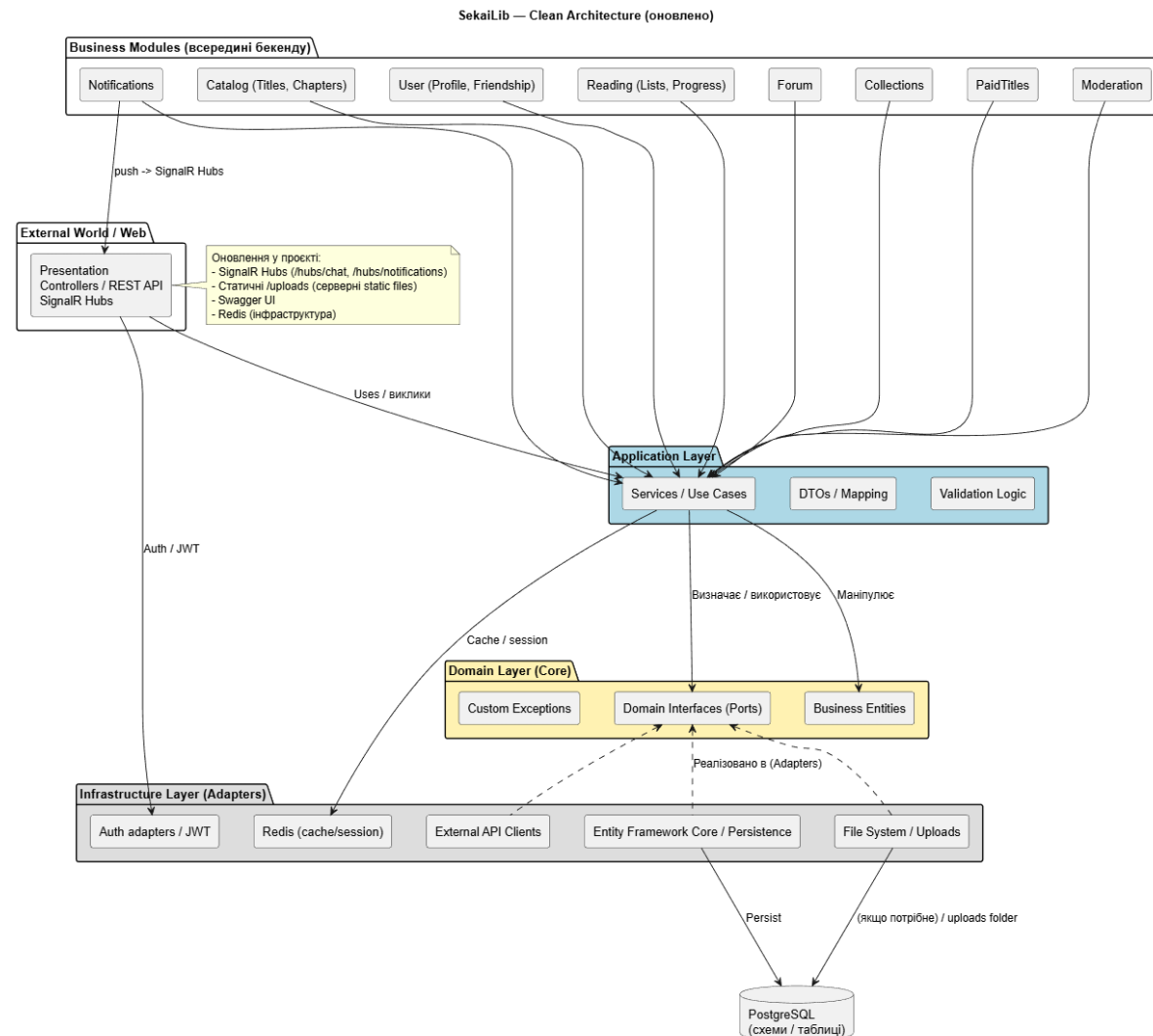


Рис. 1.6. Схематичне зображення багатошарової модульної архітектури

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

Архітектура SekaiLib - це звична взаємодія клієнта та сервера, але з суттєвим доповненням у вигляді модульного моноліту на бекенді [8]. Такий змішаний підхід вирішує одразу кілька проблем. Для початку просто розділяється інтерфейс та обчислення. Клієнтська частина відповідає лише за комфортне читання текстів, а весь важкий процес обробки лягає на сервер. Проте, якщо бекенд був звичайним суцільним кодом, з часом він перестав би справлятися з роботою. Через це він розбитий на незалежні модулі. Коли в каталозі з'являться тисячі нових електронних книжок, а трафік зросте, ця структура дозволить масштабувати систему без важкого рефакторингу.

Для серверної частини було обрано вебплатформу .NET 8 [5]. Весь бекенд із самого початку проєктувався за суворими стандартами Clean Architecture. Тому що хотілося відразу уникнути класичної проблеми заплутаного коду. Кожна частина системи отримала свою зону відповідальності, замість того щоб мішати все в одну купу, серверну логіку обережно розбито на три внутрішні шари.

Для початку розповім про Domain Layer, тут відбувається вся фундаментальна бізнес логіка SekaiLib. Усі базові сутності такі як моделі електронної книги, окремого розділу чи профілю користувача знаходяться саме у ньому. Головна ідея цього рівня полягає в абсолютній ізольованості, жодних посилань на бази даних, ORM чи якісь сторонні бібліотеки. Завдяки цьому ключові правила системи працюють повністю автономно. Навіть якщо колись доведеться глобально оновлювати зовнішні фреймворки, можна не хвилюватися, базові алгоритми вебплатформи не зламаються.

Далі йде Application Layer, Він працює як такий собі контролер усієї системи. Саме тут прописані конкретні сценарії взаємодії. Наприклад, читач перегортає сторінку книги, цей шар одразу перехоплює запит, зв'язується з ядром системи і фіксує прогрес читання. Те саме стосується керування особистими списками творів. При цьому рівень застосунку не знає про

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		24

технічні нюанси роботи з базою даних. Він просто вирішує, що саме треба зробити для виконання запиту.

Наостанок залишається шар Infrastructure Layer, він бере на себе всю монотонну низькорівневу реалізацію бекенду. Саме у ньому налаштовувалося пряме підключення до PostgreSQL, до нього винесена вся взаємодія зі сторонніми інструментами. Налаштування кешування, доступ до файлової системи, сервіси логування помилок.

Візуальна частина SekaiLib побудована за допомогою React та Vite [9, 12]. Інтерфейс вебплатформи є повністю динамічним. Як тільки читачу потрібен новий розділ, система відправляє непомітний API-запит до бекенду. Назад повертається лише чистий текст. Це архітектурне рішення має одну конкретну мету. Серверам більше не потрібно витрачати оперативну пам'ять на генерацію HTML-сторінок. Відображенням інтерфейсу займається виключно пристрій самого користувача, що ідеально економить ресурси хостингу.

Найважливішою характеристикою бекенду стала глибока внутрішня модульність. Щоб не будувати монолітну структуру, були ізолювальовані ключові процеси. Каталог електронних книжок, соціальні активності читачів та механізм відправки сповіщень - це тепер самостійні модулі зі своїм власним життєвим циклом. Якщо з часом трафік на вебплатформі критично збільшиться, не доведеться робити глобальний рефакторинг. Можна буде точково від'єднати найбільш навантажений модуль і перетворити його на незалежний мікросервіс, зберігши при цьому стабільність решти вебсервісу.

1.4. Обґрунтування вибору інструментальних засобів та вимоги апаратного забезпечення

Для вебплатформи управління цифровим контентом та читання електронних книжок потрібно знайти належні технології. Увесь стек розробки був обраний за допомогою трьох факторів, специфікою обробки великих масивів тексту, функціональними вимогами до системи та архітектурою модульного моноліту. Оскільки вебплатформа має відразу підвантажувати

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		25

контент і при цьому залишатися стабільною під час пікових навантажень, нам знадобився такий набір фреймворків і баз даних, який би ідеально працював у такій зв'язці і гарантував надійну ізоляцію внутрішніх модулів.

Для написання серверної логіки було обрано мову C# та екосистемі .NET. Тому, що для вебплатформи з великим обсягом важливо мати максимально надійний код. C# зі своєю строгою типізацією ідеально вирішує цю проблему, зводячи до мінімуму кількість випадкових помилок і збоїв ще на етапі компіляції. В якості основи для нашого Web API було використано фреймворк ASP.NET Core [3, 27]. Це інструмент, який відомий своєю продуктивністю під високими навантаженнями. Крім того, його кросплатформність дає змогу розгортати сервери вебплатформи як на Windows, так і на дешевших Linux-машинах без жодних змін у написаному коді.

У якості бази даних вибрано на PostgreSQL. Для вебплатформи електронних книжок важливо не просто зберігати тексти, а й гарантувати цілісність даних про користувачів та їхній прогрес. PostgreSQL ідеально закриває цю потребу завдяки суворій підтримці ACID-транзакцій. Мається на увазі, якщо система фіксує збереження закладки або додавання коментаря, ми можемо бути впевнені, що ці дані не загубляться навіть при раптовому збої сервера [28]. Крім того, ця СУБД має систему індексування. Це означає, що коли в каталозі SekaiLib з'являться багато нових розділів, пошук потрібної книги все одно працюватиме швидко.

Щоб уникнути проблем з хаосом даних в SekaiLib, щоб не писати ручним способом SQL-код було використано Entity Framework Core. Ця ORM виступає надійним зв'язком між PostgreSQL та бізнес-логікою вебплатформи. У коді використовуються звичні .NET об'єкти, а фреймворк уже сам генерує безпечні запити до сервера [16]. Важливий момент - це система міграцій. Архітектура дозволяє нам швидко змінювати структуру таблиць по мірі зростання вебплатформи, взагалі не ламаючи при цьому ядро системи.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

Крім основного місця для зберігання даних, для досягнення максимальної продуктивності вебплатформи під час високих навантажень до стеку було залучено in-memory базу даних Redis. Її використання дозволяє кешувати найчастіші запити до програми, що суттєво знижує навантаження на реляційну базу PostgreSQL та зменшує час відповіді сервера. Додатково, для реалізації функцій взаємодії між користувачами у режимі реального часу, додано технологію SignalR.

Основною мовою для написання клієнтської частини виступає TypeScript [15]. Таке рішення вирішене необхідністю підтримувати сувору типізацію не тільки на сервері, а й у браузері. Це дозволяє знаходити критичні помилки ще на етапі написання коду. Побудова користувацького інтерфейсу базується на React [14]. Наприклад, картка книги в каталозі існують як окремі, незалежні елементи. React забезпечує миттєвий рендеринг сторінок без повного перезавантаження браузера.

Для покращення розробки клієнтської частини та збирання вебплатформи було обрано Vite. Він забезпечує перезавантаження модулів, що важливо для швидкого проєктування інтерфейсів. Візуальне оформлення вебплатформи виконується за допомогою CSS-фреймворку Tailwind CSS. Його підхід дозволяє створювати дизайн, без написання великих таблиць стилів. Це відіграє головну роль у створенні модуля читання, де користувач може змінювати розмір шрифту та перемикається між кольором фону.

З метою нормалізації середовища та уникнення конфліктів налаштувань на різних етапах розгортання, вебплатформи спирається на технології контейнерів Docker. Усі незалежні компоненти системи перестають залежати один від одного у власних контейнерах, а їхня взаємодія виконується Docker Compose. В якості проксі-сервера використовується NGINX. Він ефективно направляє вхідні HTTP-запити до відповідних модулів моноліту.

Ефективне розгортання та забезпечення безперебійної роботи вебплатформи залежать від апаратних ресурсів сервера та клієнтських пристроїв. Через це були сформовані критерії. Список мінімальних та

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

рекомендованих параметрів для забезпечення стабільного функціонування серверної та клієнтської частин наведена в таблиці 1.5.

Таблиця 1.5.

Вимоги для функціонування програми

Параметр	Мінімальні вимоги	Рекомендовані вимоги
Процесор (CPU)	4 ядра (2.4 GHz)	8 ядер (3.2 GHz+)
Оперативна пам'ять (RAM)	8 ГБ	16 ГБ
Накопичувач (SSD)	50 ГБ вільного місця	100 ГБ+ SSD
Мережевий інтерфейс	50 Мбіт/с	100 Мбіт/с - 1 Гбіт/с
Операційна система	Windows 10/11, Ubuntu 22.04	Windows Server 2022, Debian 12

Набір переліченого стеку формує необхідний технологічний фундамент. Вибір кожної технології був вибраний за специфікою модульного моноліту. Разом вони утворюють середовище, де бекенд, база даних та інтерфейс користувача працюють як єдиний, механізм. Це надає стабільність коду та легкість у підтримці. Вебплатформа SekaiLib має надійний стек, який без проблем адаптується під зростаючі вимоги та нові навантаження.

Висновки до першого розділу

У висновку першого розділу завершується теоретичне проєктування та вирішено напрямок у якому буде виконуватися програма. Головну задачу сформульовано, це розробка спеціалізованої вебплатформи SekaiLib. Аналіз предметної області дозволив виділити ключові функціональні блоки. Система має об'єднати читачів, авторів та перекладачів у зручному цифровому середовищі.

Дослідження існуючих аналогів охопило три вебплатформи: Бака, UAuthor та hikka. Їхній функціонал для поширення текстів детально розібрано та систематизовано у зведеній таблиці. Створення нового продукту є

абсолютно потрібним. SekaiLib поєднає в собі найкращі риси сучасного каталогу та зручні інструменти для читання, яких не вистачає аналогам.

Було проведено порівняльний аналіз архітектурних рішень, за результатами якого для реалізації вебплатформи обрано модульну монолітну архітектуру, підсилену принципами чистої архітектури. Це рішення дозволяє забезпечити стабільність системи на етапі розробки та створює необхідну гнучкість для її подальшого масштабування шляхом потенційного винесення окремих модулів у незалежні мікросервіси.

На завершення першої частини вебплатформи є затвердження стеку. Інструменти для серверного та клієнтського коду було обрано з урахуванням усіх вимог системи. Також сформовано точний перелік апаратного забезпечення, що є умовою для безпечного розгортання та безперервного функціонування вебплатформи.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		29

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБПЛАТФОРМИ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИЖОК

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

Архітектурне проєктування вебплатформи SekaiLib починається з вивчення аудиторії. Різні категорії читачів пропонують різні вимоги до інтерфейсу та бекенду [17]. Цей аналітичний етап розв'язує одразу дві проблеми. Спочатку ставляться рамки до масштабу продукту, залишаючи лише необхідні модулі. Далі інформація перетворюється на технічні метрики.

По функціональним вимогам нашої системи, почну з управління доступом та безпеки. Вхід на вебплатформу побудований на надійній зв'язці JWT та OAuth. За соціальне спілкування відповідають спеціальні алгоритми, щоб у коментарях не було хаосу, було додано систему скарг та механізм блокування порушників. А якщо хтось порушив правила або образив іншого користувача просто, читач завжди може закинути його в "ігнор-лист" і більше ніколи з ним не перетинатися.

Щодо інформаційного каталогу, пошук, зручна навігація, перегляд розділів і детальні теги до кожного твору. Якщо в користувача виникають питання, працює модуль FAQ. Для анонсів технічних робіт чи якихось оновлень виділено окрему сторінку новин. Звісно, додали гнучку фільтрацію та сортування в самому каталозі. Самі команди перекладачів отримали інструменти для управління доступом до своїх релізів. Ну і модерація публікацій та коментарів також присутня.

Тепер про персоналізацію, найважливіший функціонал - це збереження прогресу читання. Алгоритм автоматично фіксує, на якому саме місці користувач зупинився. Щоб не загубитися у творах, користувачі збирають власні списки або створюють тематичні колекції. В особистому кабінеті можна налаштувати профіль під себе. Наприклад, приховати контент із тими

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		30

жанрами, які читату не подобаються. А вигляд самих каталогів миттєво перемикається між сіткою та плиткою без жодних підвисань.

Для команд перекладу було додано, механізм платного раннього доступу до свіжих розділів. Для одного твору можна вибрати улюблену команду перекладу і підписатися на сповіщення виключно про їхні оновлення.

Наостанок розповім про соціальну взаємодію, вебплатформа заточена під постійне спілкування між читачами. Твори оцінюються через повноцінні рецензії. Користувачі активно обмінюються враженнями прямо під розділами, а для більш глобальних дискусій створено повноцінний форум. Там можна спокійно створювати окремі гілки та обговорювати все що завгодно.

Тепер щодо нефункціональних вимог вебплатформи. Було закладено в основу принципи Clean Architecture, щоб отримати чіткий поділ відповідальності між шарами коду. А для зручності розгортання все це обережно поміщено в Docker-контейнери.

З приводу продуктивності, вебсервер зобов'язаний працювати швидко. Відповідь API на будь-які операції читання не має перевищувати 500 мілісекунд. Сама ж головна сторінка повинна повністю відображатися максимум за дві з половиною секунди.

Далі розповім про безпеку та надійність, весь трафік йде виключно через захищений протокол HTTPS. Валідація даних у нас працює у два етапи, спочатку перевіряємо на клієнті, а потім ще раз на сервері. Також було реалізована можливість створювати бекапи бази даних, щоб при збоях не зникали усі дані з вебплатформи.

Наостанок залишається експлуатація та підтримка, будь-яка критична дія в системі одразу фіксується в логах. Реліз нових версій повністю автоматизоване завдяки пайплайнам CI/CD. API зручно задокументоване через Swagger за всіма стандартами OpenAPI [23]. І головне, архітектура дозволяє додавати нові модулі, можна спокійно розширювати вебплатформу, взагалі не хвилюючи випадково зламати вже готовий і працюючий функціонал.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		31

Вимоги до часових характеристик реакції системи на дії користувача наведені у таблиці 2.1.

Таблиця 2.1.

Нормативні часові показники реакції системи

Тип операції	Опис дії	Цільовий час відгуку
Інформаційна	Завантаження тексту розділу / списку тайтлів	< 500 мс
Інтерактивна	Пошук, фільтрація, перемикання сторінок	< 300 мс
Транзакційна	Оновлення прогресу, виставлення оцінки	< 200 мс
Системна	Повна генерація сторінки (LCP)	< 2.5 с

Доступ до вебплатформи (рис. 2.1) виконується через реєстрацію та автентифікацію. Користувач може авторизуватися через форму входу, або виконати вхід через Google. Коли відбувається успішний вхід, людина може спілкуватися з іншими користувачами, купувати платні розділи творів, змінювати профіль та особисті дані. Ще є можливість приховувати жанри, які його не цікавлять, а також формувати список ігнорування користувачів. При читання система зберігає прогрес для кожного окремого твору. Для підтримки користувачів передбачено інформаційні сторінки, що містять новини та FAQ.

Читач налаштовує власну бібліотеку за статусами перекладу творів та формує категорії за різними темами для інших учасників які можуть цим зацікавитися. Вони додають одне одного у друзі та спілкуються через приватні повідомлення, а ще беруть участь у форумах. Також кожна рецензія написана авторизованим користувачем змінює загальний рейтинг твору.

Команди перекладу додають розділи та виконують створення творів. Вони ставлять тимчасове обмеження на платні розділи у творах та взаємодіють

з аудиторією через систему сповіщень, де читач сам вибирає за якою командою буде слідкувати для отримання повідомлень від неї.

Модератори та адміністратори керуєть SekaiLib за допомогою свого доступного функціоналу. Модератор перевіряє нові твори чи не порушують правила сайту, відстежує активність людей на форумі та обробляє скарги щодо порушень правил. Адміністратор може додавати жанри до сайту для більшої різноманітності, блокує акаунти порушників.

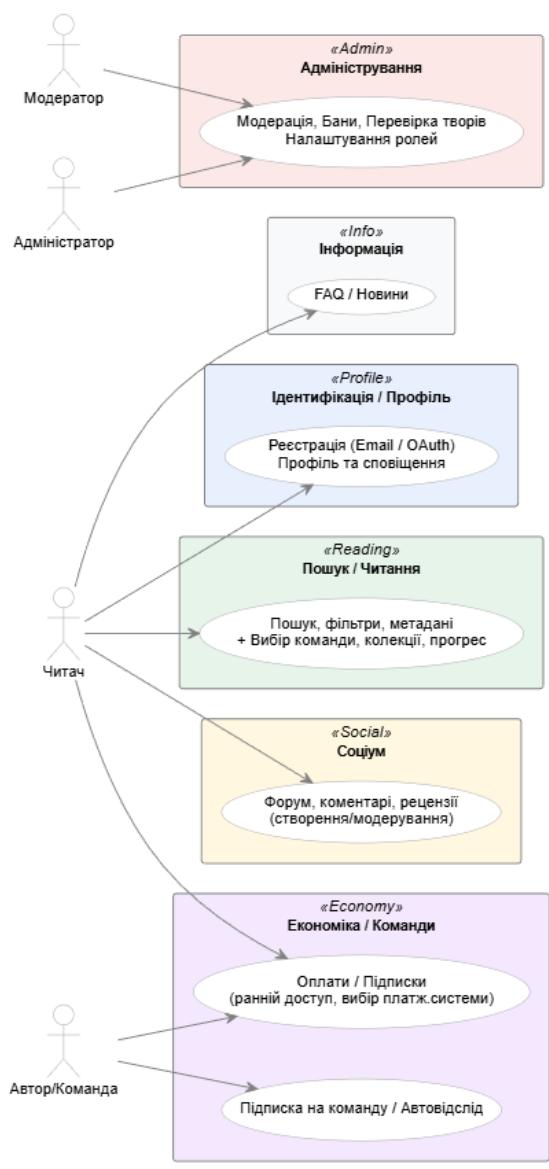


Рис. 2.1. Діаграма варіантів використання системи

Діаграма класів доменного рівня (рис. 2.2) демонструє об’єктну модель вебплатформи та є основою для подальшої розробки схеми бази даних у межах архітектури модульного моноліту.

Об'єктний підхід означає, що за різні частини програми відповідають саме ці об'єкти програми. Це допомагає ставити межі між сферою відповідальності в коді. Головну роль тут має клас User. Він не просто зберігає потрібні дані та поєднує компоненти між собою такі як, токени автентифікації користувачів вебплатформи.

Через клас Title працює вся інформаційна модель програми, в ній розміщені всі основні дані про твори, та він взаємодіє з класом Chapter, який показує розділи та керує функцією платних розділів. Також, щоб не плутатися в класифікації, є сутності Genre та TitleGenre, а за викладення розділів та творів відповідає клас TranslationTeam. Ще є окрема таблиця TranslationTeamMember, яка впливає на ролі людей у команді перекладу.

Читачі працюють із інформацією та змінюють власну бібліотеку через списки у своєму профілі. За допомогою класу UserList користувачі створюють власні списки для перегляду у профілі, та оглядають всі твори які у них є, тоді як ReadingList керує звичайними списками не тими які користувач створив сам. Також працює сутність UserReadingProgress, вона зберігає номер останнього прочитаного розділу в конкретному творі та робить відсоткову візуалізацію саме того розділу на головній сторінці у блоці «Продовжити читати».

Соціальний блок SekaiLib та система зворотного зв'язку мають вигляд ієрархії класів для рецензій та коментарів. Клас Review дозволяє писати коментарі з оцінками, а сутність ReviewComment надає можливість писати відповіді на інші коментарі. Така сама структура використана для коментарів під відповідним розділом. Взаємодія з цим покращується класом Reaction, що дозволяє користувачам ставити вподобайки або не вподобайки на коментарі та рецензії. Комунікація між користувачами працює через запити у друзі, та систему особистих повідомлень, що включає сутності Conversation, ConversationParticipant та Message.

Рівень керуваннями системою реалізований класами Moderator та Administrator, які мають доступ до керування користувачами та інформацією

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

про твори та розділи. Система сповіщень виконана через клас Notification, який реагує на різні модулі системи та відправляє дані для користувача у дзвінок.

Коли проєктувалася доменна модель увага була приділена ієрархії взаємозв'язків, що забезпечують коректну роботу бізнес-процесів. Зокрема, сутність TitleRating виступає зв'язком між читачем та твором, щоб можна було ставити рейтинг твору та відстежувати його зміни.

Наведена на рисунку структура показує високий рівень модульності, кожна підсистема має власні визначені межі, але додана в загальну екосистему через сутність User. Це гарантує, що розробка схеми фізичного рівня бази даних у наступних підрозділах буде максимально точно відповідати вимогам доменної області вебплатформи SekaiLib.

Реалізація діаграми доменної моделі в програмному коді здійснюється за допомогою технології ORM, зокрема Entity Framework Core. Завдяки цьому кожен представлений клас буде показано у відповідних таблицях реляційної бази даних PostgreSQL. Усі закладені на етапі проєктування зв'язки між сутностями змінюються у систему зовнішніх ключів, що гарантує цілісність даних. Такий підхід дозволяє відокремити бізнес логіку від безпосередньої роботи зі сховищем, спрощує структури таблиць через міграції та формує надійний рівень доступу до даних, необхідний для роботи вебплатформи.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		35

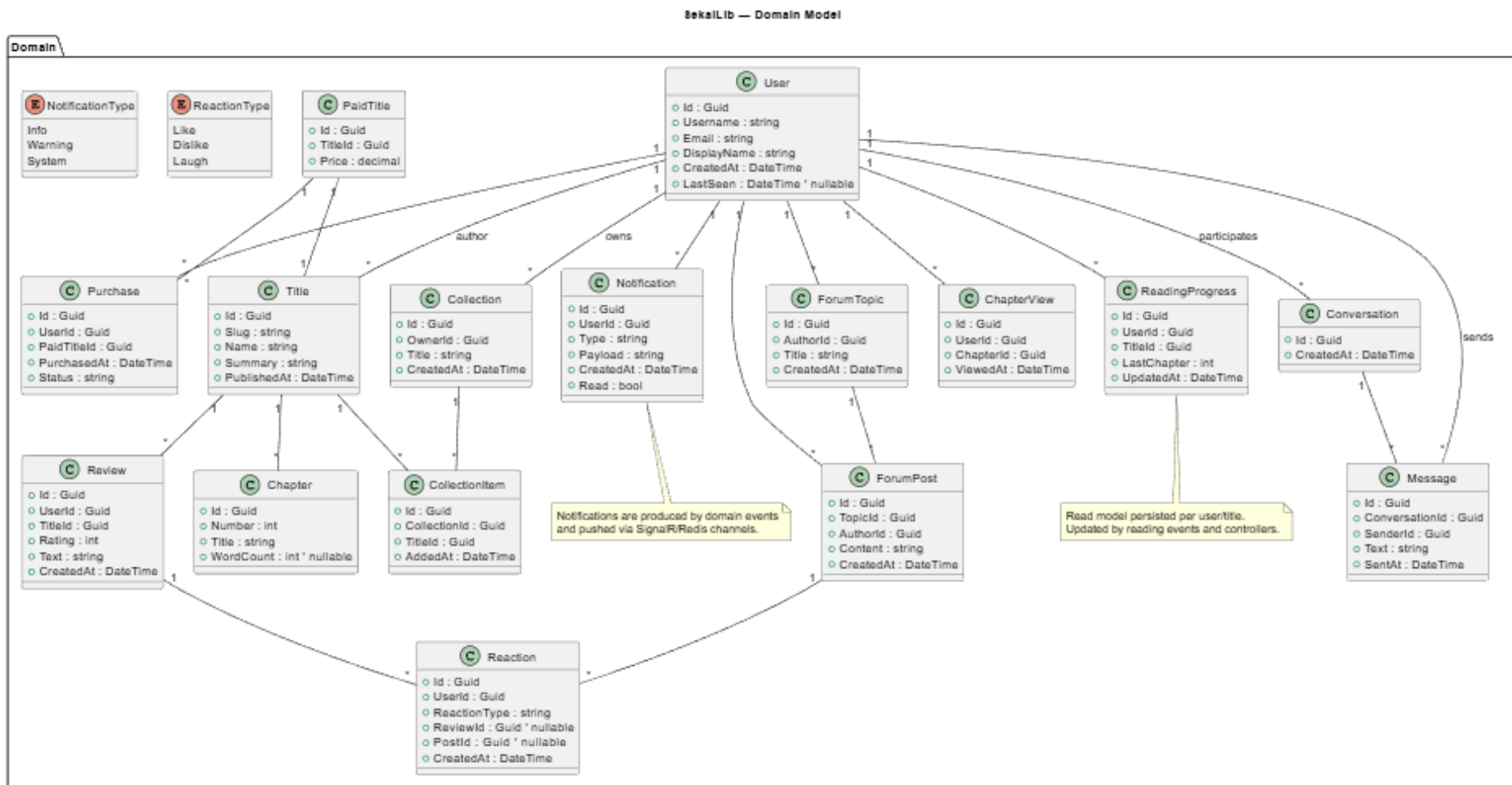


Рис. 2.2. Діаграма класів доменних моделей

2.2. Розробка бази даних системи

Надійне збереження та якісну обробку для даних гарантує реляційна база даних у СУБД PostgreSQL. Її архітектура фокусується на стандартизацію даних та підтримує цілісність інформації про усі модулі та функціонал сайту.

Ядром фрагмента бази даних (рис. 2.3) є таблиця Users. Вона збирає в єдиному місці головну інформацію профілю та його налаштування. Модель безпеки працює через сутності RefreshTokens, PasswordResetTokens та UserExternalLogins, які гарантують надійну авторизацію та безпечне відновлення доступу у випадку проблемних ситуацій. За вибір доступу звичайних користувачів відповідає поле Role в таблиці користувача [11].

Наступне про що слід розповісти - це інструменти модерації та інформування, які виділені в окремі сутності. Таблиця Reports знаходить скарги на контент та зберігає історія перегляду адміністрацією. Для контролю за порушниками працює таблиця UserWarnings, яка реєструє всі видані попередження. Своєю чергою публікацію загальних анонсів для аудиторії контролює таблиця News.

Останнє на цьому зображенні - це соціальна взаємодія, яку організовує пов'язана група таблиць. Для встановлення контактів між авторизавними користувачами існує дві сутності FriendRequests та Friendships, як працюють з запитами та переглядають вже підтверджені дружні зв'язки. Особисту комунікацію у чатах вибудовує трійка сутностей Conversations, ConversationParticipants та Messages. Вони спільно керують створенням діалогів, переліком співрозмовників. Про усі поточні системні події зв'язані з користувачем та сповіщеннями повідомляє таблиця Notifications.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

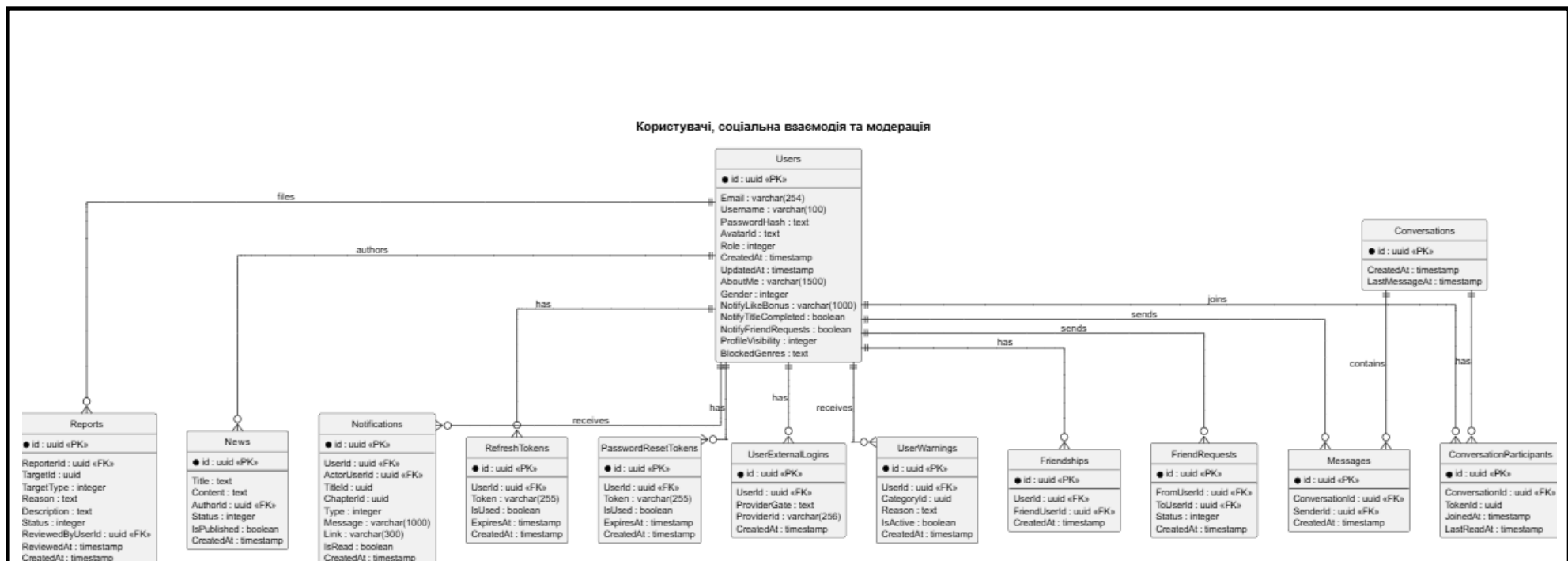


Рис. 2.3. Структура таблиць бази даних для користувача, соціальної взаємодії та модерації

Твори, читання та оплата

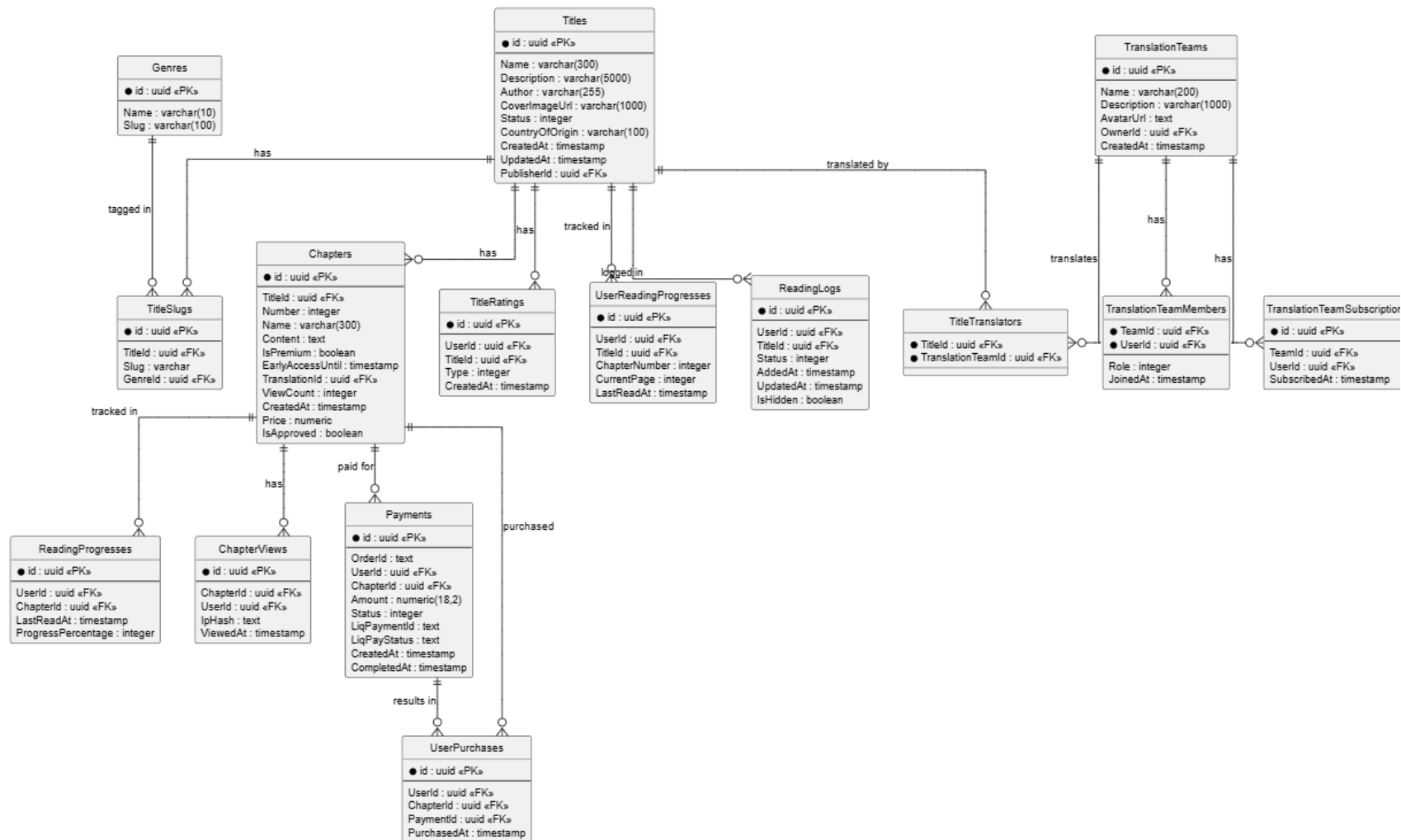


Рис. 2.4. Структура таблиц бази даних для творів, читання та оплати

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		39

Головним елементом цієї діаграми (рис. 2.4) бази даних є таблиця Titles. Вона зберігає інформацію про кожен твір, враховуючи назву, опис, автора та статус усіх творів. А у сутності Chapters знаходиться тексти до розділів, але таблиця не лише зберігає тексти розділів у творах, а й керує монетизацію через параметри платного доступу до них через таймери раннього доступу та встановлення ціни на кожний розділ окремо. Систему навігації по всьому цьому функціоналу та пошуку формують таблиці Genres та TitleSlugs.

Процес перекладу творів контролює декілька таблиць для їхньої коректної роботи. Для початку таблиця TranslationTeams реєструє команди перекладачів, а таблиця TitleTranslators закріплює за ними саме ті твори, які команда переводить. Внутрішню структуру ролей їхніх учасників ухвалює таблиця TranslationTeamMembers. Додатково читачі дізнаються про вихід нових розділів від команд через механізм підписок у таблиці TranslationTeamSubscriptions.

Також присутній окремий блок таблиць, який знаходить статистику. Таблиці UserReadingProgresses та ReadingProgresses фіксують прогрес читання, запам'ятовуючи останній відкритий рядок та розділ. Статистику переглядів кожного окремого розділу рахує сутність ChapterViews коли користувач пробув на розділі як мінімум 5 секунд, тоді як таблиця TitleRatings збирає оцінки від аудиторії для формування рейтингу твору.

Наостанок потрібно розповісти про економічний та фінансовий функціонал вебплатформи, який спирається на дві ключові таблиці. Payments обробляє всі транзакції та фіксує статуси платежів через Liqpay. Після успішного підтвердження оплати створюється запис у таблиці UserPurchases, яка надає придбаний розділ за відповідному користувачу.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		40

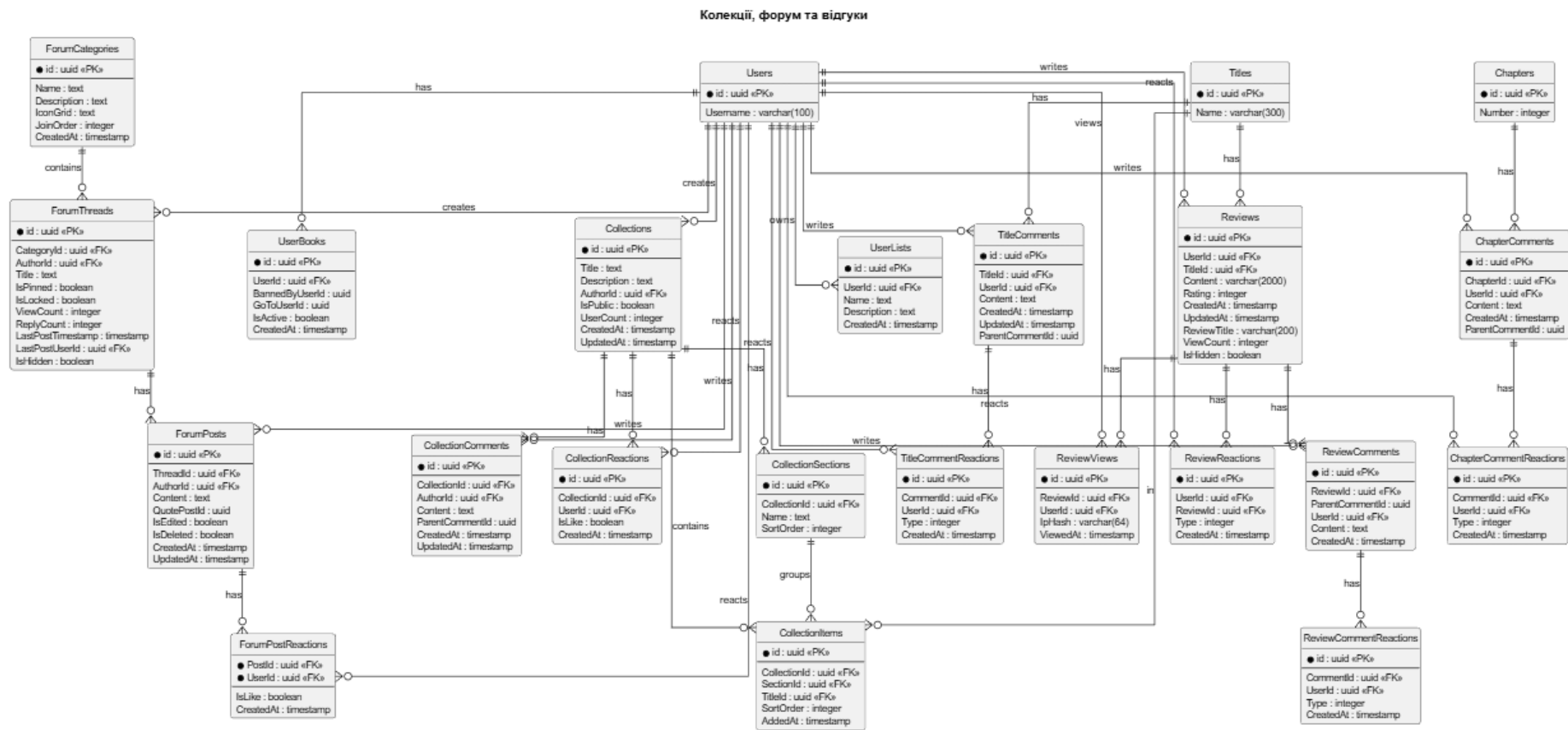


Рис. 2.5. Структура таблиц бази даних для колекцій, форуму та відгуків

Ця діаграма бази даних (рис. 2.5) відповідає за розвиток спільноти вебплатформи та генерацію користувацьких матеріалів. Читачі створюють публічні тематичні колекції через таблицю з такою же назвою, ця сутність дозволяє зручно групувати матеріали за допомогою підрозділів у CollectionSections, куди таблиця CollectionItems передає вибрані книжки. Інші учасники вебплатформи залишають свої думки у CollectionComments та ставлять вподобайки через CollectionReactions. Окремо від публічних колекцій працює таблиця UserLists, за допомогою якої читачі ведуть власні списки, які створені тільки для себе.

Також для масштабних дискусій між користувачами SekaiLib система форумний модуль. Таблиця ForumCategories задає глобальні тематичні напрямки для спілкування. Всередині цих категорій користувачі створюють нові теми через сутність ForumThreads. Усі відповіді учасників на форумі зберігає таблиця ForumPosts, а механізм швидких оцінок д повідомлення обробляє ForumPostReactions. По своєму бажанню читачі можуть додавати небажаних співрозмовників у ігнор-листи через таблицю UserBlocks.

Зворотний зв'язок між користувачами працює через систему рецензій та коментування. Таблиця Reviews використовує повноцінні розгорнуті відгуки на твори, виводячи текст та підсумкову оцінку. Алгоритми сайту додатково відстежують популярність кожної такої публікації через ReviewViews та зберігають реакції аудиторії у ReviewReactions. Користувачі обговорюють авторські рецензії за допомогою таблиці ReviewComments. Також сутність TitleComments збирає загальні враження від твору, тоді як ChapterComments фокусується на обговоренні кожного окремого розділу у кожній книжці. Кожен рівень цього спілкування підтримує власну систему вподобайок через відповідні таблиці TitleCommentReactions, ChapterCommentReactions та ReviewCommentReactions.

Для більшої інформації опишу основні таблиці з скріншотів

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				42
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.2

Опис таблиці Users

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор користувача.
Username	VARCHAR(100)			Нікнейм для відображення на вебплатформі.
Email	VARCHAR(254)			Електронна адреса для авторизації.
PasswordHash	TEXT			Захешований пароль.
AvatarId	TEXT			Посилання на зображення профілю.
Role	INT			Роль користувача
CreatedAt	TIMESTAMP			Час створення облікового запису.
UpdatedAt	TIMESTAMP			Час останнього оновлення даних профілю
AboutMe	VARCHAR(1500)			Текстова інформація про користувача.
Gender	INTEGER			Стать користувача
NotifyLikeBonus	VARCHAR(1000)			Налаштування сповіщень про бонуси та вподобайки.
NotifyTitleCompleted	BOOLEAN			Прапорець сповіщення про завершення твору.
NotifyFriendRequests	BOOLEAN			Прапорець сповіщення про нові заявки в друзі.
ProfileVisibility	INTEGER			Рівень видимості профілю для інших.
BlockedGenres	TEXT			Перелік прихованих жанрів.

Таблиця 2.3

Опис таблиці Reports

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Ідентифікатор скарги.
ReporterId	UUID		+	Користувач, який ініціював скаргу.
TargetId	UUID			Об'єкт, на який надійшла скарга.
TargetType	INTEGER			Тип об'єкта
Description	TEXT			Детальний опис порушення.
Reason	TEXT			Коротко сформульована причина порушення.
Status	INTEGER			Поточний етап розгляду адміністрацією.
ReviewedByUserId	UUID		+	Модератор, який обробив цей запит.
ReviewedAt	TIMESTAMP			Час закриття скарги.
CreatedAt	TIMESTAMP			Час подання скарги.

Таблиця 2.4

Опис таблиці Notifications

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор сповіщення.
UserId	UUID		+	Отримувач повідомлення.
ActorUserId	UUID		+	Користувач, який ініціював подію.
ChapterId	UUID			Пов'язаний розділ
Type	INTEGER			Категорія системного сповіщення.
Message	VARCHAR(1000)			Текстовий вміст.
Link	VARCHAR(1000)			Пряме посилання на подію.
IsRead	BOOLEAN			Статус прочитання користувачем.
CreatedAt	TIMESTAMP			Час генерації сповіщення.

Таблиця 2.5

Опис таблиці Conversations

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор кімнати чату.
CreatedAt	TIMESTAMP			Час створення діалогу.
LastMessageAt	TIMESTAMP			Час відправлення останнього повідомлення.

Таблиця 2.6

Опис таблиці Messages

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор повідомлення.
ConversationId	UUID		+	Посилання на спільну кімнату чату.
SenderId	UUID		+	Автор повідомлення
Content	TEXT			Текст надісланого повідомлення.
CreatedAt	TIMESTAMP			Точний час відправлення тексту.

Таблиця 2.7

Опис таблиці Titles

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор твору.
Name	VARCHAR(300)		+	Офіційна назва твору.
Description	VARCHAR(5000)			Розгорнутий текстовий опис сюжету.
Author	VARCHAR(255)			Автор оригінального твору.
CoverImageUrl	VARCHAR(1000)			Посилання на зображення обкладинки.
Status	INTEGER			Поточний стан випуску
CountryOfOrigin	VARCHAR(100)			Країна походження оригіналу.

продовж.табл.2.7

Назва	Тип даних	PK	FK	Опис
CreatedAt	TIMESTAMP		+	Час створення картки твору.
UpdatedAt	TIMESTAMP			Час останнього оновлення інформації.
PublisherId	UUID		+	Користувач, який додав твір на вебплатформу.

Таблиця 2.8

Опис таблиці Chapters

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор розділу.
TitleId	UUID		+	Посилання на твір.
Number	INTEGER			Порядковий номер розділу.
Name	VARCHAR(300)			Заголовок розділу
Content	TEXT			Безпосереднє текстове наповнення розділу.
IsPremium	BOOLEAN			Ознака платного контенту.
TranslationId	UUID		+	Команда, яка виконала переклад.
EarlyAccessUntil	TIMESTAMP			Дата завершення періоду платного раннього доступу.
ViewCount	INTEGER			Загальна кількість переглядів тексту.
CreatedAt	TIMESTAMP			Час публікації розділу.
Price	NUMERIC			Вартість розблокування контенту.
IsApproved	BOOLEAN			Статус проходження модерації.

Таблиця 2.9

Опис таблиці TranslationTeams

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор команди.
Name	VARCHAR(200)			Назва групи перекладачів.
Description	VARCHAR(1000)		+	Інформація про діяльність команди.
AvatarUrl	TEXT			Посилання на логотип спільноти.
OwnerId	UUID		+	Керівник та засновник команди перекладу.
CreatedAt	TIMESTAMP			Час реєстрації команди на сайті.

Таблиця 2.10

Опис таблиці Payments

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор транзакції.
OrderId	TEXT			Внутрішній номер замовлення в системі.
UserId	UUID		+	Користувач-платник
ChapterId	UUID		+	Розділ, за який здійснюється оплата.
Amount	NUMERIC(18,2)			Сума грошового переказу.
Status	INTEGER			Внутрішній статус обробки платежу.
LiqPaymentId	TEXT			Ідентифікатор операції у платіжному шлюзі LiqPay.
LiqPayStatus	TEXT			Статус транзакції, отриманий від LiqPay.
CreatedAt	TIMESTAMP			Час ініціалізації процесу оплати.

продовж.табл.2.10

Назва	Тип даних	PK	FK	Опис
CompletedAt	TIMESTAMP			Час успішного зарахування коштів.

Таблиця 2.11

Опис таблиці UserPurchases

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Ідентифікатор факту покупки.
UserId	UUID		+	Власник купленого контенту.
ChapterId	UUID		+	Розблокований платний розділ.
PaymentId	UUID		+	Посилання на успішну транзакцію.
PurchasedAt	TIMESTAMP			Точний час надання доступу до тексту.

Таблиця 2.12

Опис таблиці Collections

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор тематичної колекції.
Title	TEXT			Назва публічної колекції.
Description	TEXT			Розгорнутий опис тематики колекції.
AuthorId	UUID		+	Користувач, який створив цю колекцію.
IsPublic	BOOLEAN		+	Прапорець публічного доступу для інших читачів.
UserCount	INTEGER			Статистика кількості користувачів, які зберегли добірку.
CreatedAt	TIMESTAMP			Час створення запису.

продовж.табл.2.12

Назва	Тип даних	PK	FK	Опис
UpdatedAt	TIMESTAMP			Час останнього редагування інформації.

Таблица 2.13

Опис таблиці Reviews

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор рецензії.
UserId	UUID		+	Автор опублікованого відгуку.
TitleId	UUID		+	Твір, на який написано цю рецензію.
Content	VARCHAR(2000)		+	Основний текст рецензії.
Rating	INTEGER		+	Підсумкова оцінка твору від читача.
CreatedAt	TIMESTAMP			Час первинної публікації.
UpdatedAt	TIMESTAMP			Час внесення змін до тексту.
ViewCount	INTEGER			Загальна кількість переглядів рецензії.
IsHidden	BOOLEAN			Прапорець приховання матеріалу модераторами

Таблица 2.14

Опис таблиці ForumThreads

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор теми на форумі.
CategoryId	UUID		+	Розділ форуму, де створено обговорення.
AuthorId	UUID		+	Користувач, який ініціював дискусію.
Title	TEXT		+	Головний заголовок обговорення.
IsPinned	BOOLEAN		+	Статус закріплення теми вгорі списку.

продовж.табл.2.14

Назва	Тип даних	PK	FK	Опис
IsLocked	BOOLEAN			Заборона на додавання нових повідомлень.
ReplyCount	INTEGER			Загальна кількість залишених коментарів.
LastPostUserId	UUID		+	Автор найновішого коментаря у гілці.
IsHidden	BOOLEAN			Прапорець приховання теми від загального доступу.

Таблица 2.15

Опис таблиці ForumPosts

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор повідомлення.
ThreadId	UUID		+	Посилання на батьківську тему.
AuthorId	UUID		+	Автор написаного коментаря.
Content	TEXT		+	Текстове наповнення повідомлення.
QuotePostId	UUID		+	Посилання на інший коментар під час цитування.
IsEdited	BOOLEAN			Позначка про наявність змін після публікації.
IsDeleted	BOOLEAN			Позначка про видалення запису.
CreatedAt	TIMESTAMP			Точний час відправлення.
UpdatedAt	TIMESTAMP			Час останньої модифікації тексту.

Таблица 2.16

Опис таблиці UserLists

Назва	Тип даних	PK	FK	Опис
Id	UUID	+		Унікальний ідентифікатор приватного списку.
UserId	UUID		+	Власник створеного списку.

Назва	Тип даних	PK	FK	Опис
Name	TEXT			Користувачька назва категорії.
Description	TEXT			Короткий опис призначення списку.
CreatedAt	TIMESTAMP			Час формування списку.
UpdatedAt	TIMESTAMP			Час останньої зміни атрибутів.

2.3. Проектування та реалізація алгоритмів роботи системи

Діаграми активності належать до діаграм поведінки програми та потрібні для моделювання бізнес-процесів і логіки виконання алгоритмів. Вони нагадують блок-схеми, що робить їх зрозумілими для розробників та замовників. Головна особливість полягає у відображенні послідовних та паралельних дій. Діаграми активності є переходом від одного кроку у вебплатформі до наступного.

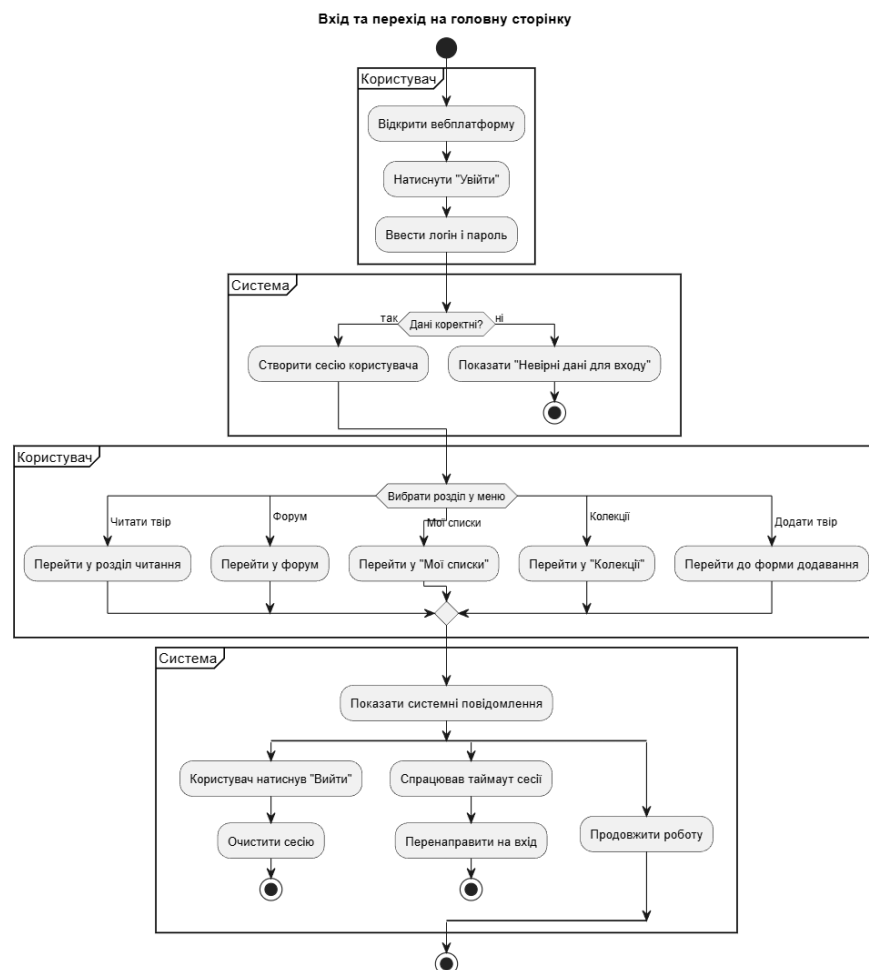


Рис. 2.6. Діаграма активності користувача при вході на головну сторінку

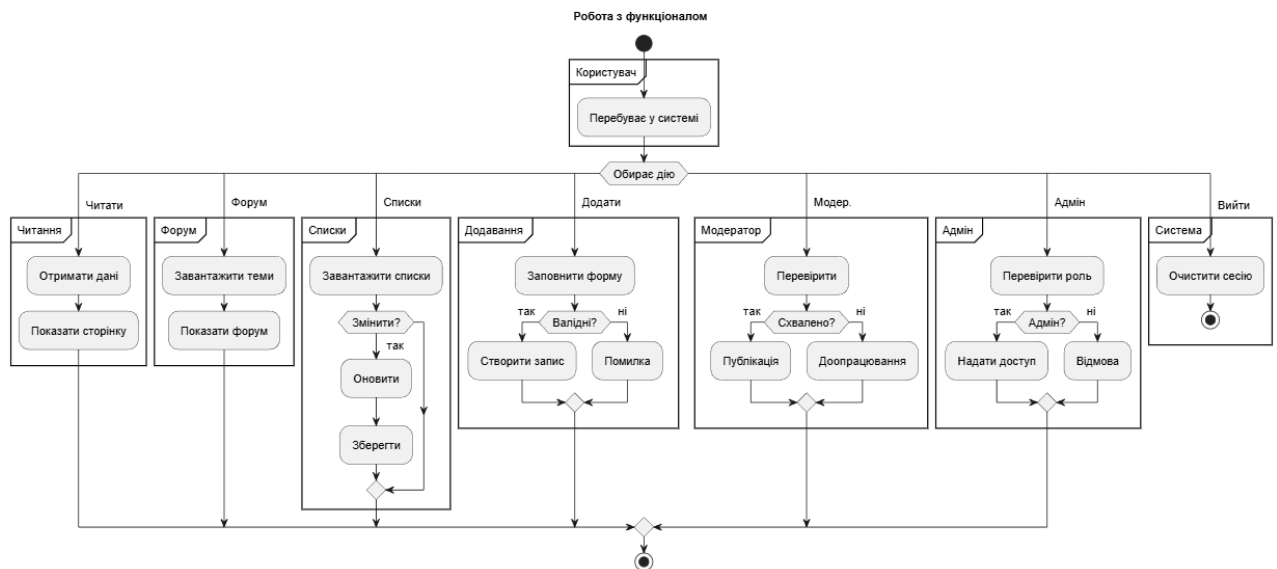


Рис. 2.7. Діаграма активності користувача при роботі з функціоналом

На діаграмі (рис. 2.6) показана робота користувача з вебплатформою, вона починається з відкриття сторінки та переходу до вікна авторизації. Майбутній читач вводить свої дані, після чого система перевіряє їх на правильність написання. Якщо щось з них написано неправильно, на екран виводиться попередження, і процес входу призупиняється до наступної спроби. У разі успішної перевірки система починає сесію з доступом до усіх можливостей вебплатформи.

Після створення або входу до акаунту читач переходить на головну сторінку, звідки може переміщуватися між всіма доступними модулями. Навігація для нього відбувається залежно від того що йому потрібно, у нього є можливість перейти до сторінки читання творів, відкрити форум для спілкування з іншими користувачами по різним питанням, переглянути власні списки читання або ж дослідити колекції. Для перекладачів передбачений перехід до форми додавання нового розділу твору з його сторінки. Разом з діями користувача працює блок для показу сповіщень. Також слідкує за активністю людини для запобігання вичерпання часу сесії та очищає дані при натисканні кнопки виходу.

У висновку можна сказати, що кожен крок починає свій сценарій роботи. До прикладу, у розділі списків користувач може не тільки передивлятися збережені твори. Процес викладення нового твору включає етап перевірки

введеної інформації та подальше надсилання на перевірку модератору. Для персоналу вебплатформи реалізований функціонал, модератори працюють із чергою нових творів або повідомлення в коментарях та на форумі та вирішують чи можна їх опублікувати чи відправити на доопрацювання, тоді як адміністратори можуть переходити до закритої панелі керування вебплатформою.

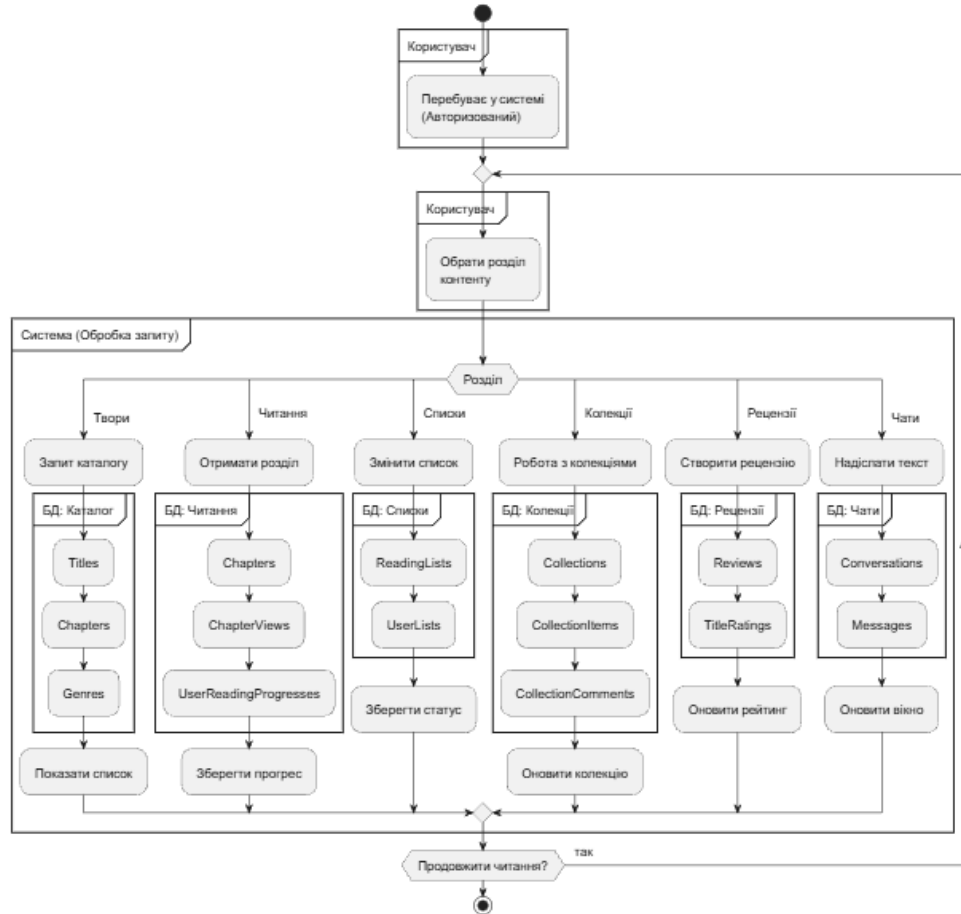
Детальна логіка роботи з головними можливостями вебплатформи (рис. 2.7) показує процеси взаємодії між читачем та системою. Коли користувач відкриває сторінку твору, сайт відразу підвантажує необхідну текстову інформацію та реалізовує інтерфейс для читання. Так само, перехід до форуму активує збір усіх тем, які були додані та виводить їх для перегляду. Робота з колекціями дозволяє не лише переглядати збережені створені списки, а й керувати ними, тобто будь-які зміни, додані читачем до своїх списків, система оновлює та зберігає у PostgreSQL.

Процес заповнення каталогу новими творами вимагає від автора заповнення форми. Після відправки матеріалу проводиться перевірка введеного на валідність. Якщо дані написано правильно, система створює новий запис, який потрапляє до модераторів на перевірку, на цьому етапі він перевіряє контент, схвалені матеріали публікуються та це видно усім користувачам, тоді як публікації з порушеннями або помилками повертаються автору на доопрацювання.

Безпека адміністративної частини гарантується перевіркою поточних прав. Якщо звичайний користувач намагається відкрити панель керування, вебплатформа блокує запит і просто повідомляє про відмову в доступі. Вхід до закритого інтерфейсу надається тільки адміністраторам. Не дивлячись на відкритий розділ, або інші дії людина завжди може завершити роботу на вебплатформі, після чого алгоритм очищає сесію та закриває доступ до приватних можливостей.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

Діаграма активності бази даних: Користувачський функціонал



Діаграма активності бази даних: Системні процеси та Керування

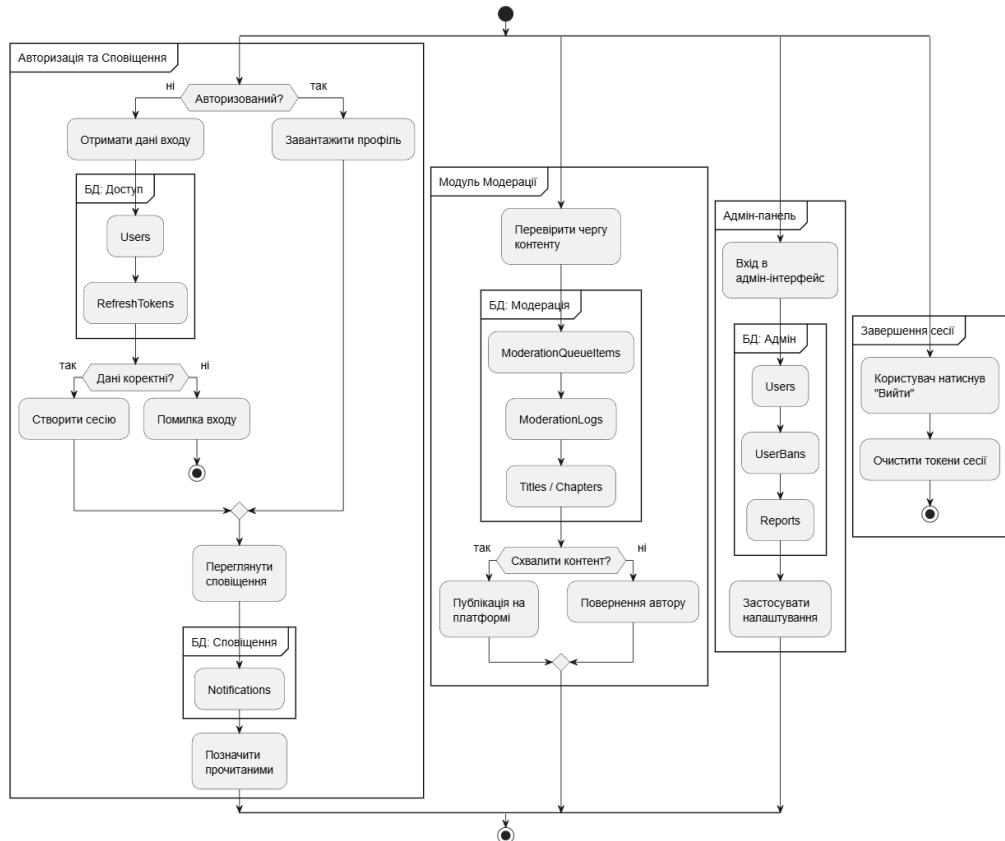


Рис. 2.8 - 2.9. Діаграми активності бази даних

Діаграма активності користувацького функціоналу бази даних (рис. 2.8) показує взаємодію між сервісною частиною вебсайту та фізичним сховищем під час виконання дій зареєстрованого користувача. Цей компонент погоджує роботу користувацьких дій, гарантуючи цілісність інформації при високій навантаженні сайту. Процес побудований навколо одного ключового елемента, звідти відбувається керування даними, що дозволяє виконувати різні модулі функцій на сайті SekaiLib.

Після того як відбувається навігація по каталогу та пошуку творів база даних виконує запити до сутностей Titles, Chapters, Genres, після цього програма починає збирати інформацію про твори, а далі передається увесь список отриманих даних на Frontend. Перехід до читання активує алгоритм, де переглядається вміст розділу твору з таблиці Chapters, логіка додавання перегляду користувача у ChapterViews та зберігання позиції читача в таблиці UserReadingProgresses та через це відбувається відновлення позиції для читання під час наступної сесії.

Робота із персоналізацією та взаємодією між користувачами, підтримують окремі таблиці. Робота читача з власними колекціями та списками змінює таблиці ReadingLists та UserLists, також підтримується стан інтерфейсу з таким же статусом у базі даних. В даний момент новий модуль колекцій працює з сутностями Collections, CollectionItems та CollectionComments. Його робота полягає в створенні колекцій, додавання до них конкретних творів та збереження користувацьких коментарів під ними.

Рецензії та оцінки контенту обробляються через блок рецензій за допомогою операцій в таблицях Reviews та TitleRatings, що дає змогу сайту змінювати рейтинг твору на основі оцінок у рецензіях. Спількування між людьми у чатах відбувається через таблиці Conversations та Messages, які відповідають за збереження історії спілкувань та збереження часу повідомлень у чатах.

Діаграма активності системних процесів та керування (рис. 2.9) показує взаємодію сервісів вебплатформи SekaiLib із базою даних під час забезпечення

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

безпеки та адміністрування. Ця схема незвичайна через те що присутнє розділення логіки на чотири незалежні потоки керування.

Спершу вебплатформа керує циклом сесій користувача. Якщо користувач не авторизований на сайті, починається процес перевірки даних, відправляючи запити до таблиць Users та RefreshTokens. Алгоритм порівнює дані з наявними у базі даних і якщо вони співпадають то створюється нова сесія. При виявленні помилок відбувається завершення виведенням повідомлення про дефект. Для вже авторизованих користувачів завантажується профіль який переспрямовує до обробки сповіщень.

По-друге впорядковує роботу модерації контенту. Процес працює чергою нових матеріалів, проглядаючи записи в таблиці ModerationQueueItems та запам'ятовуючи кожну дію модератора. Залежно від його вердикту він або схвалює матеріал та оновлює статуси у таблицях Titles або Chapters для відкриття доступу до розділу, тоді як відхилення матеріалу повертає його на доопрацювання із причинами чому так було зроблено.

Далі йде блок про можливості адмін-панелі вебплатформи. Після успішного входу під адміністратором в нього з'являється прямий доступ до інструментів управління через редагування профілів користувачів, застосування банів.

Далі йде блок про можливості адмін-панелі вебплатформи. Після успішного входу під адміністратором в нього з'являється прямий доступ до інструментів управління через редагування профілів користувачів, застосування банів. Останній четвертий потік виконує деавторизацію, тобто коли користувач натискає кнопку виходу із акаунту система скасовує сесію та повертаючи систему у звичайних стан гістя замість користувача.

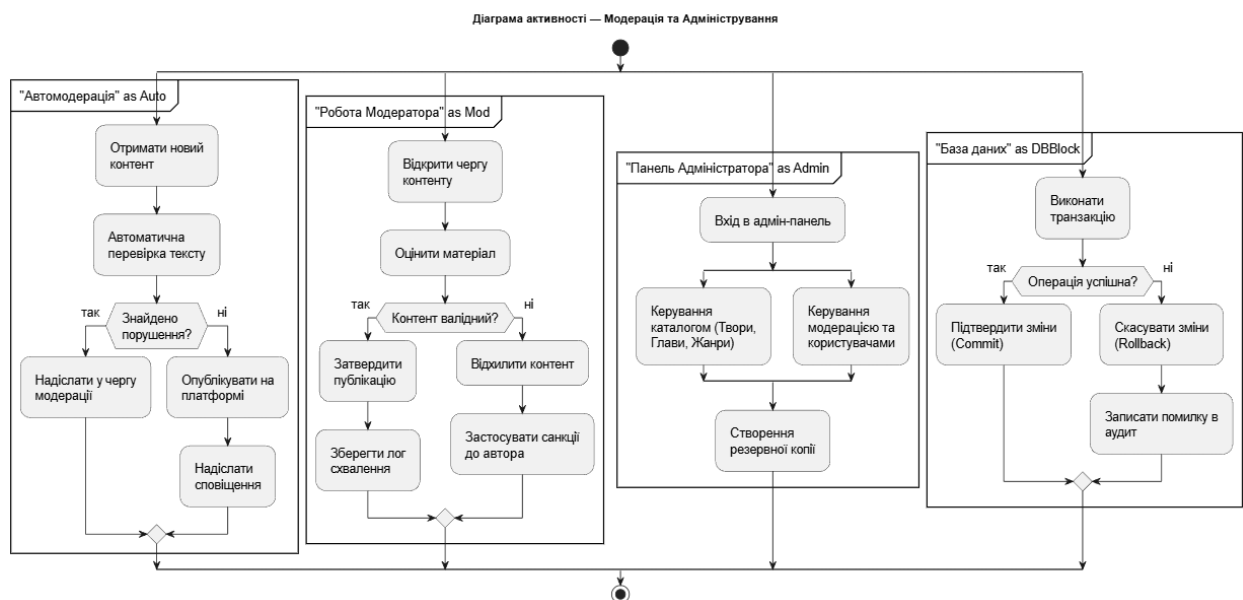


Рис. 2.10. Діаграма активності модератора і адміністратора

На діаграмі активності модерації та адміністрування (рис. 2.10) відображаються обов'язки та автоматизація контролюючих процесів у межах вебплатформи для читання книжок. На діаграмі показано 4 основні вектори, фонову перевірку контенту сервісом, роботу модераторів яка виконується вручну, налаштування адміністратора та сховище бази даних.

Перевірка стартує автоматично, щойно система отримує новий контент від користувачів. Сервісний модуль швидко проганяє текст через заздалегідь налаштовані правила фільтрації. Матеріал одразу летить у спеціальну чергу, де ним займатимуться вже модератори. А якщо автоматичний контроль пройдено успішно, твір або глава виходить у відкритий доступ, і паралельно з цим підсистема формує та розсилає сповіщення підписникам.

Ручна перевірка контенту починається просто. Модератор відкриває чергу з новими або скаргами на сумнівні матеріали і сам оцінює, що до чого. Фінальне рішення залежить тільки від розбору модератора на якість того що перевіряє. Якщо схвалено твір одразу відправляється на сайт. Але якщо модератор бачить порушення правил, контент остаточно бракується. Ну і автор отримує від системи бан або інші обмеження у своєму профілі.

Глобальне управління винесене в окремий адмінський потік, який стартує відразу після входу в закриту панель. Весь функціонал адміна ділиться

на два паралельні напрямки. Перший, це пряме керування літературним каталогом. Другий напрямок, комплексний контроль за модерацією та акаунтами користувачів. Ну а у фіналі цього адміністративного циклу зазвичай запускається створення повної резервної копії, щоб дані точно нікуди не зникли.

Абсолютно всі результати роботи цих модулів зрештою стікаються на фізичний рівень бази даних. Вона крутить усі операції всередині ізольованих транзакцій. Якщо технічна частина проходить гладко і без збоїв, база робить фінальний коміт та остаточно зберігає оновлені дані. Коли відбувається конфлікт чи помилка система просто відкочує всі поточні зміни, повертаючи базу в колишній безпечний стан. Ну і паралельно автоматично реєструє цей збій в аудиторському журналі. Щоб розробники потім могли розібратися, у чому була проблема.

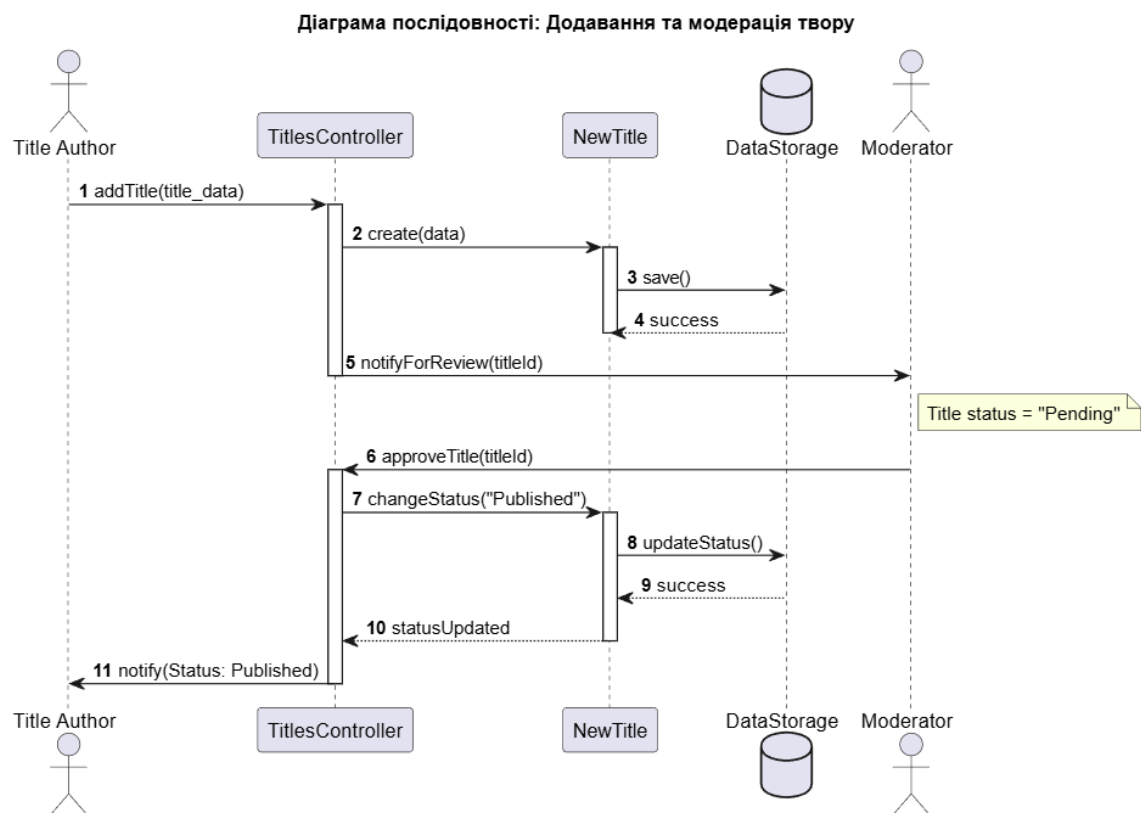


Рис. 2.11. Діаграма послідовності для додавання нового твору та його модерація

На діаграмі послідовності (рис. 2.11) показано, яка розкриває життєвий цикл додавання нового твору на сайт та його перевірки. Через інтерфейс

вебплатформи він автор починає метод `addTitle`, відправляючи дані в `TitlesController`. Отримавши цей пакет, контролер бере на себе роль того хто починає все створювати наприклад новий об'єкт через `create(data)` всередині сутності `NewTitle`, а вся інформація паралельно відправляється у сховище `DataStorage`. Саме на цьому етапі твір отримує тимчасовий статус `Pending`.

Коли запис успішно створено, контролер автоматично відправляє тригер `notifyForReview(titleId)`. Це сповіщення відправляється напряму до Модератора. Далі він вичитує текст, перевіряє його на відповідність правилам сайту і якщо все добре, викликає метод `approveTitle(titleId)`. Контролер змінює статус твору на `Published`, оновлюючи інформацію в базі даних. Ну і наприкінці контролер надсилає автору фінальний нотифікатор `notify(Status: Published)`, і після цього твір буде у відкритому доступі для того щоб читачі могли з ним ознайомитися.

Вся динаміка системи тримається на абсолютно класичних PostgreSQL процесам [18]. Наприклад системні сповіщення чи приватні чати працюють через `Observer` на базі хабів `SignalR`. Також щоб база не зламалася від високонавантажених запитів, було підключено `Cache-aside`. Цей механізм крутиться на `Redis` через `DistributedReadCacheService.cs`. Він достаньно вагомо знижує пряме навантаження на основну базу даних.

Архітектура взаємодії між керуючими компонентами та фізичною базою даних тримається на ієрархії патернів (додаток А). Взяти хоча б контролери, вони грають роль `Facade`, просто приховуючи за собою виклики до бізнес-логіки. А сама логіка обробки вже винесена окремо на `Service Layer` як у `TitleService`. Робота з даними ізольована абстракціями, для цього працює класичний патерн (додаток А) `Repository` з базовим `Repository` та різними специфічними реалізаціями.

Для гарантування цілісності даних всі розрізнені операції запису акуратно об'єднуються через `Unit of Work`. В результаті виходить гарантія цілісності транзакцій. Збереження нової рецензії та перерахунок рейтингу твору гарантовано проходять як одна неподільна операція.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		59

Діаграма комунікації: Публікація рецензії



Рис. 2.12. Діаграма комунікації публікація рецензії та оновлення рейтингу твору

На діаграмі комунікації (рис. 2.12) показано, як саме спілкуються об'єкти, коли читач вирішує залишити рецензію на прочитане. Система тут працює як злагоджений механізм, щоб гарантувати цілісність даних і правильно автоматизувати розрахунок рейтингу.

Усе починається з читача (Reader). Він відправляє ієрархічне повідомлення 1: submitReview(titleId, rating, text) напряму до ReviewController. У цих даних лежить усе необхідне: ID твору, числова оцінка та текст самої рецензії. Отримавши запит, контролер береться за роботу і координує одразу кілька паралельних процесів.

Спочатку контролер просто створює об'єкт через create(data) під NewReview. А потім закидає його прямо в DataStorage (використовуючи

виклик `save(review))`, потім збереглася рецензія. Але далі важлива частина з оновленням загальних показників твору. Контролер напряду звертається до сутності `Title` і передає їй команду `updateRating(new_rating)`

Фінал усієї цієї діаграми такий, Команда 1.4: `saveChanges()` попадає прямо в базу даних, фіксуючи там оновлений рейтинг. На цій схемі, добре видно дотримання принципу слабкої зв'язності. Сутності рецензії та твору взагалі не взаємодіють між собою. Їхню спільну роботу повністю синхронізує `ReviewController`.

Описані вище діаграми комунікації чудово візуалізують архітектуру і допомагають відловити потенційні вузькі місця ще до написання першого рядка коду. Така детальна ієрархія повідомлень потім стає чудовою базою для розробки модульних тестів. Бо кожен виклик тут працює як чітко прописаний контракт між об'єктами. А ще на схемі добре видно орієнтацію на низьку зв'язність компонентів. Це важливо для майбутнього масштабування вебплатформи. Всі ці моделі динамічної поведінки працюють, вони гарне поєднання між сухою концепцією та етапом написання бекенду.

2.4. Реалізація функціоналу вебплатформи для читання книжок

Додавання нового твору до каталогу, це один із основних сценаріїв у `SekaiLib`. На рівні веб-API все відбувається навколо методу створення в `TitlesController.cs`, який просто отримує POST-запит від користувача, який має доступ до усього контенту, Але система також обов'язково перевіряє токен доступу, і тільки потім передає модель далі в сервіс.

Самі вхідні дані потрапляють через спеціальний об'єкт `CreateTitleRequest.cs`. Там немає нічого зайвого, тільки назва, автор, опис, посилання на обкладинку, статус, країна та ID вибраних жанрів. Він класно розділяє зовнішній формат запиту і нашу внутрішню доменну модель, та й ризик пропустити якесь сміття чи небезпечні дані зводиться до мінімуму.

Вся незвичайна логіка живе у `TitleService.CreateAsync`, спочатку перевіряємо, чи взагалі існує цей користувач у базі. Формується `Title`, потім

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		61

генерується унікальний ідентифікатор. Для кожного вибраного ID жанру створюється зв'язок із твором, щоб каталог потім працював як треба.

Тільки після цього сформований об'єкт відправляється у базу даних через репозиторій. Зберігаємо зміни, система ще раз витягує цей свіжий запис з усіма пов'язаними даними, щоб нормально повернути його клієнту. Тож це далеко не просто сліпий запис у таблицю, є повноцінний цикл. Починаємо з перевірки читача та збірки доменної сутності, а закінчуємо схрещенням жанрів, збереженням у БД і фінальною віддачею актуальної інформації.

Редагування інформації про твір у SekaiLib працює через звичайний HTTP PUT-ендпоінт у TitlesController. Клієнт просто закидає туди об'єкт UpdateTitleRequest. У цьому DTO зібрані тільки потрібні поля - назва, автор, опис, посилання на обкладинку, статус, країна походження та список ID жанрів, немає нічого зайвого. Завдяки такій структурі зовнішнє API ізольоване від нашої внутрішньої доменної моделі.

Далі контролер дістає ID користувача прямо з токена доступу (через ClaimTypes.NameIdentifier) і передає виклик у TitleService.UpdateAsync. А вже всередині сервісу починається усі можливі перевірки. Спочатку дивимось, чи взагалі існує такий Title у базі, якщо немає то відразу кидаємо NotFoundException. Змінювати твір може виключно його власник або ж адміністратор. Якщо хтось інший спробує це зробити, система відразу видасть ForbiddenException. Потім просто перезаписуються всі поля на зразок Name, Author, Description та оновлюється час UpdatedAt. З жанрами логіка трохи інша, старі зв'язки TitleGenres повністю стираються, а замість них створюються нові під ті GenreIds, що прийшли в запиті. Для фінального збереження сервіс смикає UpdateAsync у репозиторії та робить _unitOfWork.SaveChangesAsync(), щоб усі зміни працювали цілісно.

Після успішного збереження клієнт отримує оновлені деталі твору у вигляді повного DTO. Але є один цікавий нюанс, якщо під час редагування статус твору змінився на Completed, система починає шукати всіх підписників і розсилати їм повідомлення через NotificationService. Ця можливість дозволяє

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

оперативно оновлювати клієнтські інтерфейси без потреби перевідправляти важкі запити щоразу, коли хтось відкриває сторінку.

У плані безпеки весь цей процес максимально закритий. Атрибут `Authorize` на контролері залізно гарантує, що гості без токена сюди просто не зможуть нічого зробити. Внутрішня логіка сервісу надійно захищає від чужого втручання та тримає в порядку всі зв'язки. І використання DTO відіграє тут не останню роль, воно просто блокує передачу неконтрольованого сміття і сильно полегшує валідацію на рівні API.

Модерація контенту працює як повністю окремий сервіс. Все поділено на два етапи. Спочатку йде автоматична перевірка, а потім вже ручна робота. Автомодація робить такий собі первинний скринінг. Якщо знайдено щось підозріле то одразу формується запис і кидає його в чергу, за весь цей функціонал відповідає `AutoModerationService`. Він просто повертає результат перевірки, і якщо спрацював флаг, елемент попадає у `ModerationQueue`.

А от основна ручна логіка вже виконується всередині `ModerationService`. Як виглядає типовий сценарій, модератор запитує чергу, а сервіс читає записи з таблиці `ModerationQueueItems`. Далі модератор просто бере запис у роботу, переглядає сам контент та всю історію автомодації. І приймає рішення, `Approve` або `Reject`.

Найголовніше тут те, що обидва ці шляхи виконують строго неподільні операції в межах однієї транзакції. При `Approve` система оновлює статус у черзі та статус самого твору, відразу додає запис у `ModerationLogs` і створює нотифікацію для автора. Всі ці операції зберігаються разом. З `Reject` історія схожа, статус міняється на `Rejected`, логується причина відмови і у автора з'являється сповіщення. Транзакція так само зберігається або ж відкочується при помилці. Звісно, модератор може зробити й інші дії, наприклад, видати бан чи попередження користувачу або підредагувати словник заборонених слів. І кожна така дія обов'язково потрапляє в `ModerationLogs`.

Нотифікації ходять через `NotificationService` та хаби `SignalR`. Щойно запис потрапляє в таблицю `Notifications`, система миттєво відправляє подію в

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

хаб. Це дозволяє оновлювати інтерфейс підписників просто в реальному часі. І дуже важливо робити всі ці змінні операції (статуси, логи, нотифікації) в одній транзакції. Щоб не було жодної моментів які будуть відрізнятися, для цього ModerationService вже використовує UnitOfWork та SaveChanges.

Всі дії модератора зберігаються в логах. Чисто для аудиту та статистики, яку завжди можна витягнути через GetStatsAsync. Взагалі, автомодерація зі своїми фільтрами та спам-правилами допомагає із зайвим навантаження. Але всі записи з флагом однаково мусять потрапляти в чергу для остаточного вердикту живої людини. Ну і доступ до всього цього контролюється ролями. Змінювати ці ролі може тільки адміністратор, що напряду впливає на те, які ендпоінти взагалі будуть доступні модераторам.

Додавання рецензії в SekaiLib працює через REST-ендпоінт у ReviewsController. HTTP POST-метод Create просто приймає запит від користувача. Далі вся основна логіка опрацюється в ReviewService через метод CreateAsync. Самі вхідні дані ми додаємо в спеціальний DTO CreateReviewRequest. Там лежить тільки заголовок, текст та рейтинг. Такий підхід гарантує нам нормальний формат і сильно полегшує серверну валідацію.

Контролер спочатку витягує userId, якщо користувач неавторизований, одразу повертається 401 помилка, далі сервіс валідує всі поля. Він перевіряє наявність тексту із заголовком, ріже довжину (заголовок до 200 символів, текст до 2000) і заганяє рейтинг у рамки від 1 до 10. якщо є перебільшення символів показується ValidationException. Потім треба впевнитися, що читач ще не писав рецензію на цей конкретний твір. Ну і звісно перевіряємо, чи взагалі існує сам твір у базі даних.

Потім працює вступає автомодерація, весь текст із заголовком одразу закидається напряду в AutoModerationService. Цей алгоритм шукає щось підозріле, якщо знаходить, відразу вішає свій прапорець, а рецензія миттєво стає невидимою, отримує статус IsHidden. І далі вона падає в чергу на перевірку до живого модератора. Необхідно для виключення спаму чи

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

токсичної інформації, щоб не заважало звичайним читачам читати корисні рецензії.

Вже після цього етапу нарешті формується сама сутність Review з усіма даними. Записується вона в базу даних через UnitOfWork та SaveChangesAsync. Після збереження сервіс просто витягує об'єкт, пакує його в DTO і спокійно віддає контролеру статус 201 Created та робота завершується і цю рецензію видно користувачам.

Будь-які важливі події обов'язково проходять через NotificationService. Система пише сповіщення в таблицю і відразу відправляє подію в хаб SignalR. UnitOfWork надійно страхує нас від часткового запису. Сервіс завжди звіряється з IUserBlockService, щоб заблоковані користувачі не могли нічого коментувати. А ще клієнт у відповідь отримує не просто голий відгук, а одразу спаковану статистику. Кількість лайків, дизлайків, коментарів та загальний рейтинг самого рецензента. Фронтенду не треба робити купу зайвих запитів під час відображення сторінки.

Висновки до другого розділу

У другому розділі була повністю спроектована архітектура вебплатформи SekaiLib. Був вибраний звичайний об'єктно-орієнтований підхід. Спочатку розібралися з усіма вимогами та ролями користувачів, щоб скласти адекватну мапу варіантів використання. Завдяки цій ієрархії тепер зрозуміло, яка частина системи за що відповідає. Далі зібрали детальну діаграму класів, у центрі всього логічно опинилися сутності User, Title та Chapter. Ну і звісно, додано допоміжні компоненти для взаємодії між користувачами та збереження прогресу читачів.

Для бази даних була вибрана реляційна PostgreSQL. Всюди було використано UUID для ідентифікаторів, також додано окремі таблиці під керування сесіями та зовнішніми логінами. Це дало хороший запас безпеки та міцності для майбутнього масштабування. А щоб дані не дублювалися, структуру максимально нормалізовано.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		65

Для роботи програми в динаміці допомогли діаграми активності та послідовності. Вони дозволили прогнати складні алгоритми ще до написання коду. Ту ж саму багаторівневу модерацію або логіку доступу до платних розділів. Діаграми комунікації взагалі підтвердили орієнтацію на слабку зв'язність. TitlesController та ReviewController виступають ключовими координаторами, по суті реалізуючи патерн Facade. Уся ж робота з базою прихована за патерном Repository та об'єднана в транзакції через Unit of Work.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		66

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ ДЛЯ ЧИТАННЯ ЕЛЕКТРОННИХ КНИЖОК

3.1. Визначення варіантів використання структури системи

Система SekaiLib розробляється спеціально для випусної кваліфікаційної роботи бакалавра. Замість мікросервісної архітектури було використано сучасний модульний та контейнеризований підхід. Кожен ключовий компонент живе у власному ізольованому середовищі. Через це розгортання стає максимально гнучким, і дає можливість спокійно масштабувати той самий Redis або PostgreSQL абсолютно незалежно від основного API [20, 21]. Це рятує і сильно полегшує управління вебплатформою під майбутні навантаження. А щоб все це було наглядно видно було підготовлено діаграму (рис. 3.1)

Вебплатформа SekaiLib спирається на модульну архітектуру і розділена на ключові компоненти. Клієнтська частина зібрана на зв'язці React та Vite [25]. За бекенд відповідає ASP.NET Core Web API (усе це для зручності запаковано в Docker-контейнери) [7]. Для зберігання даних ми взяли надійний PostgreSQL. Завдяки такому розділенню збої в одному місці не ламають усю вебплатформу. А ще можна незалежно розробляти чи оновлювати сервіси і масштабувати ресурси саме під той компонент, який зараз навантажений найбільше.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		67

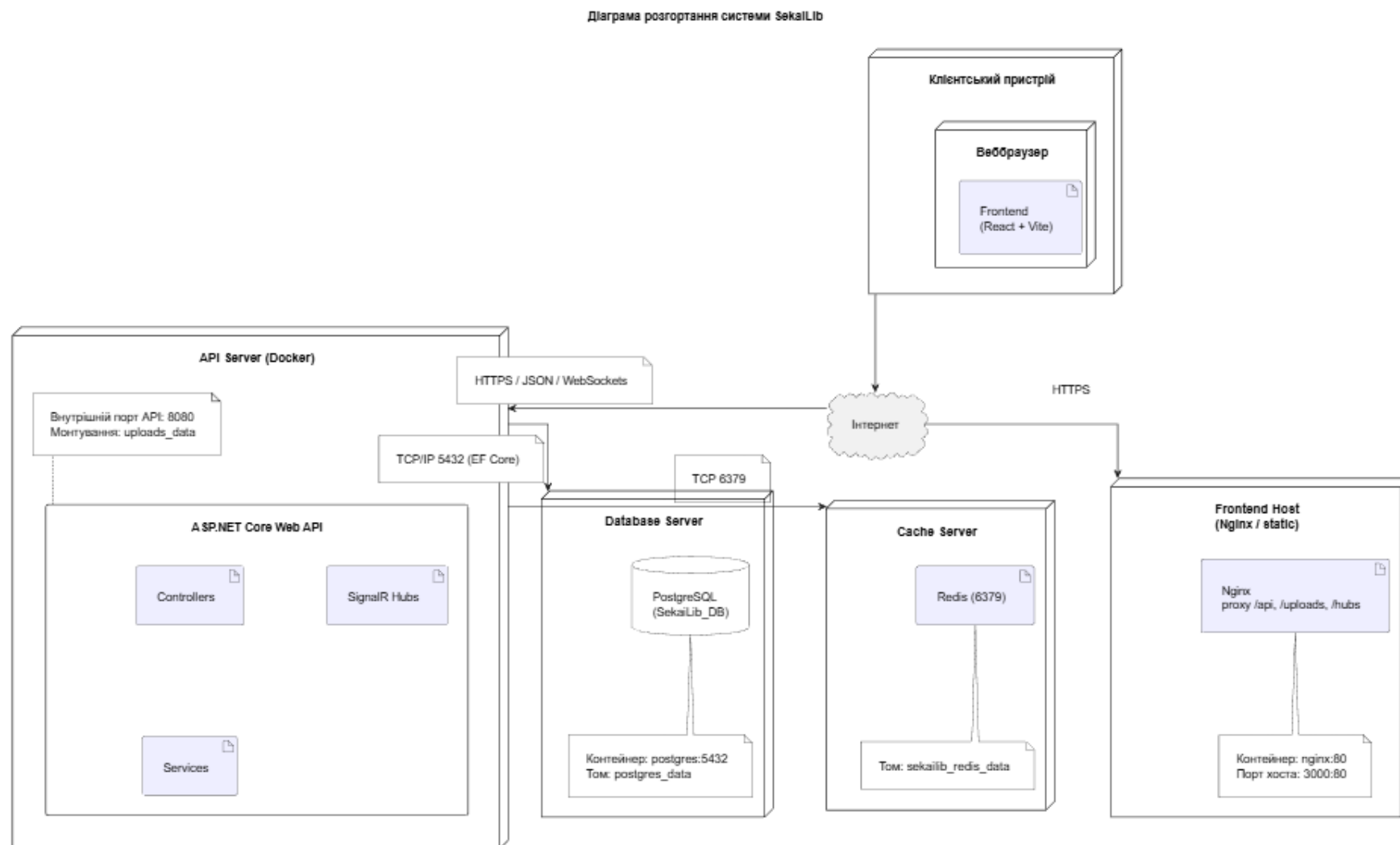


Рис. 3.1 Діаграма розгортання вебплатформи

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		68

У таблиці наведено вимоги до програми

Таблиця 3.1

Вимоги до програми

Компонент	Мінімальні вимоги	Для чого потрібно
Процесор	8 ядра 2,5 ГГц	Щоб сервер не падав під час різних стрибків трафіку.
Оперативна пам'ять	16 ГБ	Інакше докер-контейнерам з базою та кешем будуть давати збої.
Дисковий простір	50 ГБ SSD	Потрібен запас під розростання бази, нескінченні логи та завантажені користувачами обкладинки.

Процес розгортання SekaiLib можна запустити кількома шляхами, але найпопулярніший варіант, це автоматизація через Docker Compose [22, 24]. Спочатку, треба додати на сервер Docker разом із Docker Compose і просто клонувати репозиторій з кодом. Далі необхідно створити файл .env прямо в корені вебплатформи, туди треба прописати всі ключові змінні оточення, на кшталт, секретів чи портів. Потім заходимо в потрібну директорію через термінал і запускаємо збірку. Команда `docker compose up -d` підніме всі сервіси, наостанок потрібна перевірка статусів контейнерів через звичайний Bash, щоб переконатися, що все працює без помилок.

Також є можливість розгорнути без докер-контейнерів. Спочатку збирається релізна версія вебплатформи на ПК або через налаштований CI/CD конвеєр. Після збору отриману папку з бінарниками (зазвичай це `out` або

publish) просто перекидаємо на сервер. На самому сервері, прописується правильний рядок підключення до бази PostgreSQL, виставляються потрібні порти та змінні. І наостанок налаштовуємо запуск програми як системну службу systemd на Linux. Якщо база даних розгортається окремо від основного файлу налаштувань, треба виконати спеціальну команду для застосування міграцій Entity Framework Core. Без цього застосунок просто не зможе нормально зв'язатися з таблицями і не буде працювати.

Також виконану випускню кваліфікаційну роботу бакалавра було розміщено на серверах від Hetzner. Я орендовав їхній стандартний хмарний VPS. Встановив туди базовий Linux, підняв середовище Docker і розгорнув всі контейнери. Для вебплатформи SekaiLib виділених потужностей вистачає для коректної роботи.

3.2. Структура інтерфейсу вебплатформи

Вебплатформа для читання електронних книжок доступна або через локалхост або через сервер у відкритому доступі

Посилання на сайт (для захисту): <https://sekailib.tantane.me/>

Після того як авторизований користувач потрапляє на головну сторінку (рис. 3.2) він бачить навігаційну панель там розміщений клікабельний логотип, щоб повернутися на цю сторінку, також вкладки каталог, форум, новини та команди, у правому куті присутній дзвінок з усіма повідомленнями та ім'я користувача у вигляді випадаючого списку там де він може виконувати дії зв'язані з ним.

Під панеллю є відображення новин там де користувач може переглянути їх, коли адміністрація додасть їх в загальний доступ, в них видно дату сам зміст із назвою та ім'я адміністратора який цю новину виклав. Ще є слайдер який показує останні 15 перекладених розділів до творів з відображенням їх номера та країну випуску.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

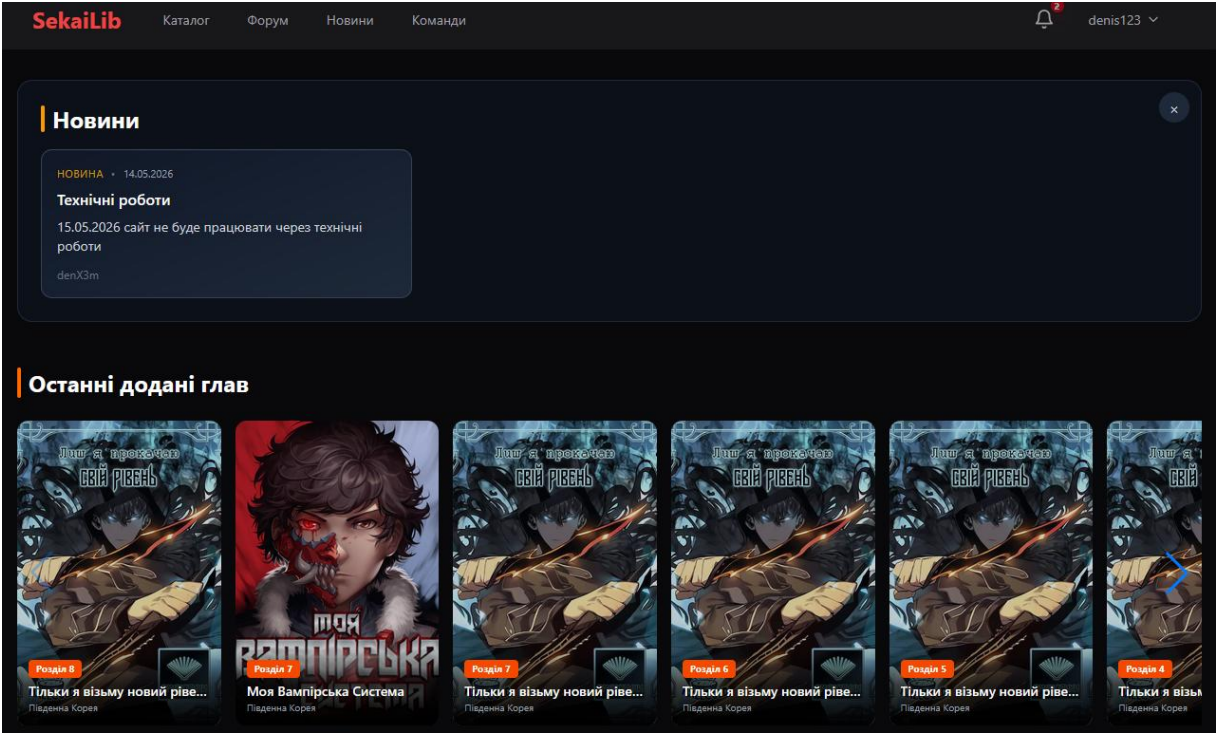


Рис. 3.2 Вигляд новин та останніх доданих розділів на головній сторінці

А вже під ним є спеціальний список оновлення від команд, (рис. 3.3) який показує оновлення від команд на які він підписався. Далі є окремий блок “Продовжити читання” в якому користувач може продовжувати читати ті розділи творів на яких він зупинився.

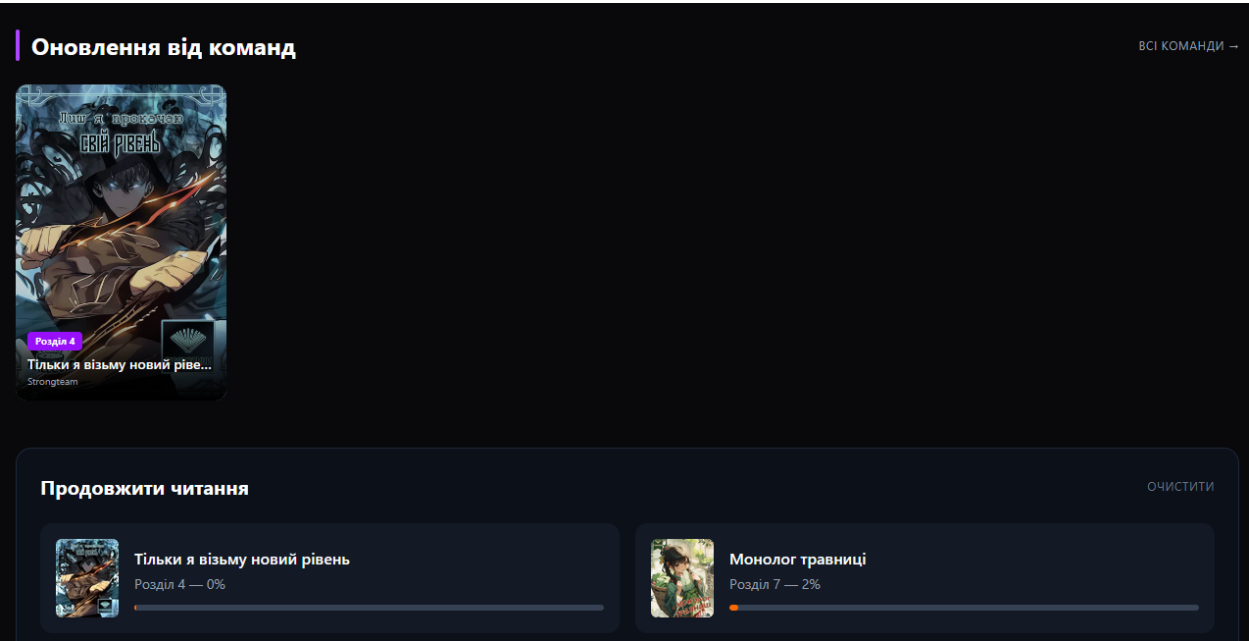


Рис. 3.3 Вигляд оновлень від команд та продовження читати користувача

Наприкінці сторінки є блок “Останні додані твори” в якому відображаються останні 10 нових творів доданих на вебплатформу (рис. 3.4),

кожен твір показаний у вигляді невеликої картки, на ній користувач може побачити потрібно інформацію таку як, обкладинку твору, його назву та ім'я автора.

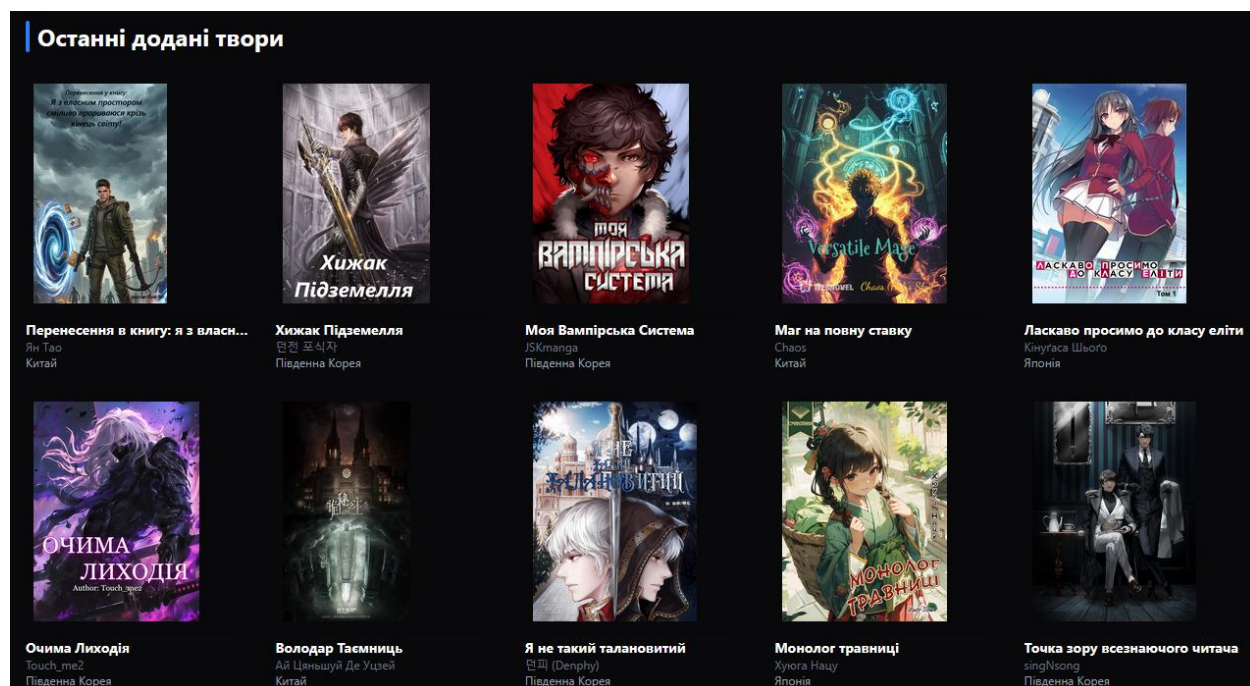


Рис. 3.4 Вигляд останніх доданих творів

Коли користувач натискає на дзвінок, він бачить випадаючий список (рис. 3.5) там присутня основна інформація про яку читачу потрібно знати: релізи нових розділів, відповіді на коментарі, особисті повідомлення чи нові заявки в друзі. Якщо щось додано до дзвінка буде показуватися червона цифра з кількістю повідомлень, але якщо їх буде більше 9 то буде видно 9+, а всередині відображається як окрема картка з обкладинкою твору, або аватаркою, заголовком, коротким текстом та коли повідомлення було відправлено користувачу, ще є червона крапка, яка означає, що повідомлення ще не прочитане.

Щоб було зручно знаходити потрібні повідомлення у загальному потоці, були додані фільтри (Всі, Розділи, Відповіді, Особисті та Інше). Зверху праворуч є червона кнопка “Прочитати все”, яка миттєво скидає всі лічильники та прибирає червоні індикатори.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

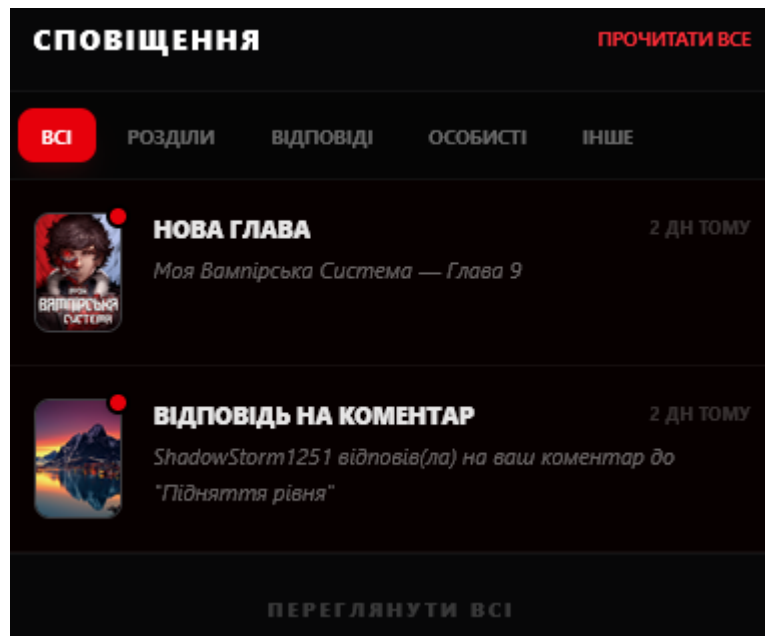


Рис. 3.5 Вигляд дзвоника користувача

А в самому низу присутнє посилання "Переглянути всі". Воно вже перекидує на окрему повноцінну сторінку з усією історією сповіщень (рис. 3.6). Ця сторінка виконана для того щоб користувач міг знаходити потрібне повідомлення якщо воно було видалене зі дзвоника

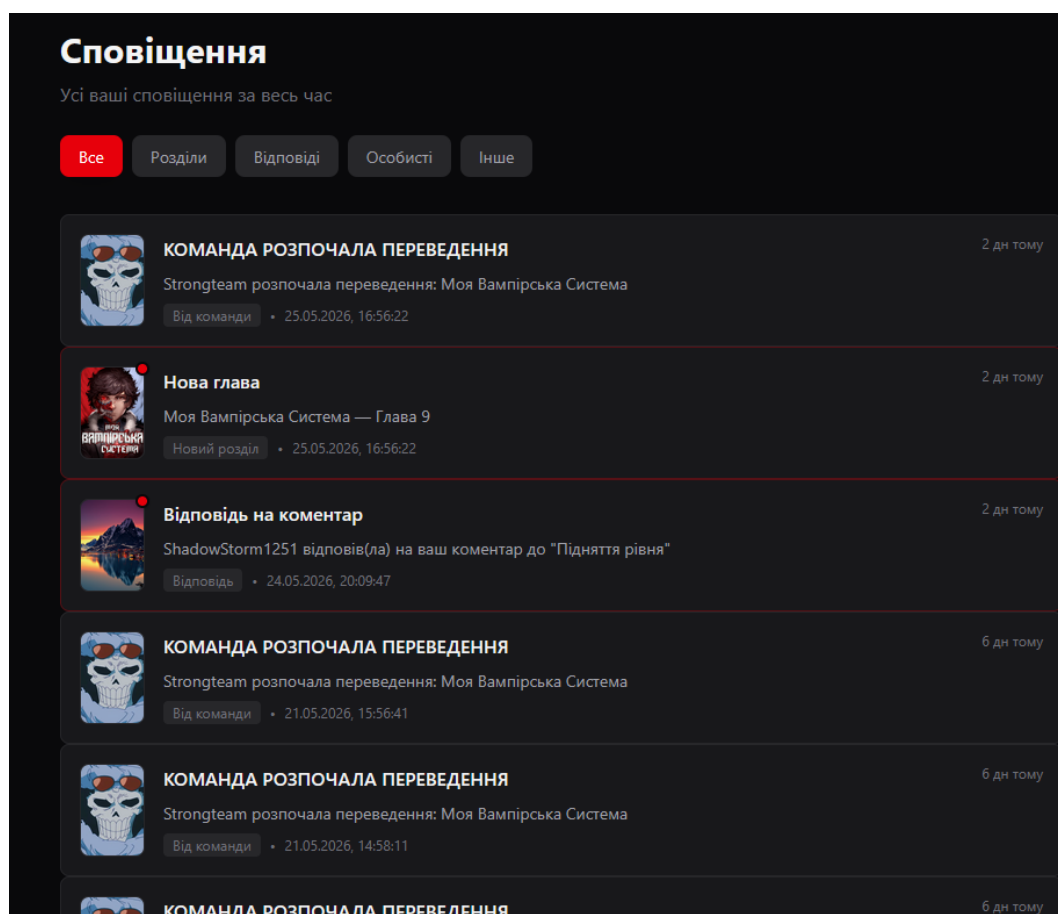


Рис. 3.6 Історія сповіщень користувача

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

При вході у каталог творів користувач бачить всі твори, які є на сайті (рис. 3.7), якщо в своїх налаштуваннях не додав один із жанрів до небажаних. Він може шукати їх за назвою або автором та фільтрувати по жанрам, країнам випуску та статусам перекладу, також є можливість очистити фільтри однією кнопкою. Самі твори зображені як обкладинка твору, його назва автор та рейтинг (якщо він є).

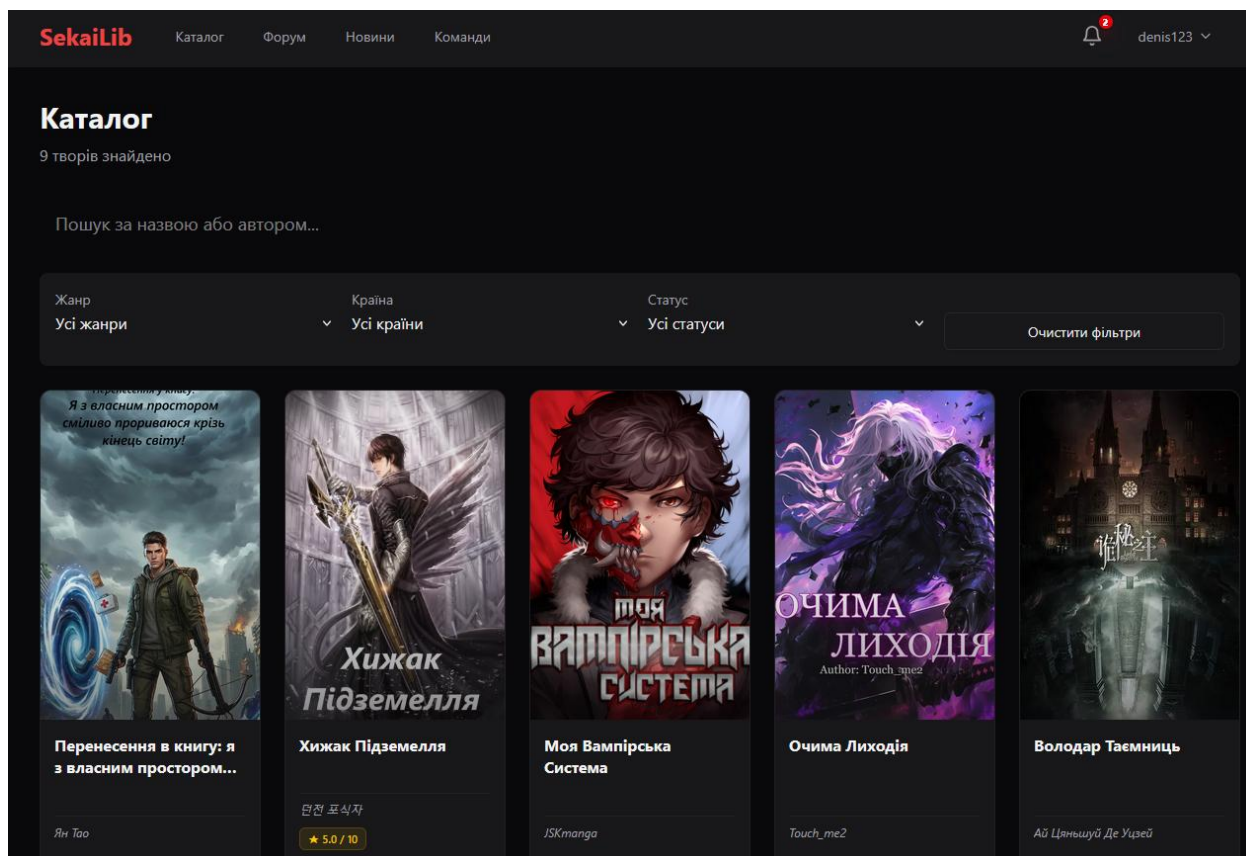


Рис. 3.7 Сторінка каталогу

Щоб користувач міг повноцінно використовувати функції вебплатформи, йому потрібно створити власний обліковий запис, коли він натискає на кнопку з'являється форма поверх основної сторінки (рис. 3.8) в якій людина може або зайти через гугл акаунт або ввести дані такі як пошта, ім'я, пароль та його підтвердження, якщо користувач уже є в базі даних то видно помилку "Validation failed", але якщо людина вже була зареєстрована то вона може вибрати вкладку вхід і після введення даних свого акаунту зможе зайти на сайт під ним.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		74

×

Створити акаунт

Вхід

Реєстрація

Continue with Google

АБО

Пошта

Ім'я користувача

Пароль

Підтвердження пароля

Validation failed

Створити акаунт

Адміністратори та модератори можуть блокувати користувачів за різні порушення правил на сайті, та воно може бути вічним або на деякий час і користувач не зможе користуватися сайтом (рис. 3.9).

Рис. 3.9 Забанений користувач

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Сфремов Ю.М.				75
Змн.	Арк.	№ докум.	Підпис	Дата		

Також була створена сторінка форуму для взаємодії читачів між собою (рис. 3.10), де щоб уникнути хаосу в обговореннях тільки адміністратор або модератор можуть створювати категорії для обговорення, на яких буде відображатися скільки в категорії присутньо тредів та коментарів в них.

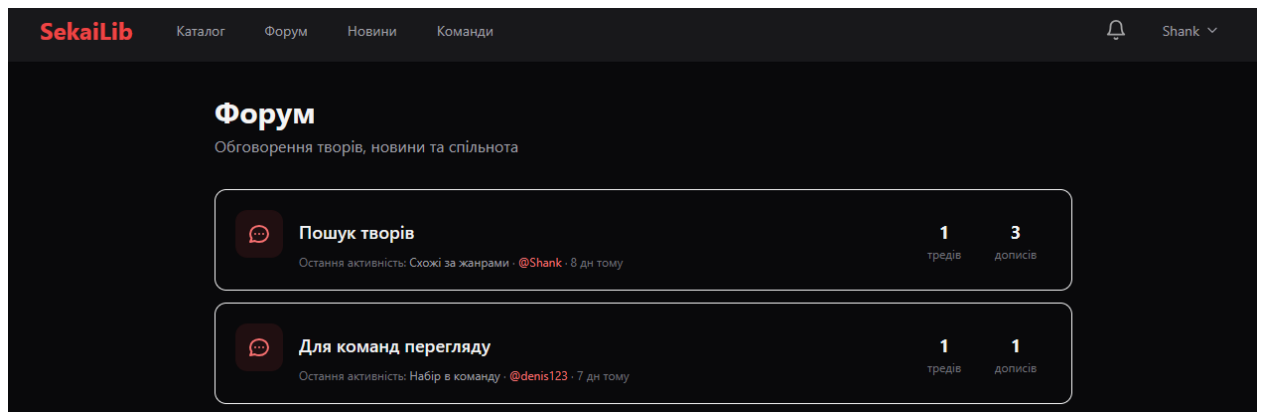


Рис. 3.10 Головна сторінка форуму

Після переходу в саму категорію користувач потрапляє у внутрішній розділ (рис. 3.11) де зареєстрований користувач може створювати треди для обговорення, саме обговорення виглядає як рядки з інформацією на них видно назву треду під нею ім'я автора який неї створив, час публікації та загальна кількість відповідей та переглядів до цього обговорення.

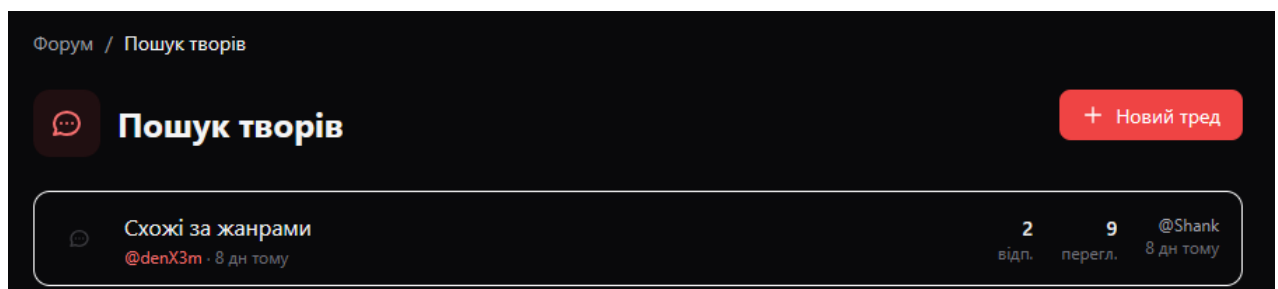


Рис. 3.11 Сторінка тредів всередині категорії

Після заходу в тред (рис. 3.12) користувач пише головну тему для обговорення або запитання, також показується дата, час, перегляди та кількість відповідей до обговорення. Стрічка треду виконана у вигляді послідовних карток-повідомлень Користувачі можуть відповідати, редагувати або видаляти свої повідомлення, а на повідомлення інших людей відповідати, та ставити лайки/дизлайки.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

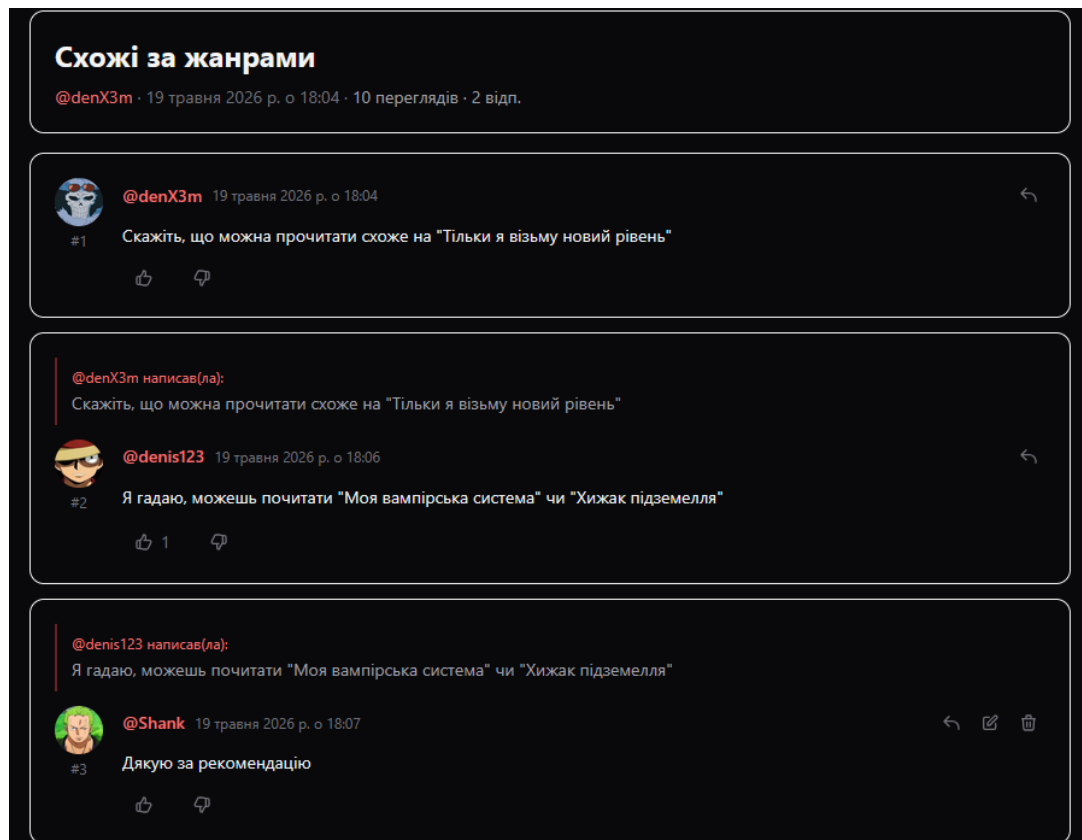


Рис. 3.12 Вигляд обговорення на форумі

Користувач може відвідати сторінку новин (рис. 3.13) на якій він може переглядати повідомлення від адміністрації для читачів та люди можуть також фільтрувати чи це новина або оновлення.

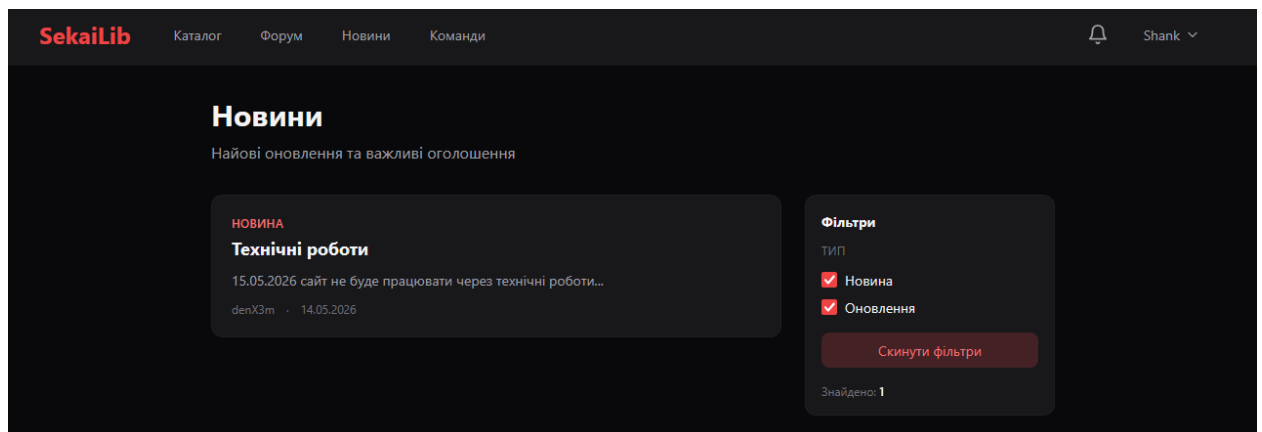


Рис. 3.13 Сторінка новин

При натисканні кнопки “Команди” користувач потрапляє на сторінку (рис. 3.14) та бачить список всіх команд на сайті, з відображенням аватарок, назви, кількості переведених розділів, творів, та підписників.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

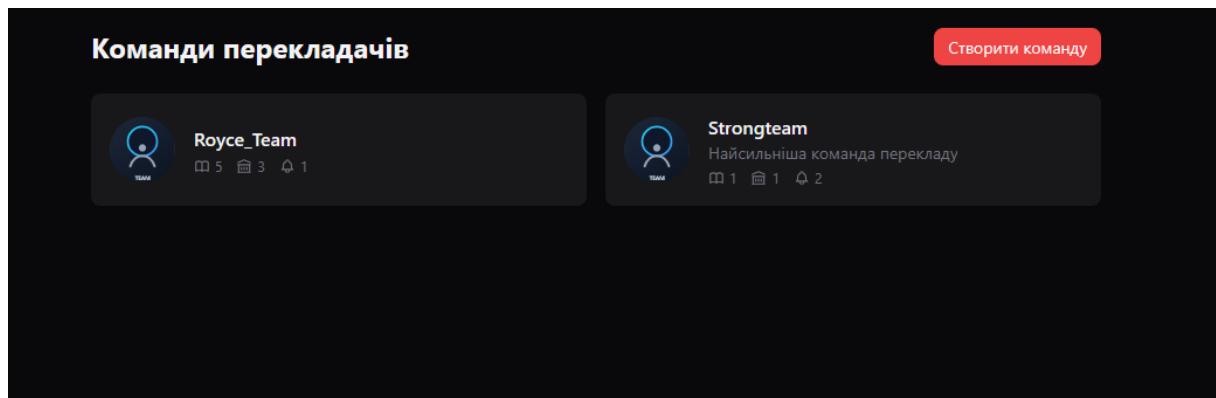


Рис. 3.14 Сторінка команд

Для користувача також є кнопка щоб він створив свою команду, потім його перекидає на форму (рис. 3.15) в якій можна написати назву команди, опис до неї, вибрати аватарку команди та фон профілю, файл можна додати зі свого пристрою чи посилання на фото, якщо нічого не буде вибрано то будет зображення стандартне.

Рис. 3.15 Форма створення команди

При заході на сторінку команди перекладу (рис. 3.16) зверху видно фонове зображення, після нього зображення команди його опис та ім'я власника (якщо зайшов власник то він може редагувати), також знизу є блок

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				78
Змн.	Арк.	№ докум.	Підпис	Дата		

який показує всі розділи які команда переклала, та вкладка учасники щоб подивитися або додати нового користувача.

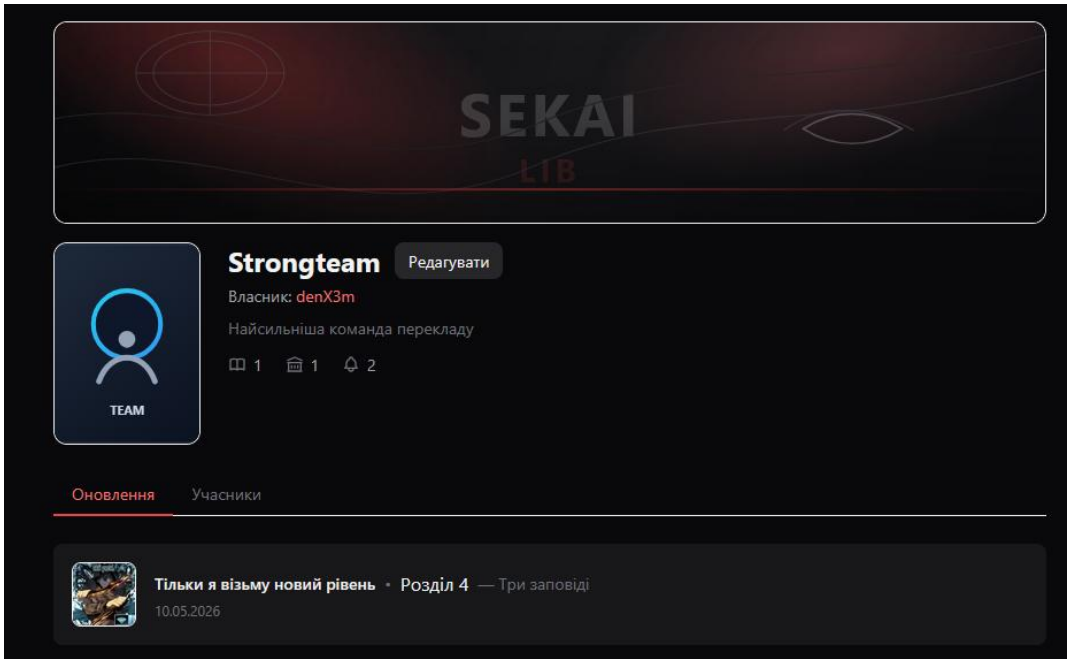


Рис. 3.16 Вигляд сторінки команди

Ще є вкладка “Колекції” яка знаходиться або в випадаючому списку при натисканні на ім’я користувача або у футері. На самій сторінці (рис. 3.17) присутня кнопка “Створити” при настиканні на яку з’являється форма в якій потрібно написати назву та опис до створеної колекції а також вибрати чи буде вона публічною чи приватною.

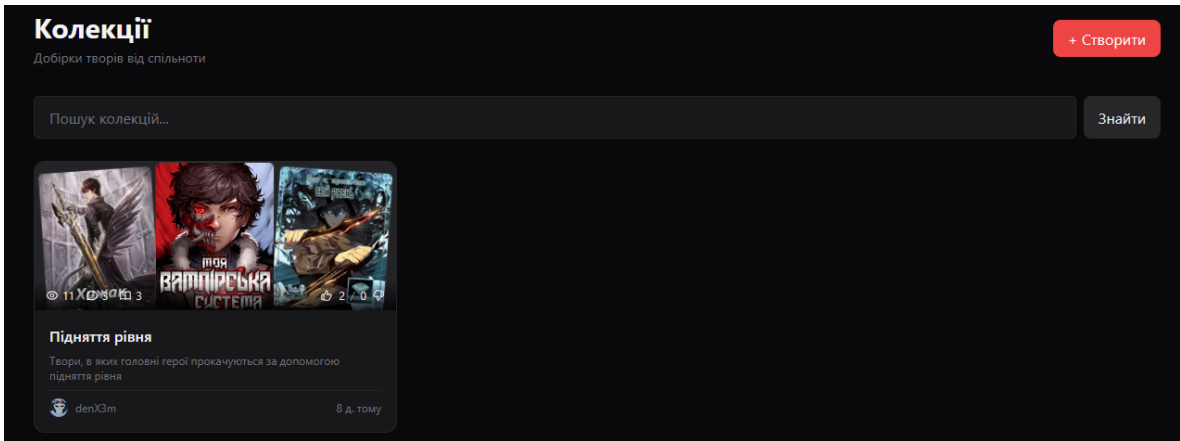


Рис. 3.17 Вигляд сторінки колекцій

При заході на сторінку колекцій (рис. 3.18) є можливість дивитися кількість переглядів та коментарів і спілкуватися з іншими користувачами внизу сторінки, якщо на сторінку зайшов її автор то є можливість додати новий

розділ для сортування творів доданих до колекції, також під розділами є можливість ставити лайки/дизлайки до того що створив користувач.

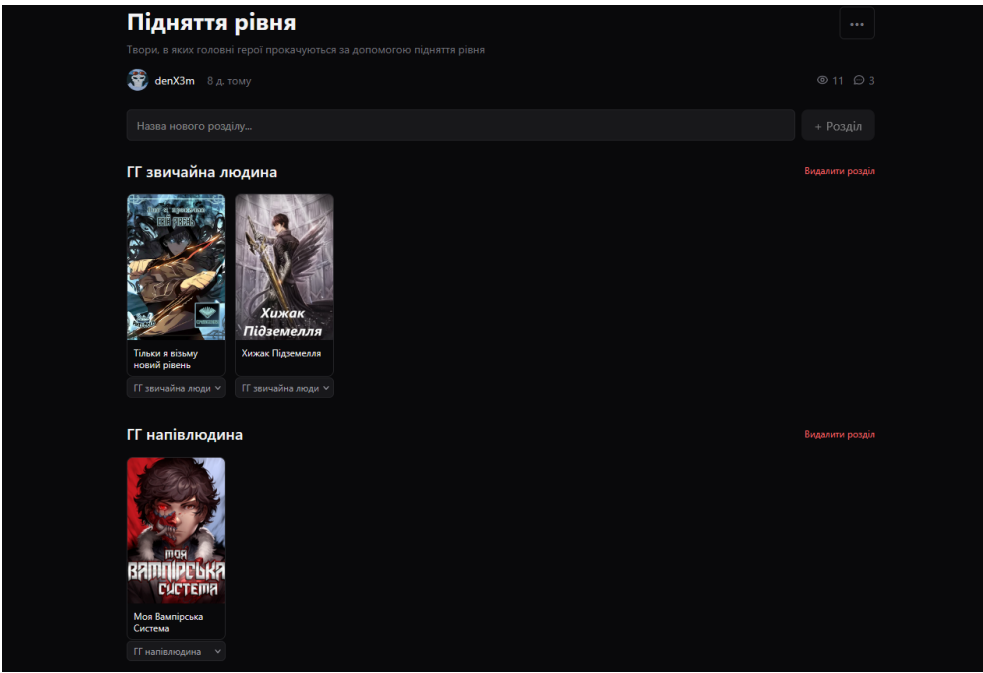


Рис. 3.18 Колекція користувача

Тепер до одної з двох основних частин - це сторінка твору, при заході на сторінку (рис. 3.19) відразу видно зображення твору, його назва, автор, та ким опубліковано, після цього йдуть метадані: статус перекладу, країна, оцінка яка вираховується з рецензій та кількість лайків та дизлайків. Трішки нижче є кнопка для додавання твору до свого списку та додавання до колекції якщо в користувача є вже створена, після цього йде детальний опис твору та вивід усіх жанрів.

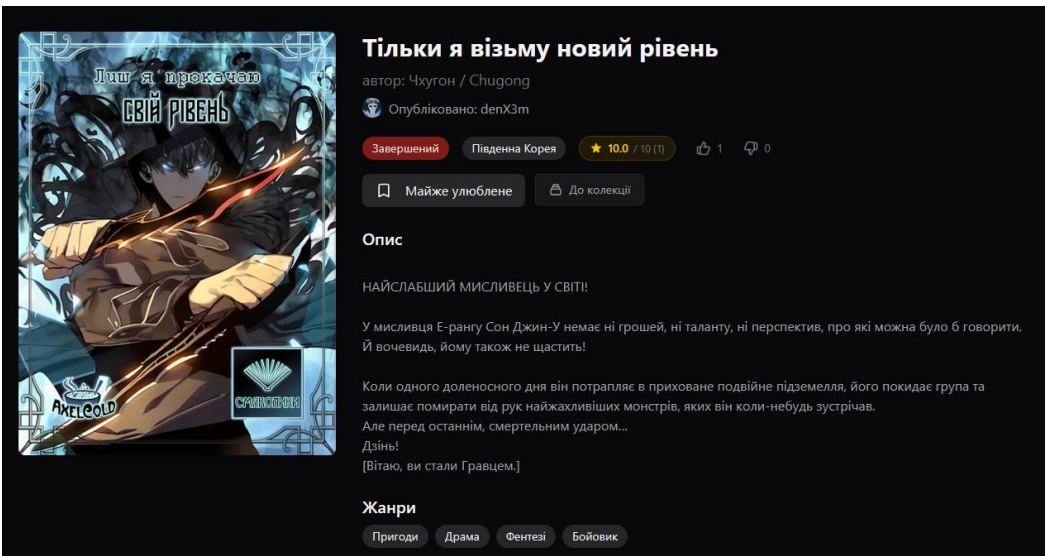


Рис. 3.19 Сторінка твору

Далі нижче наведено список команд (рис. 3.20), які переводять цей твір якщо їх там більше двох то можна вибрати саме то команду від якої користувач хоче читати. Потім присутній блок “Розділи” в якому можна сортувати переклади або вибрати всі та буде видно кожен розділ, а якщо вибрати якусь команду то буде вибрано розділи від цієї команди та ті які були додані без команд

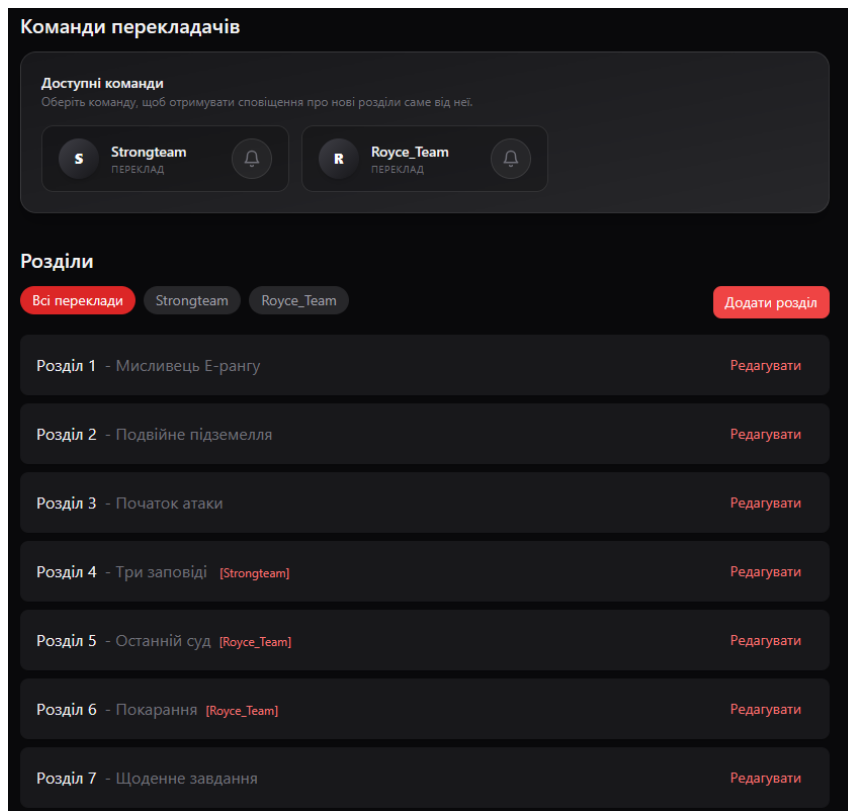


Рис. 3.20 Розділи твору

В самому кінці сторінки є можливість писати коментарі та відповідати на них або відправити скаргу на перевірку модератору (рис. 3.21).

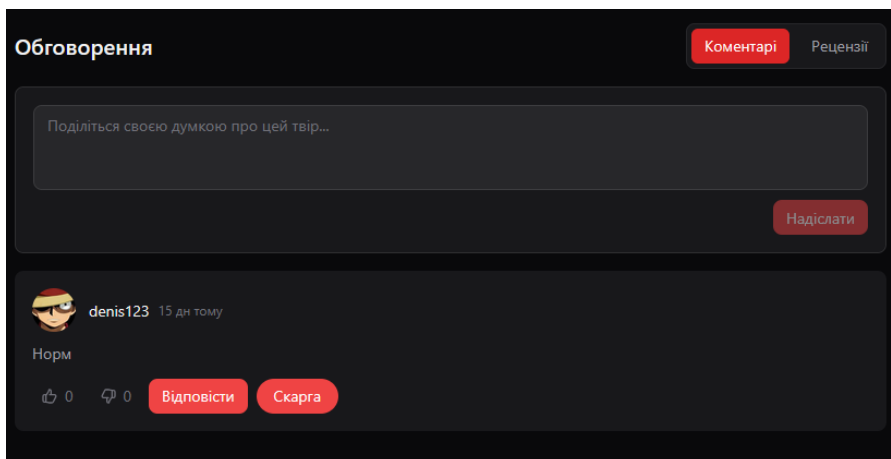


Рис. 3.21 Коментарі під твором

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

Також там є кнопка змінити коментарі на рецензії, при натисканні на цю кнопку видно усі рецензії (рис. 3.22), також можна окремо дивитися усі, позитивні, нейтральні, негативні та сортувати їх. Рецензія відображається блоком з ім'ям користувача, коли вона була написана, оцінкою, вступом та неповним описом, після цього видно кількість переглядів, коментарів, лайків та дизлайків. Унизу блоку є оцінка рецензента, яка потрібна для відображення рейтингу довіри до користувача, яка вираховується лайками до його рецензій, та кнопка читати повністю яка перевідправляє на сторінку рецензії [13].

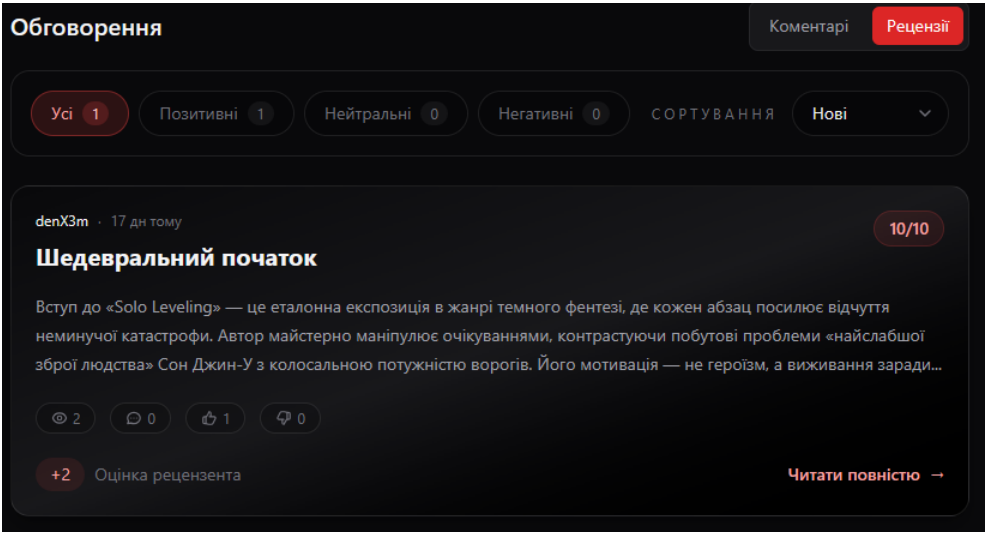


Рис. 3.22 Рецензія під твором

На сторінці рецензії (рис. 3.23) відображається автор, оцінка рецензента, повний вміст рецензії та редування та видалення для самого автора, також можна прокоментувати неї або відправити скаргу.

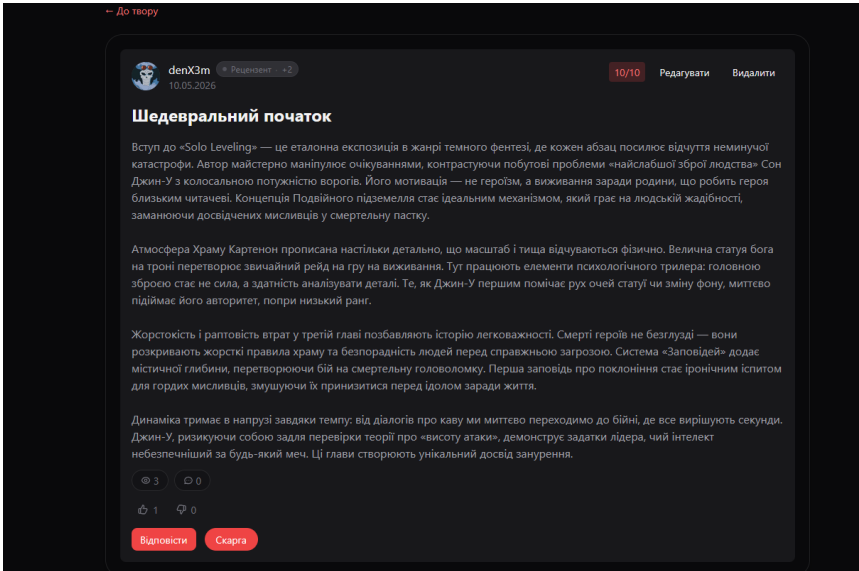


Рис. 3.23 Сторінка рецензії

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

Кнопка додавання розділу (рис. 3.24) на головній сторінці доступна лише користувачам у який є учасником команди перекладу після натискання, на кнопку додавання розділу, з'являється нова сторінка, на якій треба ввести номер розділу, назву, її текст та вибір від якої команди перекладу викладається твір, також унизу є можливість чи це буде преміум розділ, який, буде платним, якщо це преміум розділ то з'являється поле для ціни твору та інше поле щоб користувачам було видно коли розділ стане безкоштовним.

Рис. 3.24 Сторінка додавання розділу

Якщо користувач намагається відкрити розділ який команду перекладу виклала як платний на певний час, то система перенаправляє читача на сторінку з обмеженим доступ замість читання (рис. 3.25). Головну увагу зосереджено на основному інформаційному блоці, спочатку відображається іконка замка, під неї написано назву твору та номер розділу з його назвою, також написана ціна розділу та коли вона стане безкоштовною і кнопка придбати за конкретну сумму яка відправляє користувача на сторінку оплати.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		83

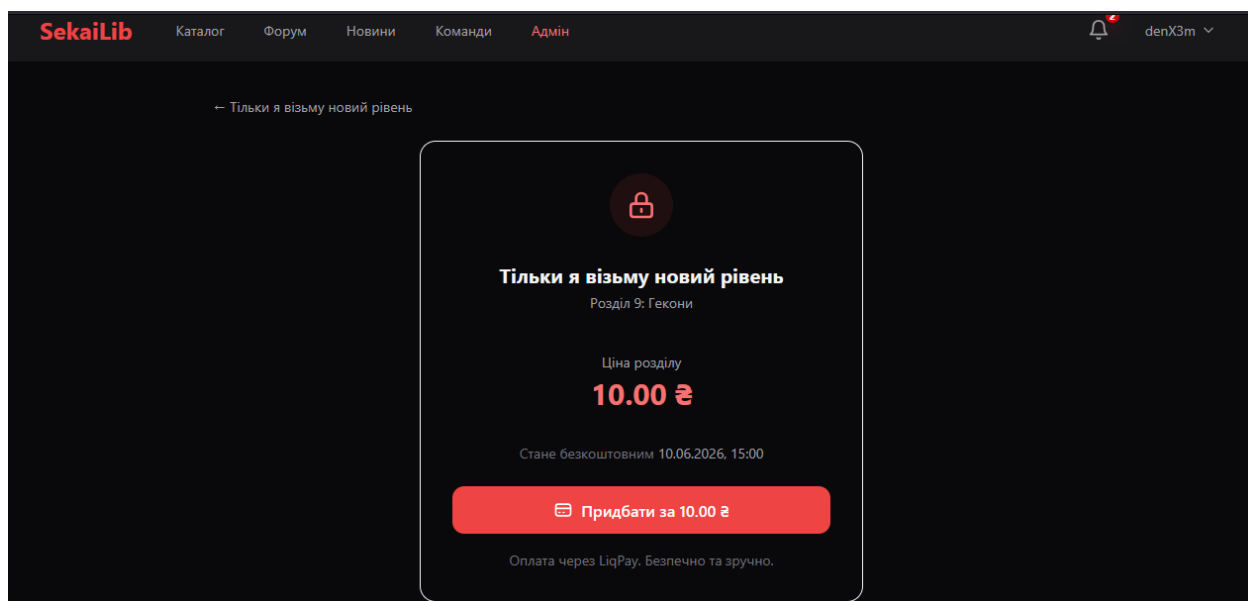


Рис. 3.25 Вигляд не відкритого розділу

Після натискання кнопки придбати користувача перенаправлює на сторінку оплати через Liqpay (рис. 3.26), користувач може або ввести дані карти для оплати, або придбати по QR-коду.

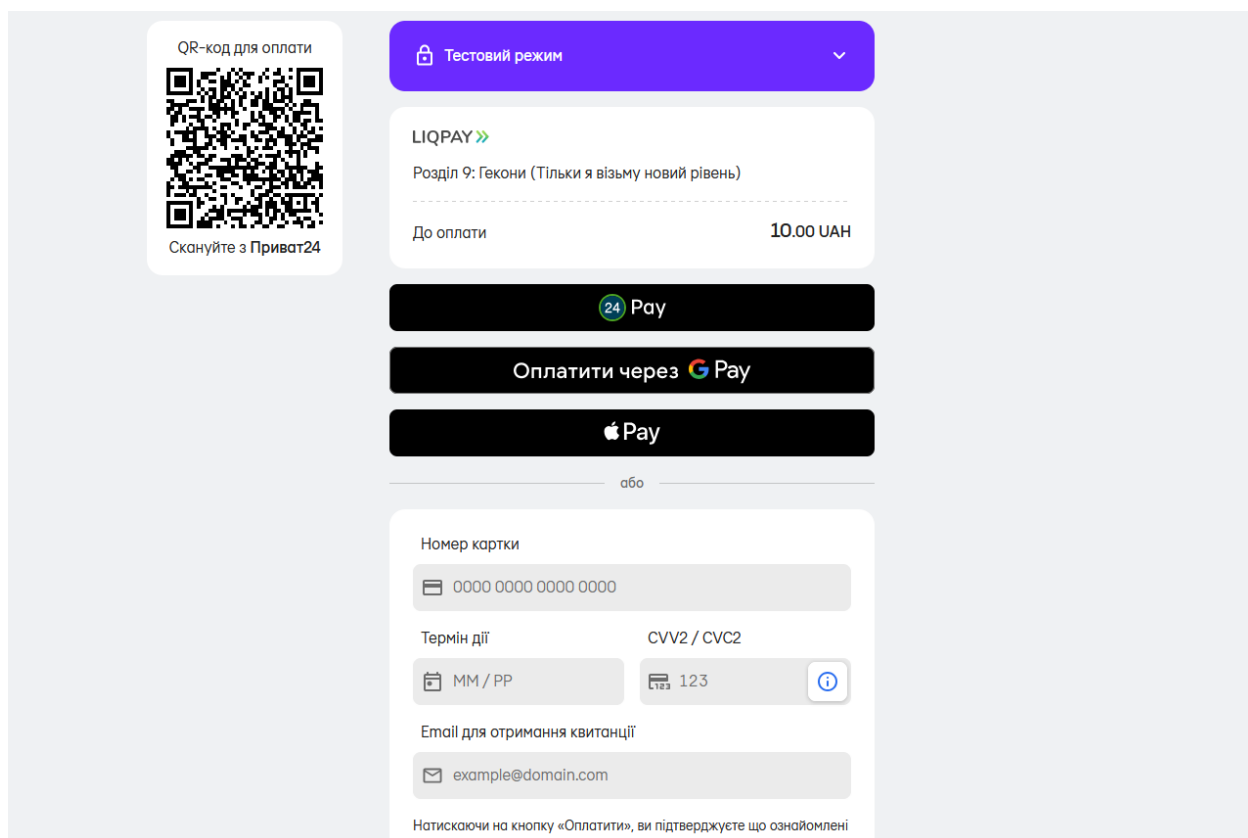


Рис. 3.26 Сторінка оплати

Коли він переходить на сторінку оплати то розділ на творі відображається як “Перевіряємо оплату” (рис. 3.27).

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		84

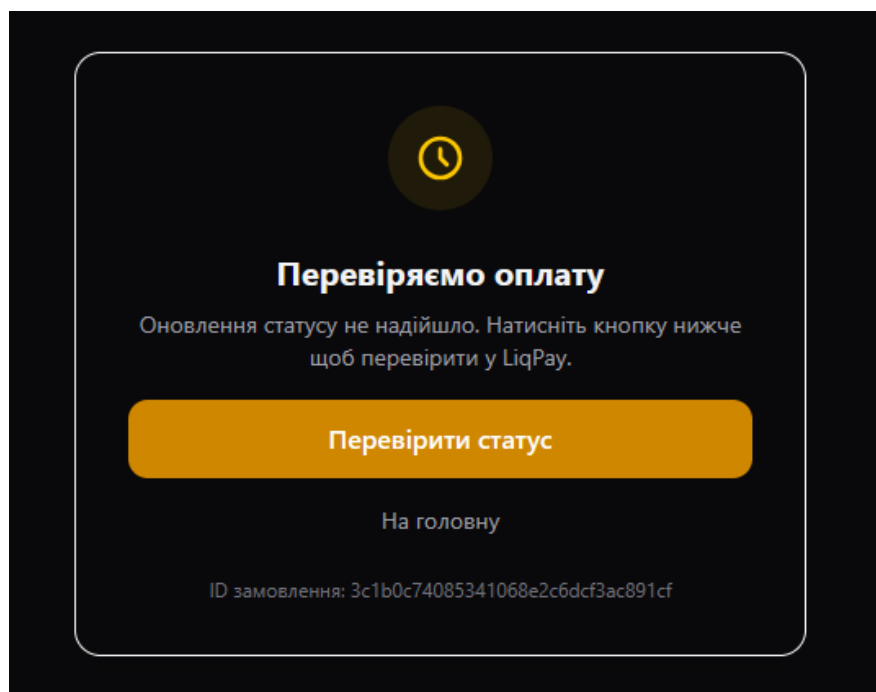


Рис. 3.27 Очікування оплати

Після оплати (рис. 3.28) на сайті проводиться перевірка чи все введено вірно, якщо так то розділ для людини відкривається та вона може його прочитати.

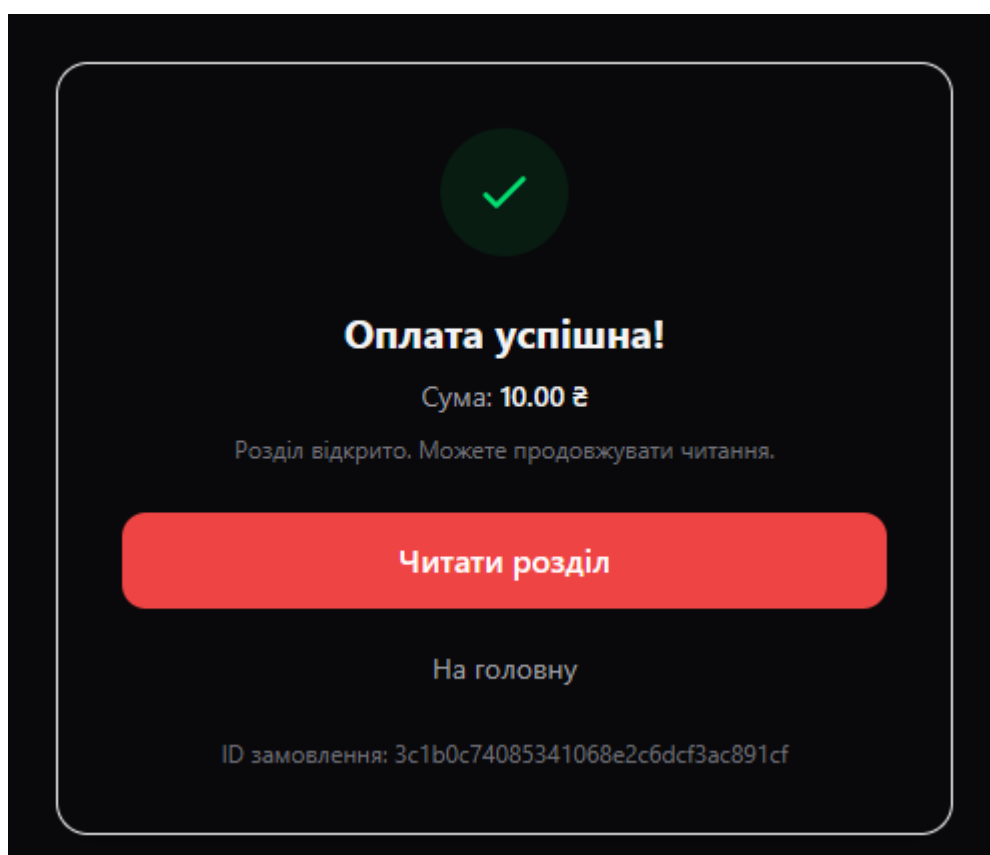


Рис. 3.28 Виконана оплата

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		85

Друга головна частина - це читання самих творів, на сторінці розділів (рис. 3.29) є можливість повернутися до сторінки твору вибрати тему сторінки (світла, темна або сепія) та розмір шрифту (маленький, стандартний або великий), також крім назви твору та розділу є кількість переглядів.

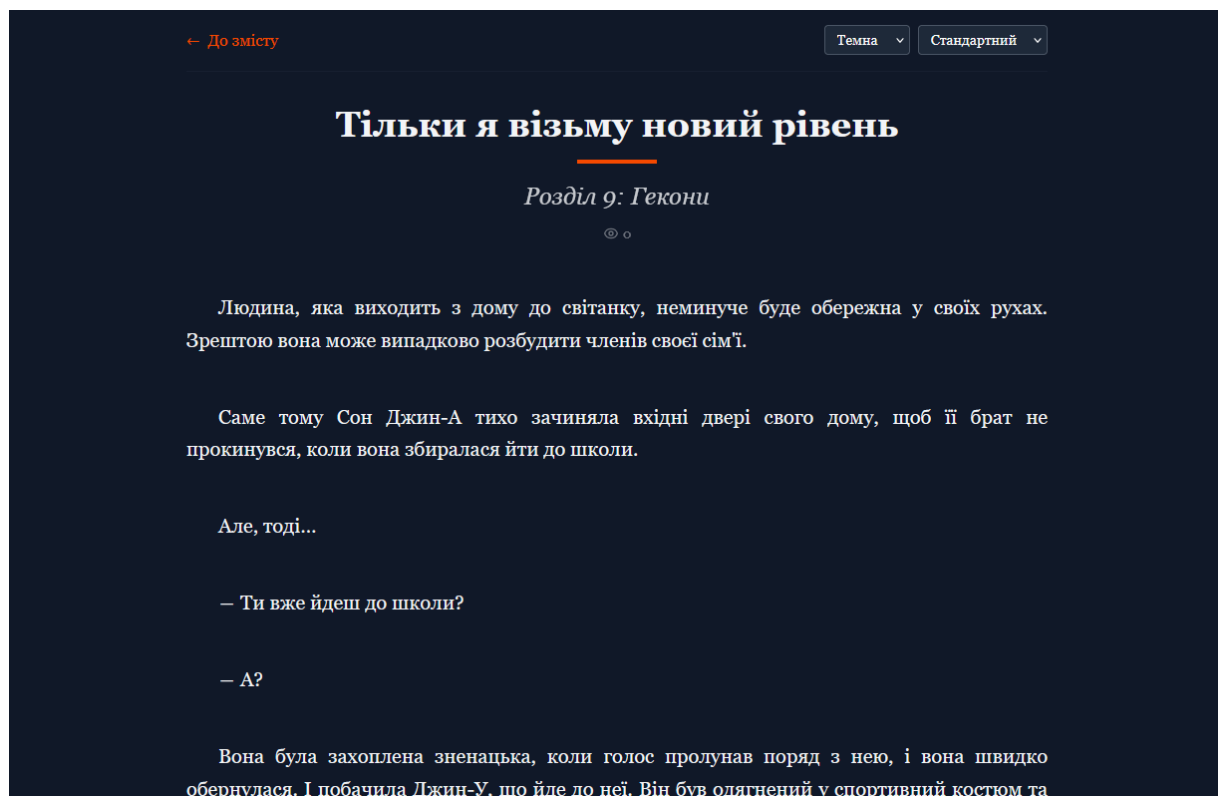


Рис. 3.29 Сторінка розділу

Також в кінці сторінки є список коментарів (рис. 3.30) та цього розділу та стрілка щоб швидко повернутися до верху сторінки.

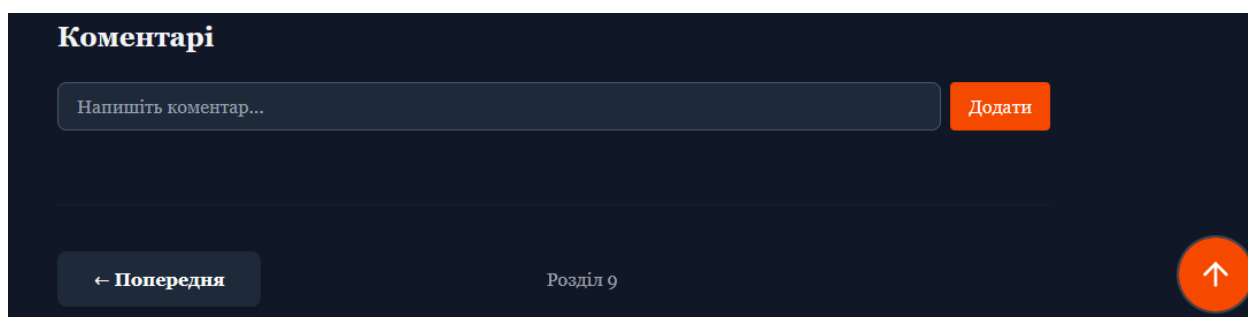


Рис. 3.30 Коментарі до розділу

Кожен авторизований користувач після натискання на своє ім'я та вибору свого профілю може увійти до нього (рис. 3.31). У самому профілі користувач може змінювати аватарку бачити свою пошту та id, бачити час реєстрації, кількості друзів, статть та пункт про себе.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		86

Також користувач може перейти в налаштування профілю, або опублікувати новий твір. Нижче наведено всі особисті переписки в конкретного читача

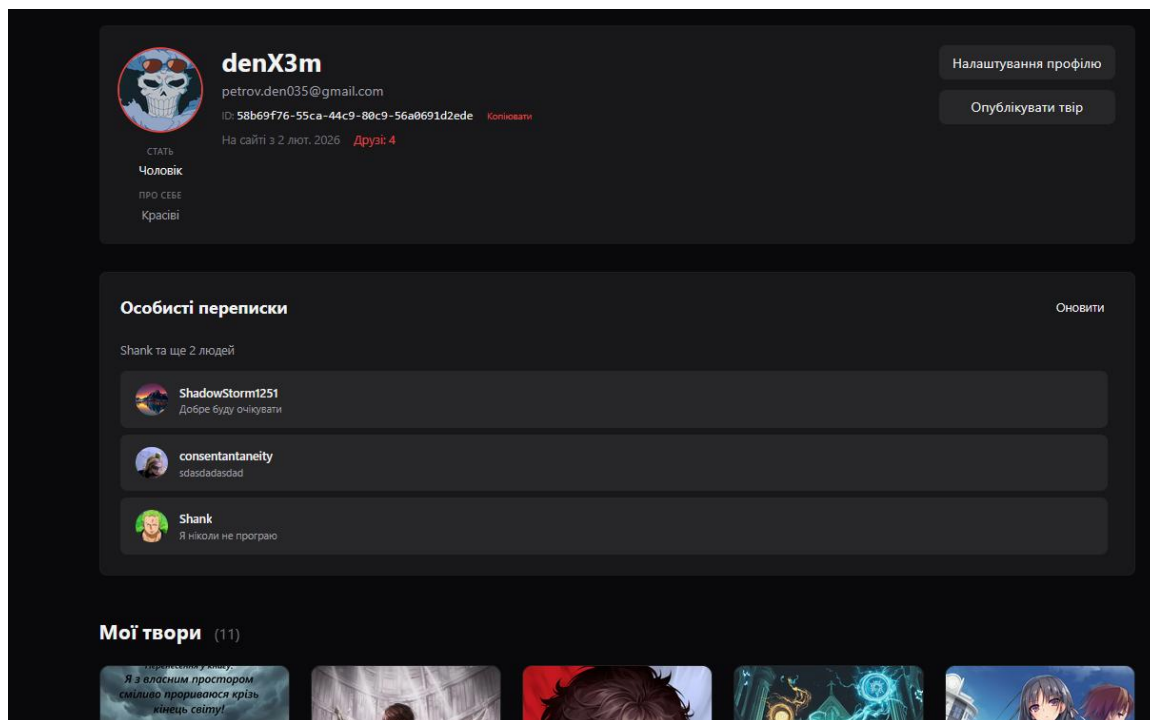


Рис. 3.31 Профіль користувача

У профілі при натисканні на друзі, користувач переноситься на сторінку (рис. 3.32) з якої може переглянути, всіх своїх друзів та перейти на сторінку особистих повідомлень або видалити їх із товаришів, всі заявки у друзі та відправлені йому запити показуються на окремих вкладках.

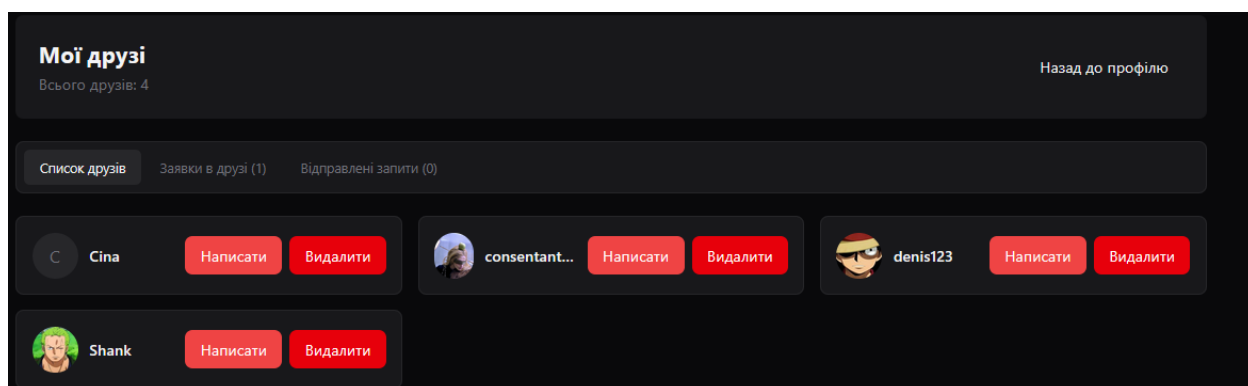


Рис. 3.32 Сторінка друзів

В профілі користувача можна перейти в особисті чати (рис. 3.33). У лівій частині екрану відображено блок керування переписками. У його верхній частині знаходиться рядок пошуку, який дозволяє знаходити потрібні чати

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

серед великої кількості, нижче наведений список усіх чатів з відображенням аватарки користувача, його імені та останнього повідомлення у цій переписці. Знизу лівої частини присутній кошик за допомогою якого можна видалити чат із користувачем

Права частина показує сам чат у верхній частині чату показано саме з яким користувачем зараз відбувається спілкування. Щоб розуміти коли було спілкування між людьми то реалізовано виведення дати у чаті коли було це спілкування. Повідомлення активного користувача підсвічуються червоним кольором його часов та аватаркою, у співрозмовника те саме крім відображення чорного фону на його повідомленні. Якщо користувач виділить своє повідомлення то він може його або видалити або редагувати, якщо він внесе якісь зміни то воно буде видно і його співрозмовнику.

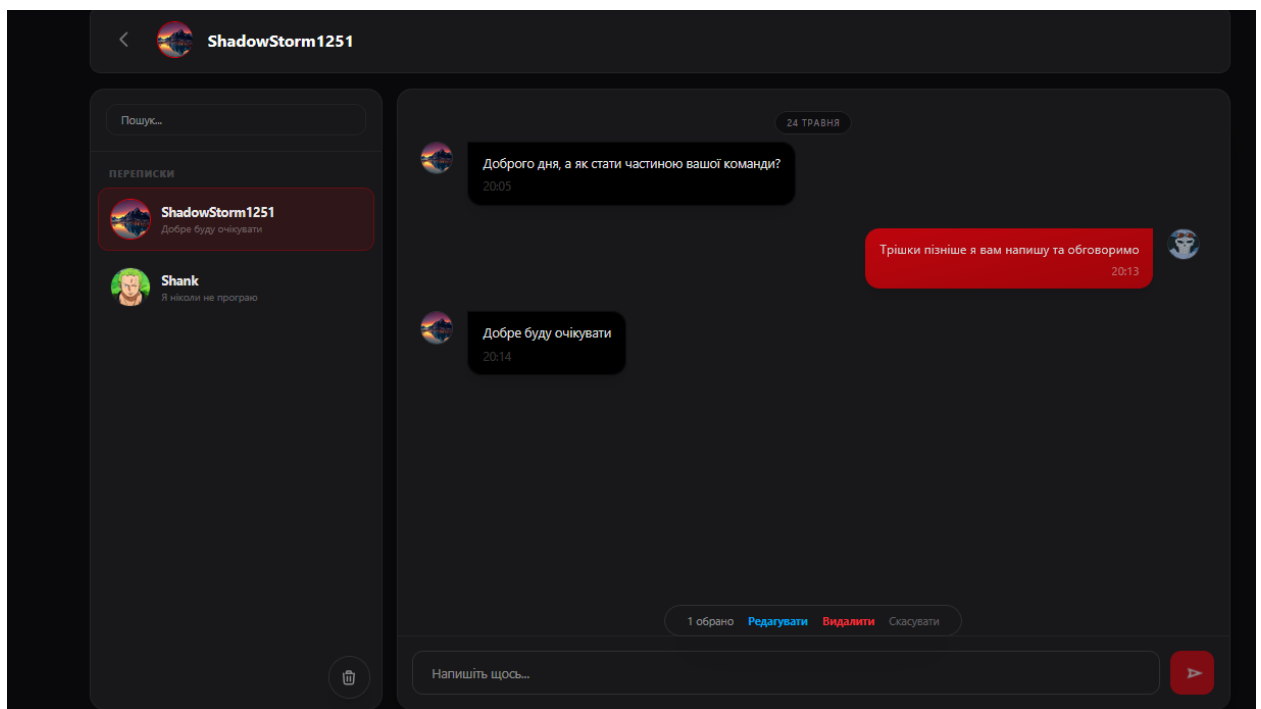


Рис. 3.33 Особисті чати

Користувач може зайти до свого списка творів (рис. 3.34) де збережені дані, які він читає та має стандартні списки, читаю, заплановано, завершено, припинено, улюблені та може створювати свої кастомні списки які він може створити до п'яти штук, та користувач може сортувати їх та розміщувати де завгодно між іншими списками, також є можливість змінювати відображення творів списком або плиткою та сортування по даті додавання (нові або старі)

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				88
Змн.	Арк.	№ докум.	Підпис	Дата		

та по назві (A-Z, Z-A) ще при натисканні кнопки редагувати користувач має можливість через стрілки змінювати свої списки місцями як йому буде зручно.

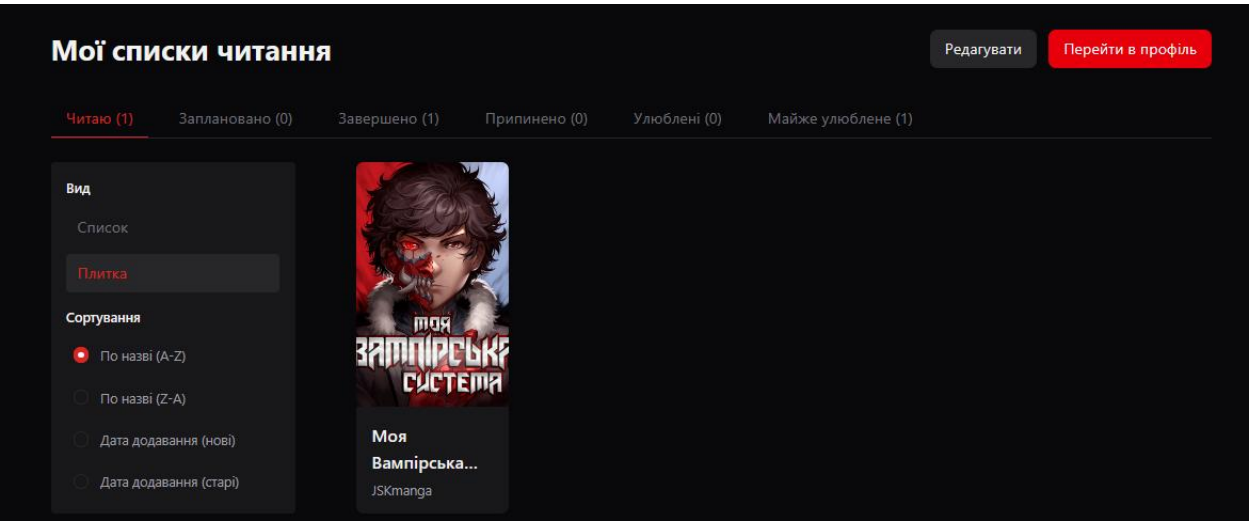


Рис. 3.34 Списки читання користувача

Також при перегляді чужого профілю (рис. 3.35) буде видна інформація про нього наприклад пошта, коли був зареєстрований, кількість його друзів та всі опубліковані ним твори та йому можна написати повідомлення, заблокувати (щоб не було видно його теми форуму та колекції) та перейти до списків цього читача.

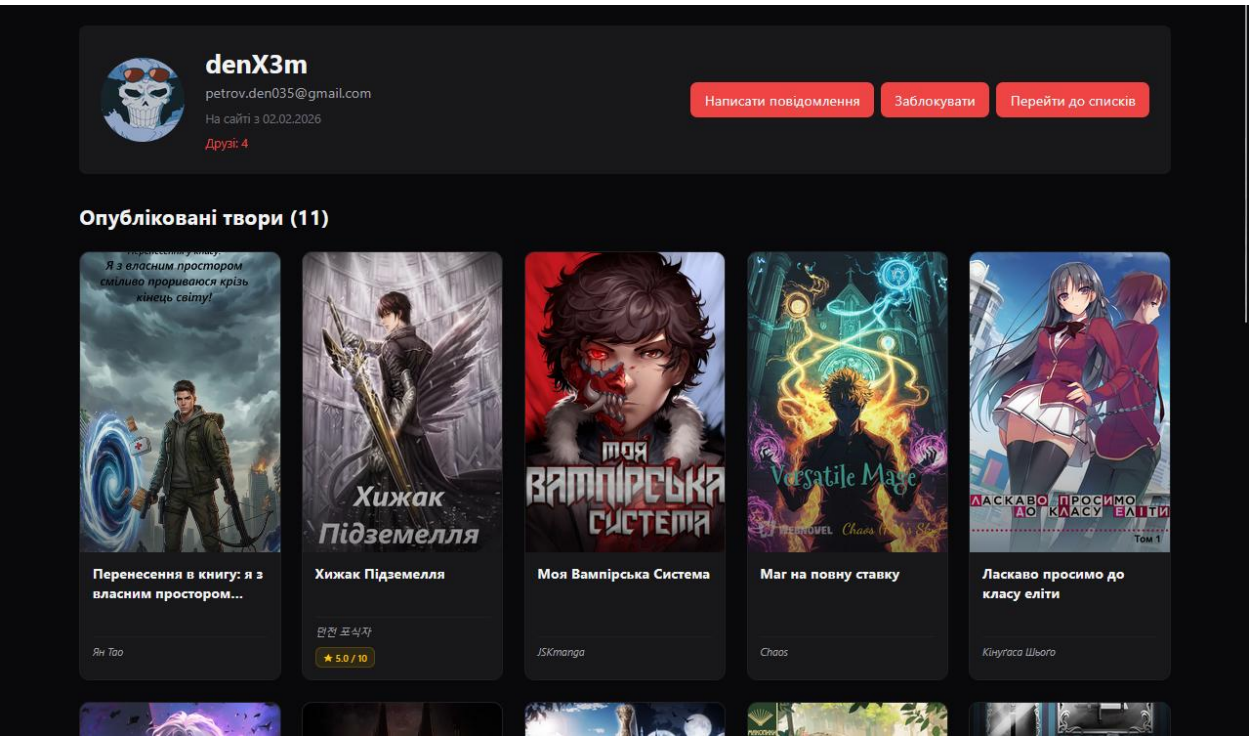


Рис. 3.35 Профіль іншого користувача

При перегляді списків іншого користувача (рис. 3.36) можна подивитися всі його твори та окремо перейти на кастомні списки.

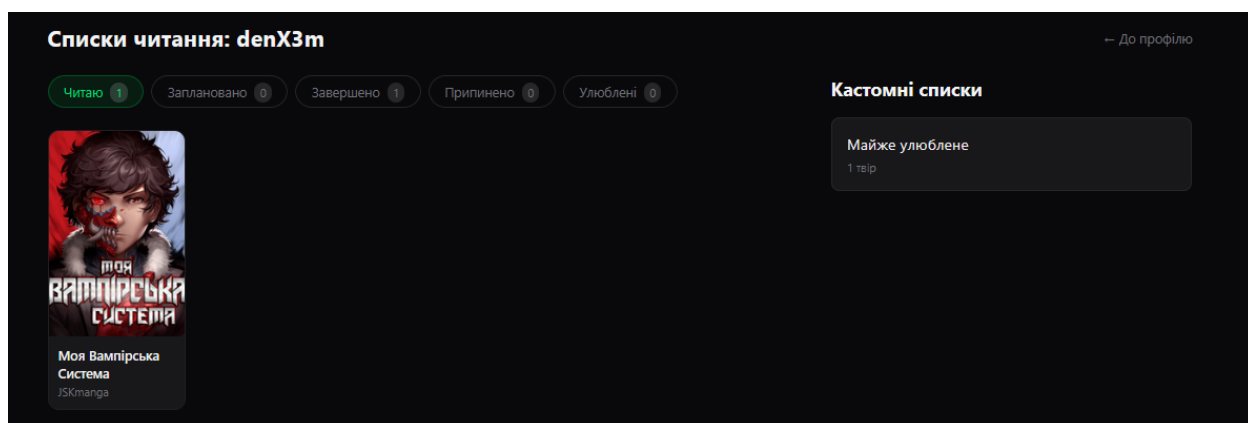


Рис. 3.36 Списки читання іншого користувача

Через профіль користувач може зайти в налаштування (рис. 3.37) та воно відбувається в пунктах, Профіль, Сповіщення, Приватність, Ігнор-лист, Фільтр контенту, Безпека і вхід та Платежі. У вкладці профіль користувач може змінити нікнейм, вибрати свою стат'я та написати про інформацію про себе яка буде видна усім користувач.

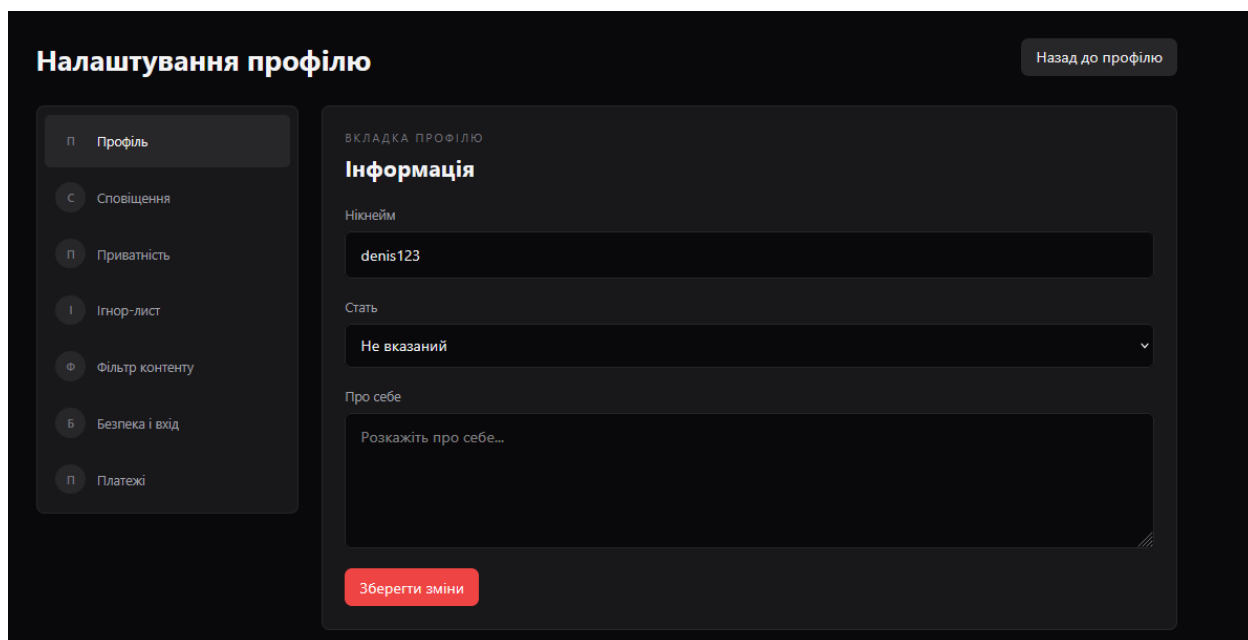


Рис. 3.37 Налаштування профілю

В налаштуванні сповіщень (рис. 3.38) у користувача є два блоки, перший “Загальні” у якому за допомогою чекбоксів користувач може вирішувати, чи хоче він отримувати повідомлення про зміну статусу твори на “Завершений”, а також увімкнути або вимкнути про нові заявки в друзі від

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

інших користувачів. У другому блоці користувач може обрати з яких його списків він хоче отримувати повідомлення, якщо він не захоче бачити повідомлення з якогось списку він просто зніме галочку біля його назви.

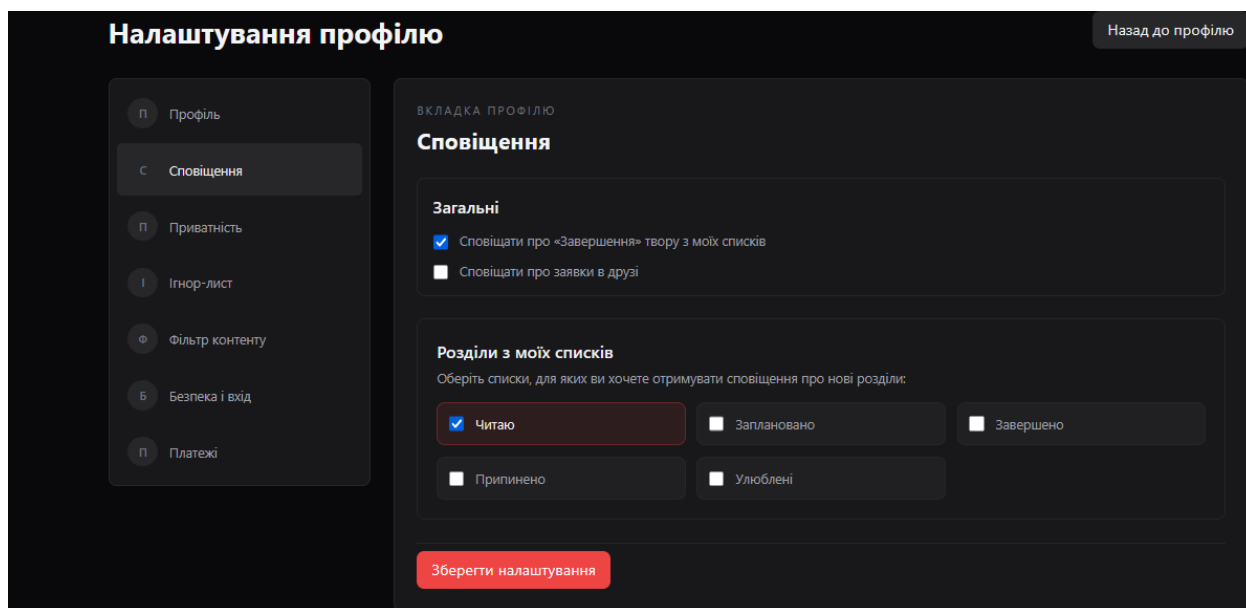


Рис. 3.38 Налаштування сповіщень

В налаштуванні приватності (рис. 3.39) користувач може вибрати відображення свого профілю іншим користувачам, а сам профіль може бути: публічним (коли його бачать всі), тільки для друзів, для всіх, крім ігнор-листа (ті хто заблоковані не можуть дивитися) та приватний.

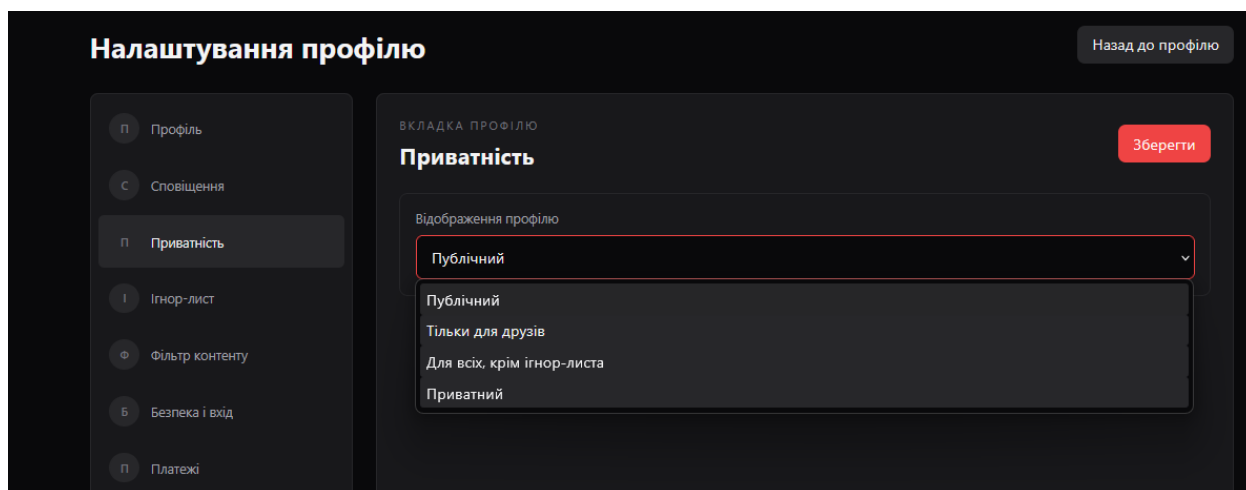


Рис. 3.39 Налаштування приватності

В ігнор-листі (рис. 3.40) користувач може додати інших людей щоб вони не бачили його коментарів, не могли писати вам повідомлення та не було видно їхнього контенту.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		91

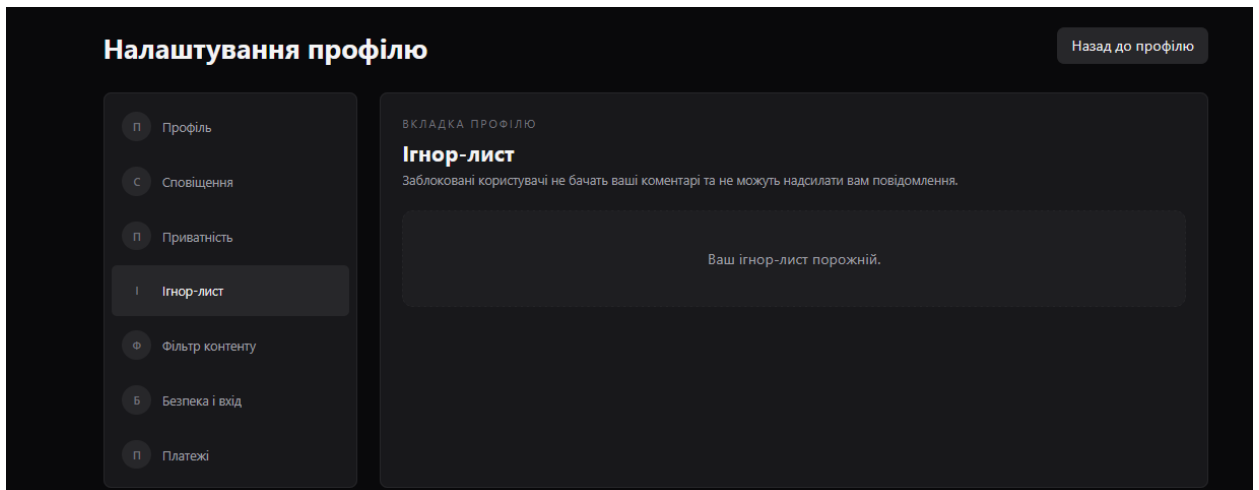


Рис. 3.40 Налаштування ігнор-листа

В фільтрах контенту (рис. 3.41) користувачу надається можливість редагувати стрічку каталогу виключаючи з неї небажані твори. Інтерфейс виглядає як сітка, яка містить перелік усіх жанрів на вебплатформі. Користувачу пропонується через чекбокс відмітити ті жанри, які йому не подобаються. Наприклад, як на скріншоті читач не зацікавлений у жанрі ”Школа”, він позначає цей пункт галочкою. Після зберігання цього налаштування у каталозі не буде показуватися всі твори з цим жанром тільки для цього користувача.

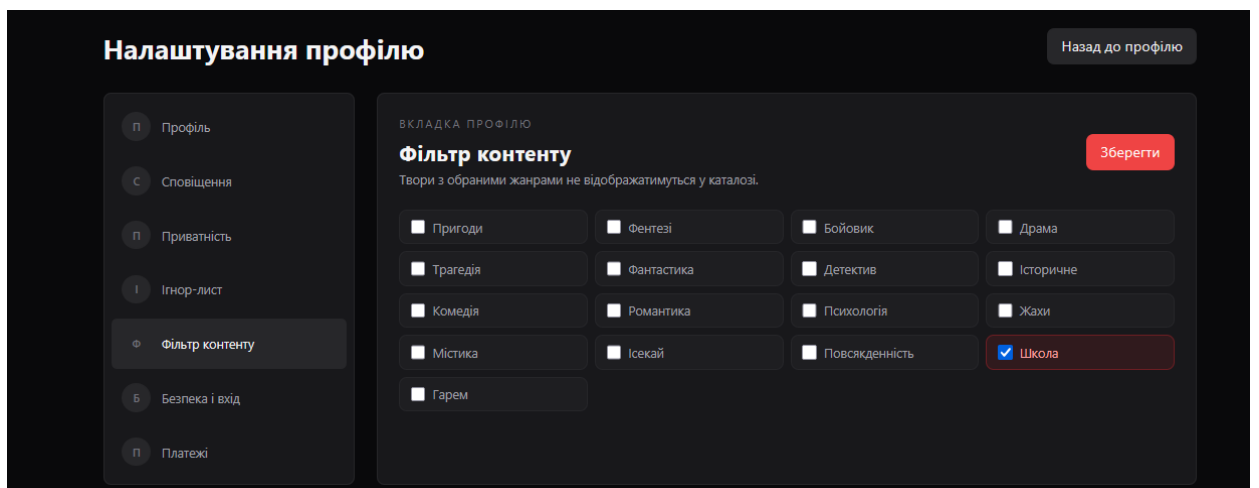


Рис. 3.41 Налаштування ігнор-листа

При налаштуванні безпеки та входу (рис. 3.42) у користувача є блок зміни паролю в якому для зміни йому потрібно написати поточний пароль та вгадати новий який буде складатися мінімум із шести символів, якщо все буде зроблено правильно то у користувача буде створено новий пароль. У

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		92

блоці прив’язані акаунти, якщо користувач просто зареєструвався то він може прив’язати свій гугл акаунт для того щоб не загубити свій прогрес, якщо користувач зайшов відразу за допомогою то він буде тут вказаним.

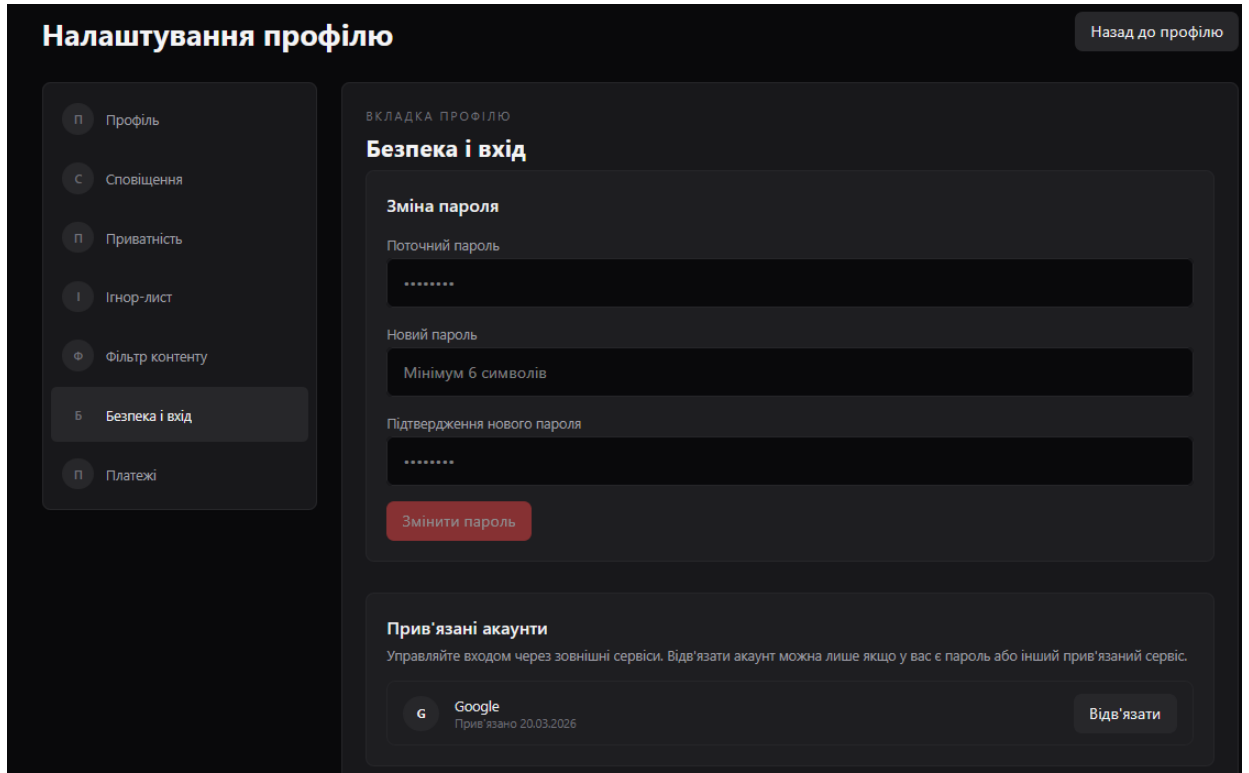


Рис. 3.42 Налаштування безпеки

У вкладці платежі (рис. 3.43) користувач може подивитися список всіх платних розділів які користувач придбав за увесь час користування вебплатформою.

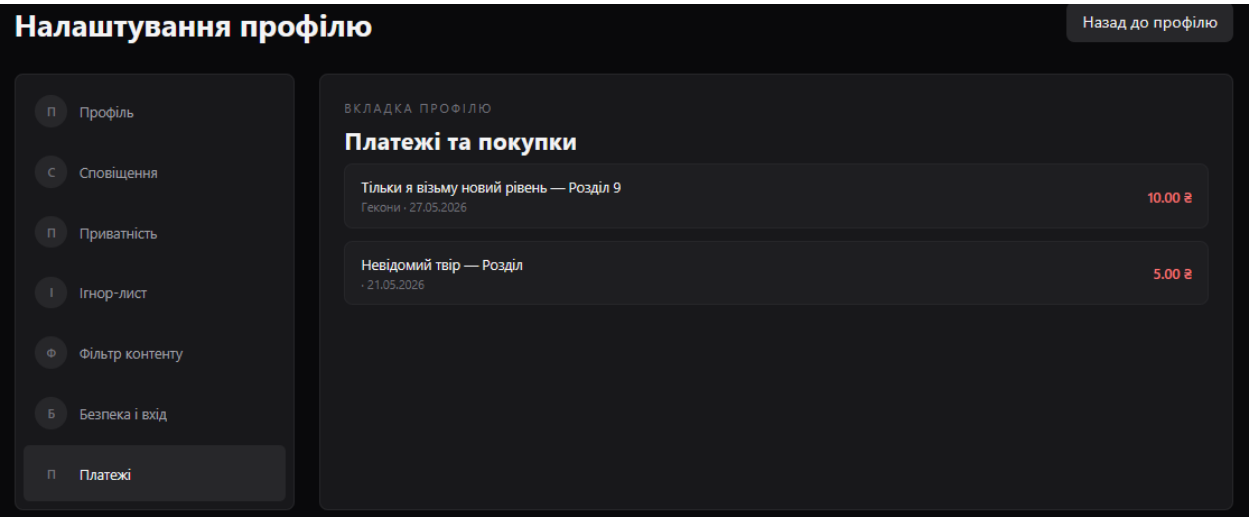


Рис. 3.43 Список платежів

Також в футері є кнопка FAQ, щоб перейти на список поширених запитань та ознайомитися з правилами вебплатформи (рис. 3.44).

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

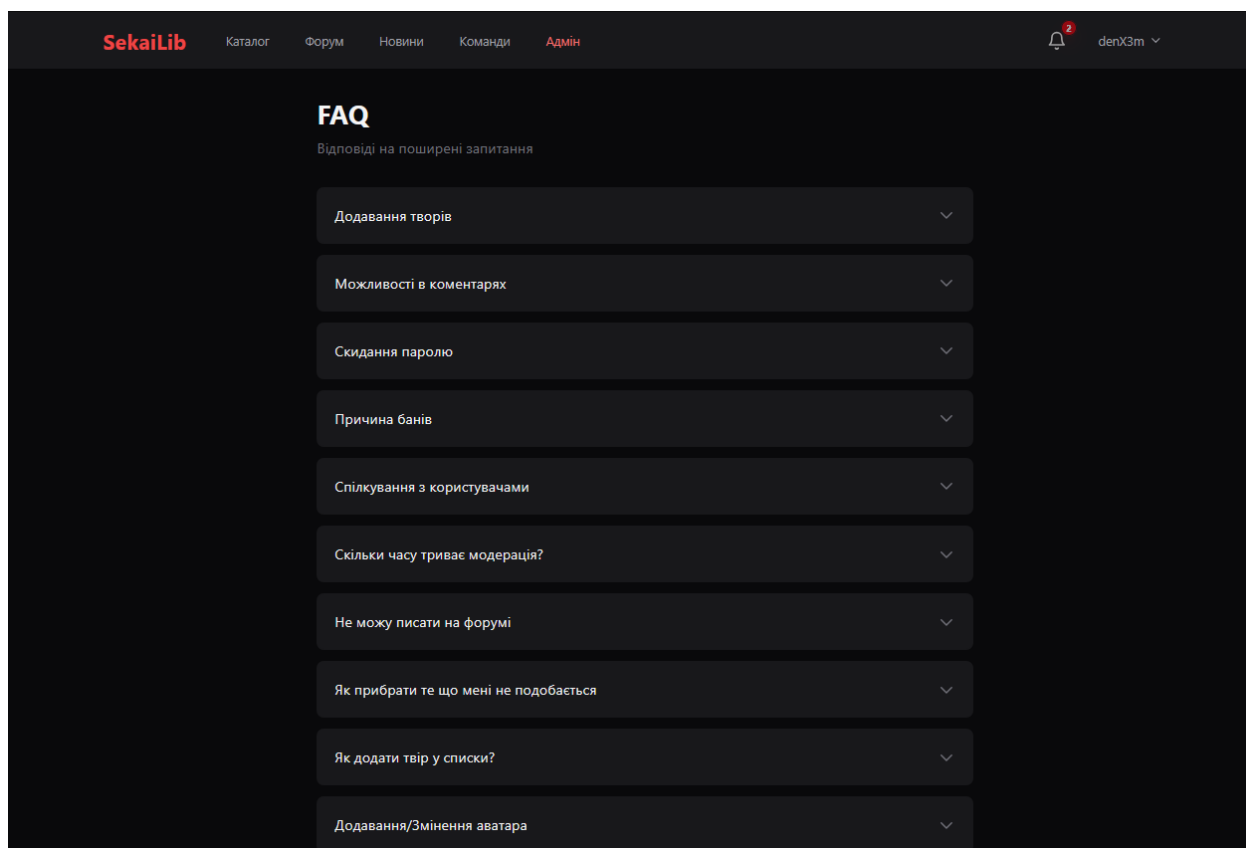


Рис. 3.44 Відображення FAQ

Для керування вебплатформою у адміністратора є окреме меню (рис. 3.45) у вигляді інформаційної дошки з інтерактивними об'єктами де кожна картка відповідає за конкретну систему програми та має свою назву та іконку, щоб адміністратор міг легше орієнтуватися, а у вкладках він може керувати творами, жанрами, банами, скаргами, новинами, FAQ, та платежами.

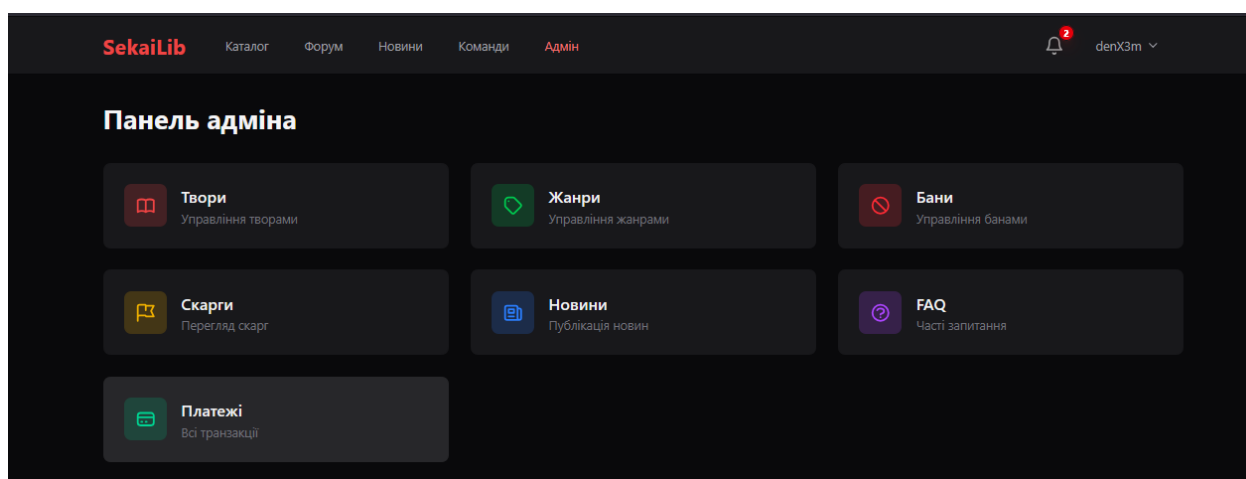


Рис. 3.45 Панель адміна

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

У вкладці “Керування творами” (рис. 3.46) адміністратор має повний контроль над творами на вебплатформі. Сторінка має вигляд таблиці, кожен її рядок показує окремий твір, та містить його метадані, обкладинку, назву, автора та статусу. Для управління ними було додано кнопку “Додати твір” в якому користувач попадає на форму для додавання творів, також присутня кнопка “Дії” за допомогою якої користувач може редагувати або видаляти твір.

твір	автор	статус	дії
Перенесення в книгу: в з власним простором смисло прориваюся крізь кінець сапу!	Ян Тао	0	Редагувати Видалити
Хижак Підземелля	현전 포식자	1	Редагувати Видалити
Моя Вампірська Система	JSKlampa	0	Редагувати Видалити
Маг на повну ставку	Chaos	0	Редагувати Видалити
Ласкаво просимо до класу еліти	Кинугаса Шього	0	Редагувати Видалити
Очіма Лиходія	Touch_me2	0	Редагувати Видалити
Володар Темниці	Ай Циньшуй Де Удней	3	Редагувати Видалити
Я не такий талановитий	만픽 (Dempfy)	0	Редагувати Видалити

Рис. 3.46 Керування творами

Такий самий підхід реалізовано і у вкладці “Управління жанрами” (рис. 3.47), який відповідає за категоризацію на вебплатформі. На самій сторінці зображений список усіх жанрів. Адміністратор має можливість додавати нові жанри після натискання на кнопку його буде відправляти на потрібну форму і у нього є можливість редагувати або видаляти некоректні жанри

назва	дії
Пригоди	Редагувати Видалити
Фентезі	Редагувати Видалити
Бойовик	Редагувати Видалити
Драма	Редагувати Видалити
Трагедія	Редагувати Видалити
Фантастика	Редагувати Видалити
Детектив	Редагувати Видалити
Історичне	Редагувати Видалити
Комедія	Редагувати Видалити
Романтика	Редагувати Видалити
Психологія	Редагувати Видалити

Рис. 3.47 Керування жанрами

У вкладці банів користувачів (рис. 3.48) адміністратор, якщо замітив якесь порушення, забанити користувача ввівши його id, написавши термін и причину блокування а також перегляд всіх активних банів.

Бани користувачів

Заблокувати користувача

ID користувача

UUID користувача

Термін дії (необов'язково)

дд.мм.гггг, --:--

Причина

Причина блокування

Заблокувати

Активні бани (0)

Немає активних банів

Рис. 3.48 Сторінка банів користувачів

На сторінці скарг (рис. 3.49) адміністратор переглядає усі скарги та вирішує що з цим робити і якщо потрібно може забанити користувачів із цієї вкладки.

Скарги

Розглянуто

Спам

Від: consentantaneity · ID цілі: 21a4983b... · 22.05.2026

Деталі

Видалити

Відхилено

Спам

Від: consentantaneity · ID цілі: 21a4983b... · 22.05.2026

Деталі

Видалити

Розглянуто

Образливий контент

не норм

Від: consentantaneity · ID цілі: 21a4983b... · 22.05.2026

Деталі

Видалити

Рис. 3.49 Сторінка скарг

На сторінці новин (рис. 3.50) адміністратор може створювати та редагувати новини для сповіщення користувачів про технічні роботи чи оновлення.



Рис. 3.50 Сторінка новин адміністратора

Присутня сторінка FAQ (рис. 3.51), яка виглядає як інформаційні картки, на яких є номер запису, заголовок, та короткий перегляд тексту самої відповіді. У правому верхньому куті є кнопка “Нове питання”, яка відправляє адміністратора на форму створення довідкового матеріалу. Якщо якісь матеріал було введено невірно або його взагалі не мало бути адміністратор може їх або редагувати або видаляти.

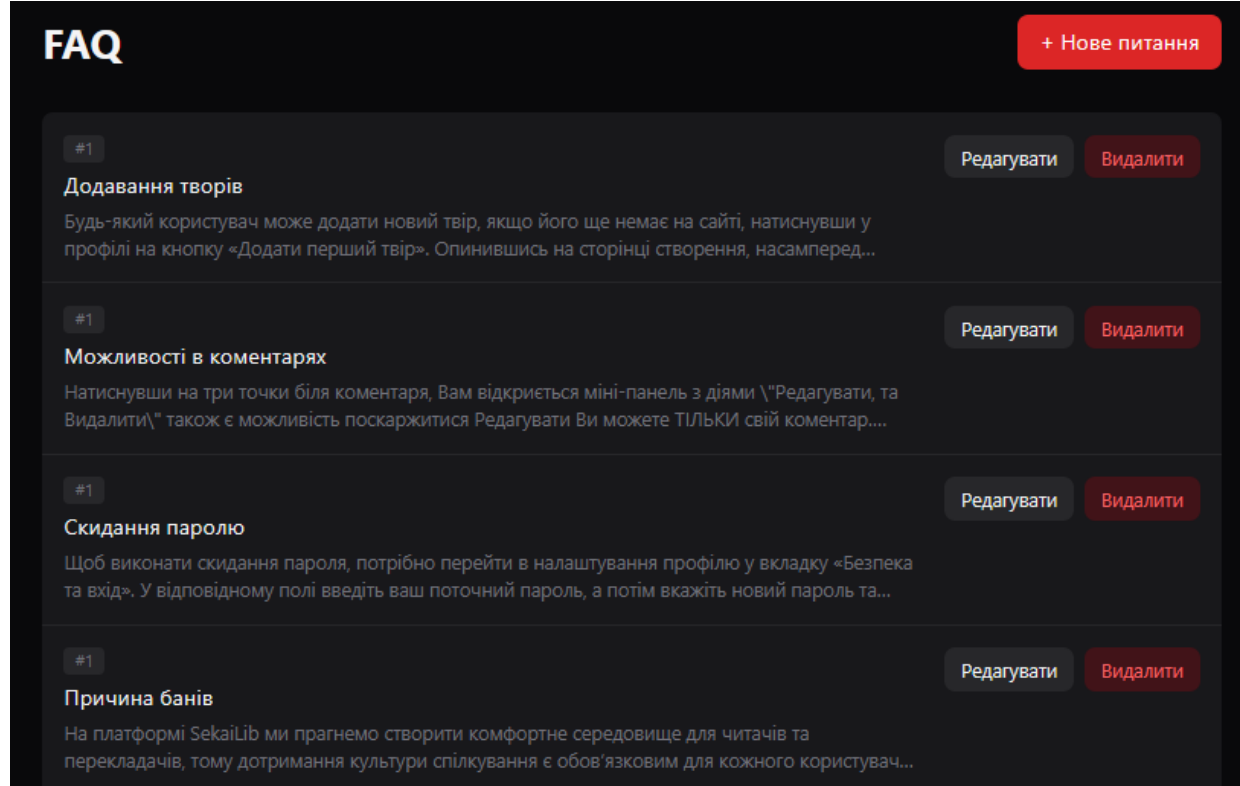


Рис. 3.51 Сторінка редагування FAQ

На сторінці платежів (рис. 3.52) адміністратор переглядає всі проведені транзакції та він бачить скільки успішних, очікуваних та кількість отриманої

виручки, також може знайти необхідний платіж по id або імені користувача та відсортувати за статусом.

Користувач	Розділ	Сума	Статус	Order ID	Дата
denX3m	Тільки я візьму новий ... Розд. 9 — Гекони	10.00 UAH	Тест (sandbox)	3c1ba574883341068e2c... LP: 2862231923	27.05.26, 22:18 27.05.26, 22:21
denX3m	—	5.00 UAH	Очікує	0fe50ea79bf24675b36d...	25.05.26, 16:56
denX3m	—	5.00 UAH	Тест (sandbox)	b158c1e03cf443589e01... LP: 2858428236	21.05.26, 13:09 21.05.26, 13:09
denX3m	—	5.00 UAH	Помилка LiqPay: error	5b1033a914b94442afc4...	21.05.26, 13:01 21.05.26, 13:01
denX3m	—	5.00 UAH	Помилка LiqPay: error	790c071a9c224229b356...	20.05.26, 22:04 20.05.26, 22:04
denX3m	—	5.00 UAH	Помилка LiqPay: error	3e4bd108cf5346818323...	20.05.26, 18:25 20.05.26, 18:25
denX3m	—	5.00 UAH	Помилка LiqPay: error	384608a541f746558061...	20.05.26, 16:04 20.05.26, 16:04

Рис. 3.52 Сторінка транзакцій

У модератора також є своя панель (рис. 3.53), у якій він може дивитися чергу перевірок, список скарг та банів, у нього є своя сторінка де він може шукати користувачів для виконувannya дій, журнал для перегляду всіх своїх дій та стоп-слова для автомодерації.

МОДЕРАЦІЯ		Панель модератора	
Дашборд	1 Очікують перевірки	0 Відкриті скарги	0 Попереджень сьогодні
Черга	0 Активні бані		
Скарги			
Користувачі			
Журнал			
Стоп-слова			
Швидкі дії			
Переглянути чергу Стоп-слова Журнал дій			

Рис. 3.53 Панель модератора

В черзі модерації (рис. 3.54) модератор бачить додані твори або розділи, від користувачів які він може схвалити або відхилити якщо там присутнє порушення правил вебплатформи.

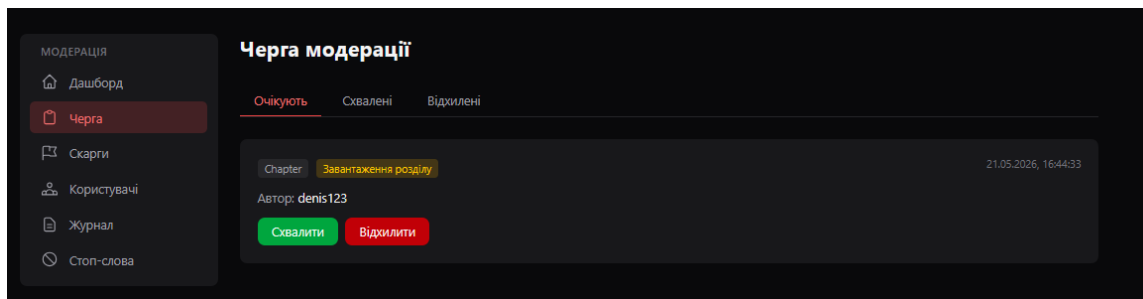


Рис. 3.54 Черга модераторів

У вкладці “Скарги” (рис. 3.55) є список всіх звернень користувачів. Кожна скарга оформлена у вигляді блоку що містить усю інформацію необхідну для прийняття рішення. В самому блоці вказується до якої частини була скарга, статус перегляду модератором та нижче наведено від якого читача було скарга і на якого, ще нижче наведена причина та присутнє посилання при натисканні на яке модератор може подивитися саме повідомлення на яке відправлена скарга.

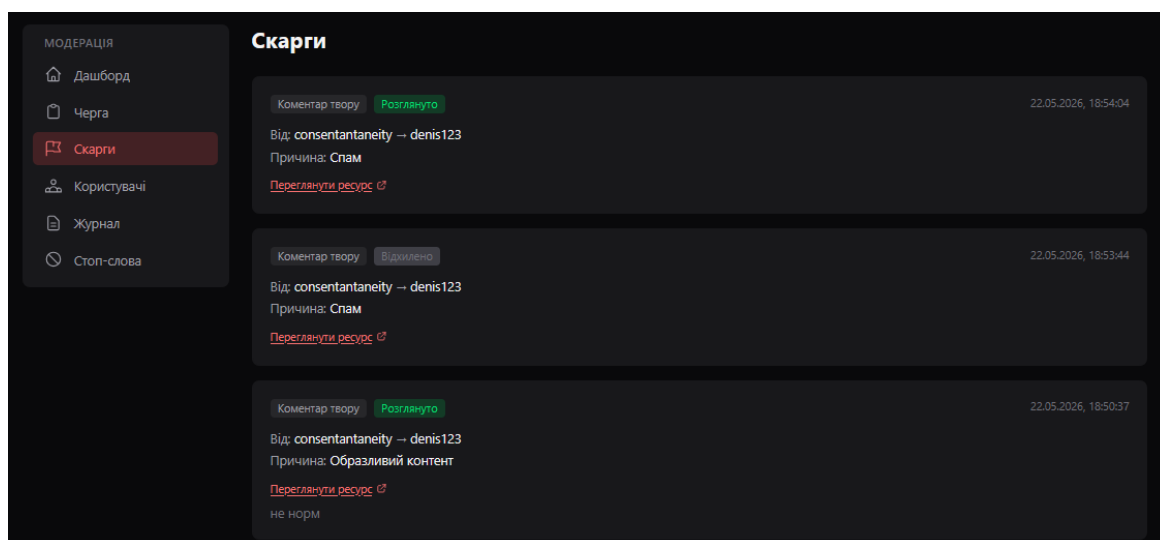


Рис. 3.55 Сторінка скарг модератора

У модератора на вкладці “Користувачі” (рис. 3.56) є можливість пошуку читачів та відправляти їм попередження за некоректні дії, заблокувати або подивитися список всіх попереджень наданих цьому користувачу.

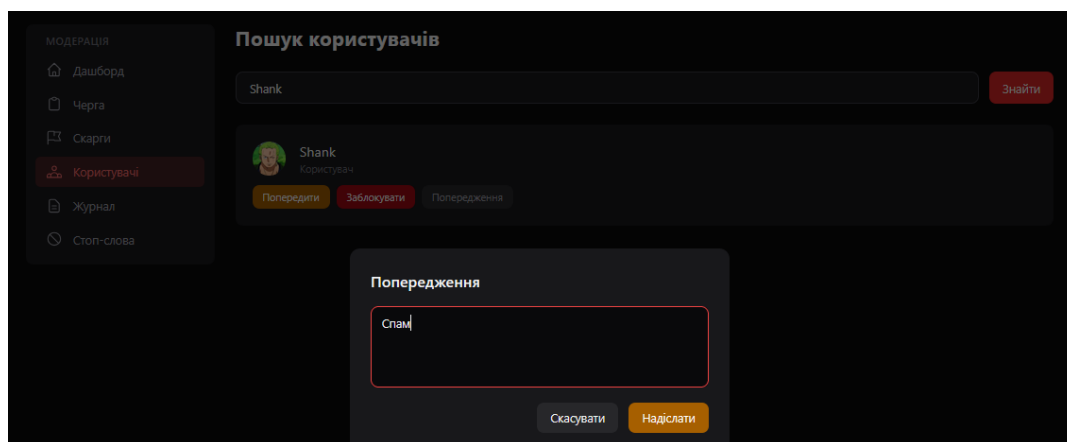


Рис. 3.56 Пошук користувачів для видачі попереджень

При заході модератора на вкладку “Журнал дій” (рис. 3.57) цей розділ виконує функцію переглядання та класифікації усіх дій виконаних модератором, зліва кожної дії буде відображатися яка дія була виконана а справа від цього детальніший опис над ким або чим було проведено дії та час коли це відбувалося.

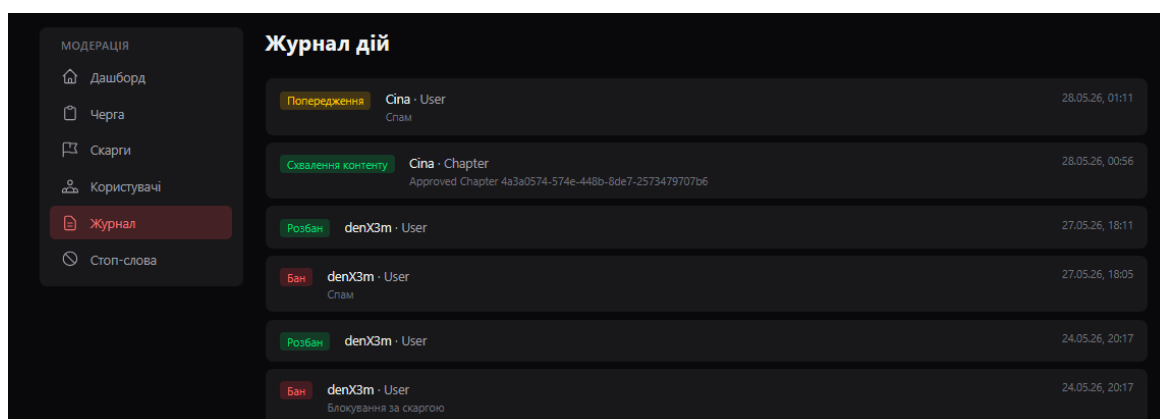


Рис. 3.57 Вигляд журналу дій

На вебплатформі присутня автомодерація (рис. 3.58) через стоп-слова, якщо в коментарях, трехах або рецензіях будуть слова із цього списку то вони не будуть загальними для користувачів та будуть відправлені на модерацію.

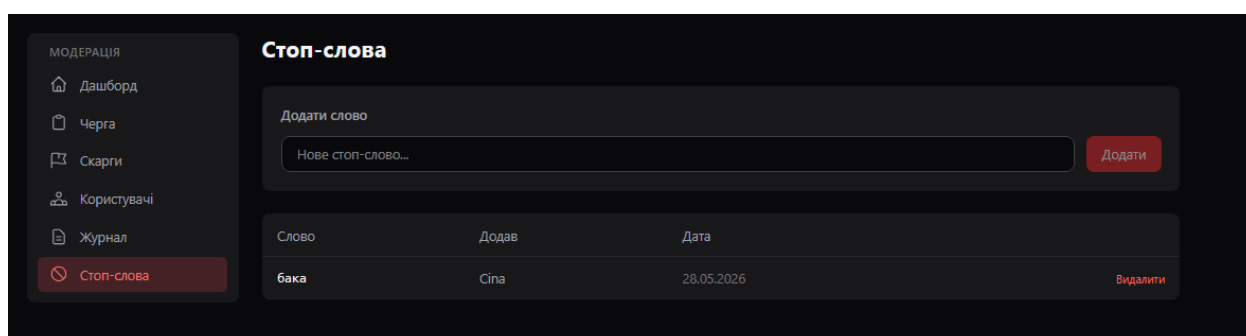


Рис. 3.58 Автомодація

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				100
Змн.	Арк.	№ докум.	Підпис	Дата		

Таким чином, у цьому розділі було спроектовано та зроблено клієнтську й серверну частини вебплатформи SekaiLib. Зроблені функції забезпечують фільтри контенту, персоналізацію сповіщень, а також модулі комунікації у вигляді форуму та приватних чатів.

3.3. Тестування вебплатформи

Без тестування жодна вебплатформа нормально не буде працювати, тому після написання основного коду, відразу була проведена за перевірка. Це потрібно в першу чергу щоб виловити всі неочевидні баги у функціоналі. І звісно, треба було залізно переконатися, що готова програма реально робить саме те, що ми вимагали від неї ще на самому старті розробки.

Для реалізації контролю якості тестування було розділено на кілька етапів, які будуть пов'язні між собою. На стороні сервера перевірялося наскільки швидко відбувається обробка запитів до бази даних, коректність збереження даних та надійність авторизації користувачів. Потім було почато тестування клієнтського інтерфейсу. Оскільки у системі є можливість на взаємодію з читачами, акцент робився на перевірці адаптивності дизайну, використання різних браузерів та правильності роботи всіх інтерактивних елементів. Було проведено ручні перевірки, щоб гарантувати, що читання розділів, робота фільтрів каталогу, публікація коментарів та відображення сповіщень функціонують без проблем та зручно як на великих моніторах, так і на екранах мобільних пристроїв.

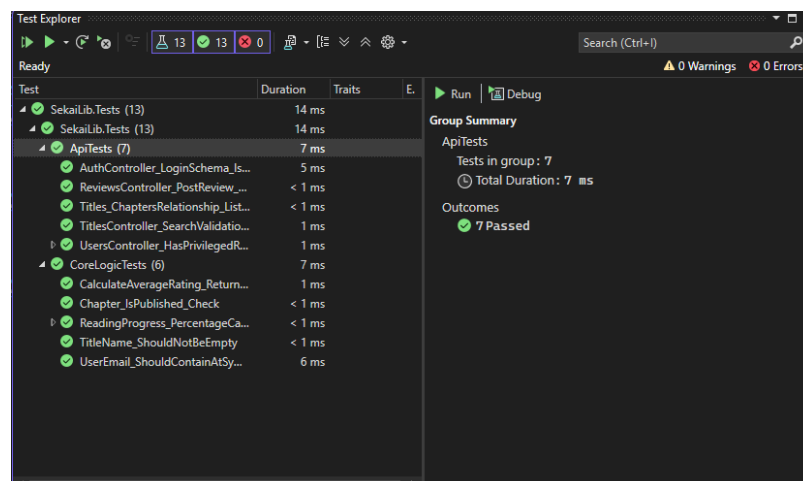


Рис. 3.59 Результат тестування API та бізнес логіки

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

Для надання найвищої якості програмного продукту, процес автоматизованого тестування було розбито на окремі перевірки, кожна з яких відповідає за конкретний момент роботи системи. Першим етапом стала перевірка контролера, що керує пошуковою системою каталогу. Було розроблено тест, який гарантує, що API правильно обробляє вхідні запити користувачів, крім цього перевіряє мінімально допустиму довжину введеного тексту. Це необхідно для запобігання сильному навантаженню на базу даних через занадто короткі запити.

Наступним кроком було протестовано модуль взаємодії між користувачами - контролер публікації рецензій. Автоматизована перевірка контролює цілісність даних, що надсилаються на сервер, алгоритм переконується, що текстове поле відгуку не залишається порожнім, а виставлена читачем оцінка обов'язково знаходиться у встановленому математичному діапазоні (від одного до десяти балів).

Також було звернено увагу на тестування зовнішнього інтерфейсу, було приділено відповідну увагу на безпеці та механізму авторизації сайту. Для цього реалізовано перевірку правильності формування алгоритму входу в обліковий запис. Програма демонструє, що введена електронна адреса проходить синтаксичну валідацію містить обов'язковий спеціальний символ, а пароль відповідає мінімальним вимогам безпеки щодо кількості символів.

Щоб захистити адміністративну панель та інструментів модерації було розроблено тест перевірки ролей користувачів. Ця перевірка відображає запити від різних користувачів із різними ролями і гарантує, що доступ до незвичайних функцій надається тільки обліковим записам із визначеними ролями «Адміністратор» або «Модератор», а запити від звичайних читачів блокуються.

Останній етап тестування на бекенді перевіряє цілісність зв'язків між сутностями бази даних. Коли система отримує запит на потрібний твір, вона автоматично без помилок підтягує повний список його розділів. Порожні результати тут просто неможливі для існуючих зв'язків.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		102

Другий блок тестування перевіряє внутрішню бізнес-логіку вебплатформи. Одним із найважливіших тестувань є алгоритми розрахунку середнього рейтингу твору. Під час тестування система створює масив оцінок і перевіряє, чи співпадає результат обчислення середнього арифметичного значення з математичним показником.

Механізм відстеження прогресу читання був так само протестований. Спеціальний модульний тест перевіряє правильність розрахунку відсотка прочитаного розділу. Алгоритм порівнює наскільки прогорнута сторінку з розділом із його загальним обсягом, тим самим гарантуючи, що функція "Продовжити читання" відображає на головній сторінці точні дані.

Окремими тестами було перевірено валідацію доменних моделей на рівні ядра системи. Ці перевірки дають гарантію, що жоден твір не може бути зареєстрований без вказання його назви, а статуси публікації для кожного окремого розділу набувають коректних значень.

Такий підхід до модульного тестування кожної окремої функції дозволив виявити потенційні вразливості ще на етапі розробки, гарантуючи абсолютну надійність, безпеку та стабільність функціонування готової вебплатформи [19, 30].

Для реалізації адаптивності обійшлися без зовнішніх сервісів. Просто відкрив стандартні інструменти розробника прямо в браузері. Це дозволило швидко прогнати всі ключові сторінки вебплатформи через абсолютно різні розміри екранів.

А головною метою було побачити, чи всі наші кнопки, блоки з творами та довгі тексти залишаються на своїх місцях. Кожна сторінка цілком адекватно перегруповує елементи та нормально підлаштовується під потрібну роздільну здатність пристрою.

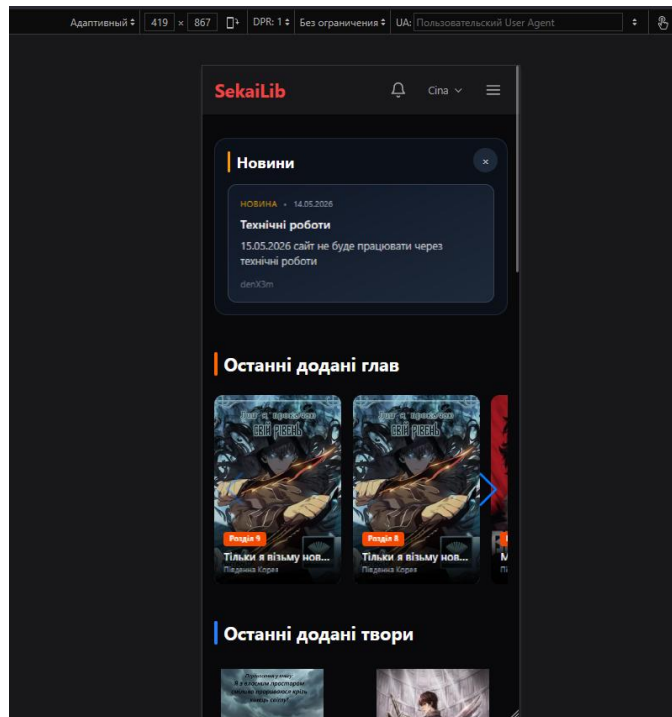


Рис. 3.60 Вигляд адаптивності на телефоні

Висновки до третього розділу

У третьому розділі було повністю розібрано реальний процес розгортання та тестування вебплатформи SekaiLib. Спочатку побудована детальну діаграму, там видно всю взаємодію між браузером користувача, докер-контейнерами на бекенді та базою даних. Потім покроково описано та показано на скріншотах весь доступний функціонал як для користувачів так і для адміністратора і модератора хостингу на базі серверів Hetzner, каталоги творів, модуль читання книжок, блок сповіщень та особисті профілі.

Тестування було виконано комплексно, серверну частину перевірена через автоматизовані юніт-тести в Test Explorer. Перевірено як роботу самих контролерів API, так і критичну бізнес логіку. На фронтенді проведено аудит продуктивності та чистоти коду. А щоб перевірити адаптивність верстки, просто було відкрито вбудовані інструменти розробника в браузері. Так можна переконатися, що інтерфейс SekaiLib чудово підлаштовується під будь-які екрани мобільних телефонів чи планшетів.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				104
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У процесі виконання випускної кваліфікаційної роботи бакалавра було проведено дослідження та розроблено Вебплатформа для читання електронних книжок» під назвою SekaiLib. Головною метою першого розділу був детальний аналіз існуючих конкурентів у галузі електронних бібліотек та вебплатформ для читання книжок. Це дозволило об'єктивно порівняти їхні сильні та слабкі сторони з моєю вебплатформою.

Під час виконання завдань було вивчено найкращі підходи до організації контенту, роботи з командами перекладачів та соціальної взаємодії читачів. На основі цього аналізу було сформульовано чіткі функціональні та нефункціональні вимоги до системи, що розроблялася.

У роботі описано архітектурне проектування за принципами Clean Architecture. Було визначено ключові сценарії використання, побудовано об'єктно-орієнтовану модель із застосуванням .NET 8 та React і спроектовано надійну логічну структуру бази даних на основі PostgreSQL.

Наостанок, було проаналізовано ключові алгоритми вебплатформи, такі як збереження прогресу читання. Показано успішне застосування підходу модульного моноліту з розмежуванням відповідальності між шарами системи Domain, Application, Infrastructure. Отримані результати підтверджують правильність обраних архітектурних та технологічних рішень і слугують надійною базою для успішного завершення випускної кваліфікаційної роботи бакалавра.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		105

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Петров Д.В., Єфремов Ю.М. ВЕБПЛАТФОРМА ДЛЯ ЧИТАННЯ РАНОБЕ. Тези доповідей XVI Міжнародної науково-технічної конференції Інформаційно-комп'ютерні технології - 2026, 02-03 квітня 2026 року. Житомир: «Житомирська політехніка», 2026.
2. Петров Д.В., Єфремов Ю.М. АРХІТЕКТУРА СИСТЕМИ ПЕРСОНАЛІЗАЦІЇ В ВЕБПЛАТФОРМАХ ДЛЯ ЧИТАННЯ РАНОБЕ. Тези доповідей XVI Міжнародної науково-технічної конференції Інформаційно-комп'ютерні технології - 2026, 02-03 квітня 2026 року. Житомир: «Житомирська політехніка», 2026.
3. ASP.NET Core, .NET 8 [Електронний ресурс] / Microsoft. - URL: <https://learn.microsoft.com/dotnet/> (дата звернення: 14.02.2026).
4. Мартін Р. Чиста архітектура. Мистецтво розроблення програмного забезпечення / Роберт Мартін; пер. з англ. - Харків: Фабула, 2019. - 368 с.
5. Троелсен А., Джепікс Ф. С# 10 і платформа .NET 6. Сучасна кросплатформна розробка / Ендрю Троелсен, Філіп Джепікс. - Київ : Діалектика, 2022. - 1328 с.
6. Entity Framework Core [Електронний ресурс] / Microsoft. - URL: <https://learn.microsoft.com/ef/core/> (дата звернення: 09.03.2026).
7. ASP.NET Core SignalR [Електронний ресурс] / Microsoft. - URL: <https://learn.microsoft.com/aspnet/core/signalr> (дата звернення: 23.04.2026).
8. Фаулер М. Архітектура корпоративних програмних додатків / Мартін Фаулер. - Київ: Діалектика, 2020. - 544 с.
9. React - A JavaScript library for building user interfaces [Електронний ресурс] / Meta Platforms, Inc. - URL: <https://react.dev/> (дата звернення: 10.03.2026).
10. Ньюмен С. Створення мікросервісів. Проектування дрібних систем / Сем Ньюмен; пер. з англ. - Київ: Наш Формат, 2021. - 336 с.

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ПЗ	Арк.
		Єфремов Ю.М.				106
Змн.	Арк.	№ докум.	Підпис	Дата		

11. Соммервіль І. Інженерія програмного забезпечення / Іан Соммервіль. - 10-те вид. - Київ: Діалектика, 2019. - 688 с.
12. Vite [Електронний ресурс] / Evan You. - URL: <https://vitejs.dev/> (дата звернення: 20.02.2026).
13. Tailwind CSS [Електронний ресурс] / Tailwind Labs. - URL: <https://tailwindcss.com/> (дата звернення: 19.03.2026).
14. Фрімен А. React для професіоналів / Адам Фрімен. - Київ: Діалектика, 2022. - 800 с.
15. TypeScript [Електронний ресурс] / Microsoft. - URL: <https://www.typescriptlang.org/> (дата звернення: 08.02.2026).
16. Ріхтер Д. CLR via C#. Програмування на платформі Microsoft .NET Framework 4.5 мовою C# / Джеффрі Ріхтер. - 4-те вид. - Київ: Видавничий дім «Вільямс», 2018. - 896 с.
17. Node.js [Електронний ресурс] / Node.js Foundation. - URL: <https://nodejs.org/> (дата звернення: 30.02.2026).
18. Гама Е. Патерни об'єктно-орієнтованого проєктування / Еріх Гама, Річард Хелм, Ральф Джонсон, Джон Вліссідес. - Харків: Фоліо, 2020. - 400 с.
19. Playwright - end-to-end testing [Електронний ресурс] / Microsoft. - URL: <https://playwright.dev/> (дата звернення: 22.04.2026).
20. PostgreSQL 17 Documentation [Електронний ресурс] / PostgreSQL Global Development Group. - URL: <https://www.postgresql.org/docs/17/> (дата звернення: 04.02.2026).
21. Redis [Електронний ресурс] / Redis Ltd. - URL: <https://redis.io/> (дата звернення: 05.02.2026).
22. Docker & Docker Compose [Електронний ресурс] / Docker, Inc. - URL: <https://docs.docker.com/> (дата звернення: 17.03.2026).
23. NGINX Documentation [Електронний ресурс] / NGINX, Inc. - URL: <https://nginx.org/> (дата звернення: 30.02.2026).

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				107
Змн.	Арк.	№ докум.	Підпис	Дата		

24. Playwright Docker images [Електронний ресурс] / Microsoft. - URL: https://hub.docker.com/_/microsoft-playwright (дата звернення: 23.03.2026).
25. Vite [Електронний ресурс]: npm package. - URL: <https://www.npmjs.com/package/vite> (дата звернення: 11.03.2026).
26. Еванс Е. Предметно-орієнтоване проєктування. Структуризація складних програмних систем / Ерік Еванс. Київ: Діалектика, 2020. 592 с.
27. Фрімен А. ASP.NET Core 6 для професіоналів / Адам Фрімен. Київ: Діалектика, 2023. 1008 с.
28. Клеппман М. Розроблення застосунків, що працюють з даними / Мартін Клеппман. - Київ: Діалектика, 2019. - 624 с.
29. Макконнелл С. Досконалий код / Стів Макконнелл; пер. з англ. - Київ: Наш Формат, 2019. - 912 с.
30. Ошероув Р. Мистецтво автономного тестування з прикладами на C# / Рой Ошероув. - Київ: Діалектика, 2021. - 368 с.

ДОДАТКИ

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		109

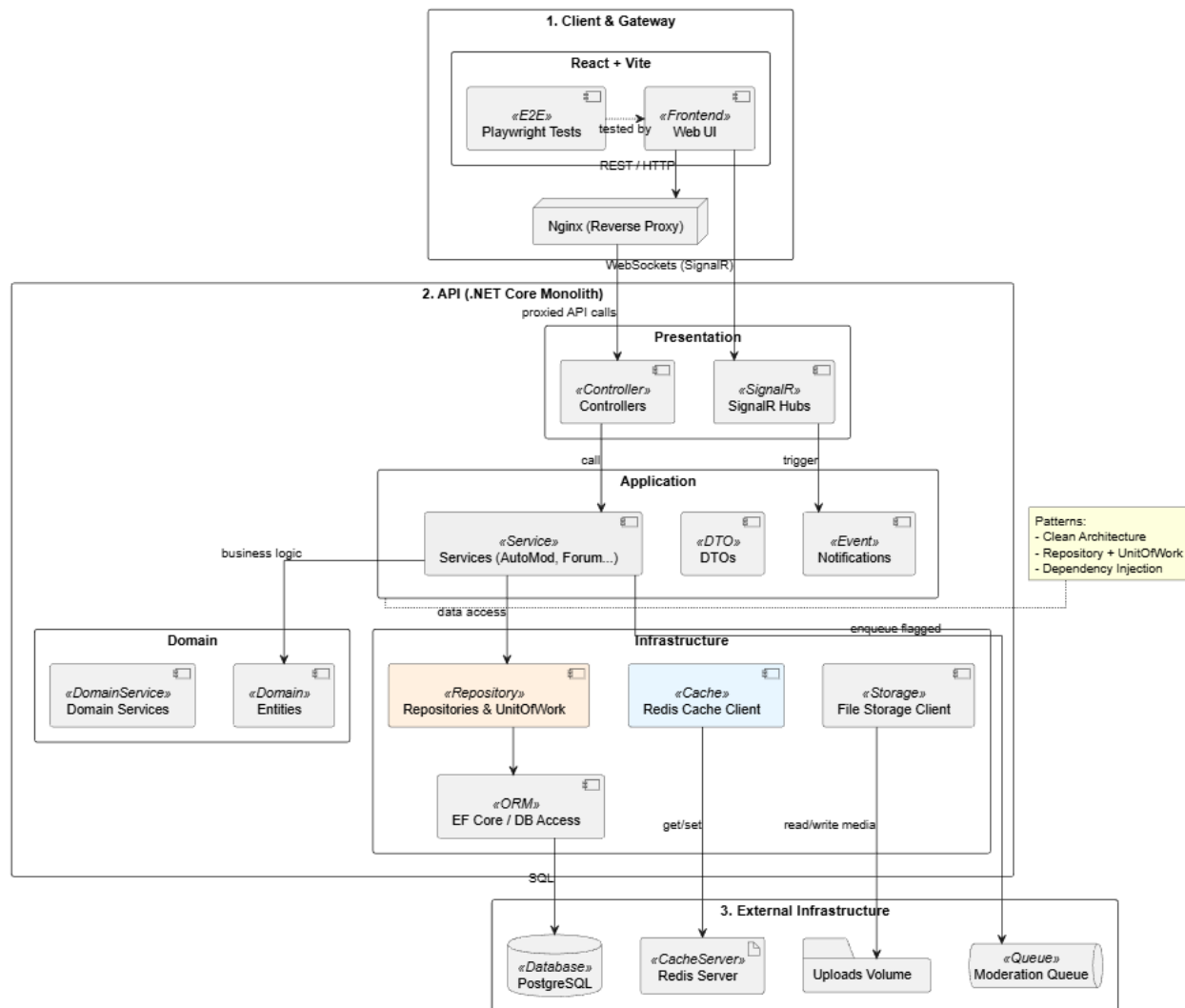


Рис. Діаграма патернів

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		110

Використання патерну Repository + UnitOfWork у SekalLib

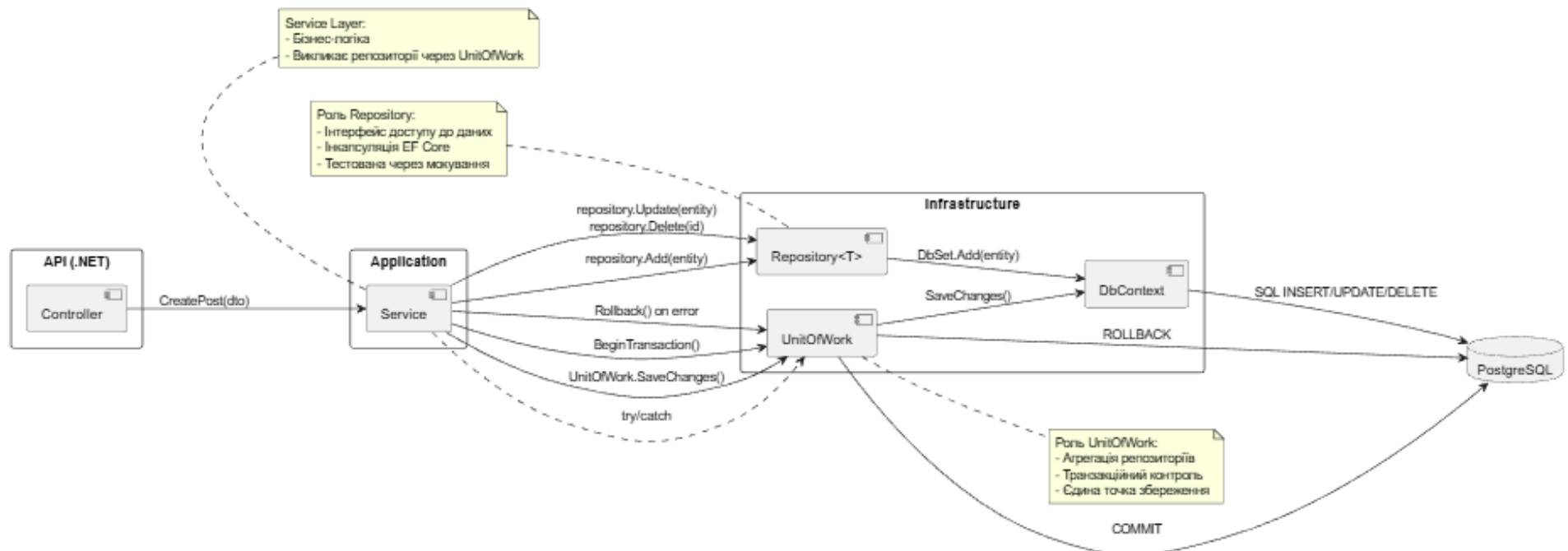


Рис. Діаграма використання Repository та UnitOfWork

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІПЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		111

Лістинг класу ChapterService.cs

```

using System.Security.Cryptography;
using System.Text;
using Microsoft.EntityFrameworkCore;
using SekaiLib.Application.DTOS.Chapters;
using SekaiLib.Application.DTOS.Notifications;
using SekaiLib.Application.DTOS.Titles;
using SekaiLib.Application.Exceptions;
using SekaiLib.Application.Interfaces;
using SekaiLib.Domain.Entities;
using SekaiLib.Domain.Enums;
using SekaiLib.Domain.Interfaces;
using SekaiLib.Domain.Interfaces.Repositories;

namespace SekaiLib.Application.Services;

public class ChapterService : IChapterService
{
    private readonly IUnitOfWork _unitOfWork;
    private readonly INotificationService _notifications;
    private readonly IUserBlockService _userBlockService;
    private readonly IViewTrackingService _viewTracking;
    private readonly IModerationService _moderation;

    public ChapterService(IUnitOfWork unitOfWork, INotificationService
notifications,
        IUserBlockService userBlockService, IViewTrackingService viewTracking,
        IModerationService moderation)
    {
        _unitOfWork = unitOfWork;
        _notifications = notifications;
        _userBlockService = userBlockService;
        _viewTracking = viewTracking;
        _moderation = moderation;
    }

    public async Task<IEnumerable<ChapterDto>> GetChaptersByTitleAsync(Guid titleId,
Guid? teamId = null)
    {
        var titleExists = await _unitOfWork.Titles.ExistsAsync(titleId);
        if (!titleExists)
            throw new NotFoundException("Title", titleId);

        var chapters = await _unitOfWork.Chapters.GetByTitleIdAsync(titleId);

        var query = chapters.AsEnumerable();
        if (teamId.HasValue)
            query = query.Where(c => c.TranslationTeamId == teamId.Value);

        var title = await _unitOfWork.Titles.GetByIdAsync(titleId);
        if (title == null)
            throw new NotFoundException("Title", titleId);

        return query
            .OrderBy(c => c.Number)
            .Select(c => new ChapterDto(c.Id, c.Number, c.Name, c.PublishedAt,
c.IsPremium, c.Price, c.TranslationTeamId, c.TranslationTeam?.Name, c.TitleId,
title.Name, title.CoverImageUrl, c.ViewCount));
    }

    public async Task<ChapterContentDto> GetChapterContentAsync(Guid chapterId)
    {

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Єфремов Ю.М.				112
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        var chapter = await _unitOfWork.Chapters.GetByIdAsync(chapterId);
        if (chapter == null)
            throw new NotFoundException("Chapter", chapterId);

        return await BuildChapterContentDto(chapter);
    }

    public async Task<ChapterContentDto> GetChapterContentByNumberAsync(Guid
titleId, int chapterNumber, Guid? userId = null)
    {
        var chapter = await _unitOfWork.Chapters.GetByTitleAndNumberAsync(titleId,
chapterNumber);
        if (chapter == null)
            throw new NotFoundException($"Chapter number {chapterNumber} for Title
{titleId} was not found");

        return await BuildChapterContentDto(chapter, userId);
    }

    private async Task<ChapterContentDto>
BuildChapterContentDto(Domain.Entities.Chapter chapter, Guid? userId = null)
    {
        var title = await _unitOfWork.Titles.GetByIdAsync(chapter.TitleId);
        if (title == null)
            throw new NotFoundException("Title", chapter.TitleId);

        var allChapters = (await
_unitOfWork.Chapters.GetByTitleIdAsync(chapter.TitleId))
            .OrderBy(c => c.Number)
            .ToList();

        var currentIndex = allChapters.FindIndex(c => c.Number == chapter.Number);
        int? previousChapterNumber = currentIndex > 0 ? allChapters[currentIndex -
1].Number : null;
        int? nextChapterNumber = currentIndex < allChapters.Count - 1 ?
allChapters[currentIndex + 1].Number : null;

        var earlyAccessExpired = chapter.EarlyAccessUntil.HasValue
            && DateTime.UtcNow > chapter.EarlyAccessUntil.Value;
        var isPremium = chapter.IsPremium && chapter.Price > 0 &&
!earlyAccessExpired;
        var isLocked = false;

        if (isPremium)
        {
            if (userId.HasValue)
            {
                var hasPurchase = await _unitOfWork.UserPurchases.Query()
                    .AnyAsync(p => p.UserId == userId.Value && p.ChapterId ==
chapter.Id);
                isLocked = !hasPurchase;
            }
            else
            {
                isLocked = true;
            }
        }

        return new ChapterContentDto(
            chapter.Id,
            chapter.Number,
            chapter.Name,
            isLocked ? string.Empty : chapter.Content,
            chapter.PublishedAt,
            title.Id,
            title.Name,

```

```

        previousChapterNumber,
        nextChapterNumber,
        chapter.ViewCount,
        isPremium,
        isLocked,
        chapter.Price,
        chapter.EarlyAccessUntil
    );
}

public async Task<ChapterContentDto> CreateAsync(Guid userId, Guid titleId,
CreateChapterRequest request)
{
    var title = await _unitOfWork.Titles.GetByIdAsync(titleId);
    if (title == null)
        throw new NotFoundException("Title", titleId);

    var user = await _unitOfWork.Users.GetByIdAsync(userId);
    if (user == null)
        throw new UnauthorizedException();

    var canCreate = title.PublisherId == userId || user.Role ==
UserRole.Administrator;

    if (!canCreate && request.TranslationTeamId.HasValue)
    {
        var teamMember = await _unitOfWork.TranslationTeamMembers.Query()
            .FirstOrDefaultAsync(m => m.TeamId ==
request.TranslationTeamId.Value && m.UserId == userId);

        if (teamMember != null)
            canCreate = true;
    }

    if (!canCreate)
    {
        var isAnyTeamMember = await _unitOfWork.TranslationTeamMembers.Query()
            .AnyAsync(m => m.UserId == userId);

        if (isAnyTeamMember)
            canCreate = true;
    }

    if (!canCreate)
        throw new ForbiddenException();

    var existingChapter = await
_unitOfWork.Chapters.GetByTitleAndNumberAsync(titleId, request.ChapterNumber);
    if (existingChapter != null)
        throw new ValidationException(new Dictionary<string, string[]>
{
    { nameof(request.ChapterNumber), new[] { $"Глава з номером
{request.ChapterNumber} вже існує" } }
});

    var isTeamUpload = request.TranslationTeamId.HasValue
        && title.PublisherId != userId
        && user.Role < UserRole.Administrator;

    var chapter = new Chapter
    {
        Id = Guid.NewGuid(),
        TitleId = titleId,
        Number = request.ChapterNumber,
        Name = request.Name,
        Content = PrepareTextForDb(request.Content),
    }
}

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		114

```

        IsPremium = request.IsPremium,
        Price = request.Price,
        TranslationTeamId = request.TranslationTeamId,
        IsApproved = !isTeamUpload,
        PublishedAt = DateTime.UtcNow,
        CreatedAt = DateTime.UtcNow,
        EarlyAccessUntil = request.EarlyAccessUntil.HasValue
            ? DateTime.SpecifyKind(request.EarlyAccessUntil.Value,
DateTimeKind.Utc)
            : null,
    };

    await _unitOfWork.Chapters.AddAsync(chapter);

    if (isTeamUpload)
        await _moderation.EnqueueAsync("Chapter", chapter.Id, null, userId,
"chapter_upload");

    await _unitOfWork.SaveChangesAsync();

    var readingEntries = await _unitOfWork.ReadingLists.Query()
        .Where(rl => rl.TitleId == titleId)
        .Select(rl => new { rl.UserId, rl.Status, rl.UserListId })
        .ToListAsync();

    var followerGroups = readingEntries.GroupBy(e => e.UserId);

    var finalFollowers = new List<Guid>();
    foreach (var group in followerGroups)
    {
        var followerId = group.Key;
        if (followerId == userId) continue;

        var follower = await _unitOfWork.Users.GetByIdAsync(followerId);
        if (follower == null) continue;

        int[] notifyStatuses = Array.Empty<int>();
        string[] notifyUserListIds = Array.Empty<string>();
        try
        {
            if (!string.IsNullOrEmpty(follower.NotifyListStatuses))
                notifyStatuses =
System.Text.Json.JsonSerializer.Deserialize<int[]>(follower.NotifyListStatuses) ??
Array.Empty<int>();
        }
        catch { }
        try
        {
            if (!string.IsNullOrEmpty(follower.NotifyUserListIds))
                notifyUserListIds =
System.Text.Json.JsonSerializer.Deserialize<string[]>(follower.NotifyUserListIds) ??
Array.Empty<string>();
        }
        catch { }

        var shouldNotify = false;

        foreach (var entry in group)
        {
            if (entry.UserListId.HasValue)
            {
                var listIdStr = entry.UserListId.Value.ToString();
                if (notifyUserListIds.Contains(listIdStr))
                {
                    shouldNotify = true;
                    break;
                }
            }
        }
    }

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Єфремов Ю.М.				115
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        }
    }
    else
    {
        if (notifyStatuses.Contains((int)entry.Status))
        {
            shouldNotify = true;
            break;
        }
    }
}

if (shouldNotify)
    finalFollowers.Add(followerId);
}

if (request.TranslationTeamId.HasValue)
{
    var teamsOnThisTitle = await _unitOfWork.Chapters.Query()
        .Where(c => c.TitleId == titleId && c.TranslationTeamId != null)
        .Select(c => c.TranslationTeamId.Value)
        .Distinct()
        .ToListAsync();

    var followerSubscriptions = await
        _unitOfWork.TranslationTeamSubscriptions.Query()
        .Where(s => finalFollowers.Contains(s.UserId) &&
            teamsOnThisTitle.Contains(s.TeamId))
        .ToListAsync();

    var filtered = new List<Guid>();
    foreach (var followerId in finalFollowers)
    {
        var userSubscribedTeams = followerSubscriptions
            .Where(s => s.UserId == followerId)
            .Select(s => s.TeamId)
            .ToList();

        if (userSubscribedTeams.Any())
        {
            if (userSubscribedTeams.Contains(request.TranslationTeamId.Value))
                filtered.Add(followerId);
        }
        else
        {
            filtered.Add(followerId);
        }
    }

    finalFollowers = filtered;
}

foreach (var followerId in finalFollowers)
{
    await _notifications.CreateAsync(new CreateNotificationRequest(
        followerId,
        NotificationType.NewChapter,
        "Нова глава",
        $"{title.Name} - Глава {chapter.Number}",
        $"/titles/{titleId}/chapters/{chapter.Number}",
        userId,
        titleId,
        chapter.Id
    ));
}

```

```

        if (request.TranslationTeamId.HasValue)
        {
            var teamChaptersCount = await _unitOfWork.Chapters.Query()
                .CountAsync(c => c.TitleId == titleId && c.TranslationTeamId ==
request.TranslationTeamId.Value);

            if (teamChaptersCount == 1)
            {
                var teamSubscriberIds = await
_unitOfWork.TranslationTeamSubscriptions.Query()
                    .Where(s => s.TeamId == request.TranslationTeamId.Value)
                    .Select(s => s.UserId)
                    .Distinct()
                    .ToListAsync();

                var team = await
_unitOfWork.TranslationTeams.GetByIdAsync(request.TranslationTeamId.Value);

                foreach (var subscriberId in teamSubscriberIds.Where(id => id !=
userId))
                {
                    await _notifications.CreateAsync(new CreateNotificationRequest(
subscriberId,
NotificationType.NewTeamChapter,
"КОМАНДА РОЗПОЧАЛА ПЕРЕВЕДЕННЯ",
${team?.Name ?? "Команда"} розпочала перекладання:
{title.Name}",
"/titles/{titleId}",
userId,
titleId,
chapter.Id
));
                }
            }

            return await BuildChapterContentDto(chapter);
        }

public async Task<ChapterContentDto> UpdateAsync(Guid userId, Guid chapterId,
UpdateChapterRequest request)
{
    var chapter = await _unitOfWork.Chapters.GetByIdAsync(chapterId);
    if (chapter == null)
        throw new NotFoundException("Chapter", chapterId);

    var title = await _unitOfWork.Titles.GetByIdAsync(chapter.TitleId);
    if (title == null)
        throw new NotFoundException("Title", chapter.TitleId);

    var user = await _unitOfWork.Users.GetByIdAsync(userId);
    if (user == null)
        throw new UnauthorizedException();

    var canManage = title.PublisherId == userId || user.Role ==
UserRole.Administrator;

    if (!canManage && chapter.TranslationTeamId.HasValue)
    {
        canManage = await _unitOfWork.TranslationTeamMembers.Query()
            .AnyAsync(m => m.TeamId == chapter.TranslationTeamId.Value && m.UserId
== userId);
    }

    if (!canManage)
        throw new ForbiddenException();
}

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Єфремов Ю.М.				117
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        if (chapter.Number != request.ChapterNumber)
        {
            var existingChapter = await
                _unitOfWork.Chapters.GetByTitleAndNumberAsync(chapter.TitleId,
                    request.ChapterNumber);
            if (existingChapter != null && existingChapter.Id != chapterId)
                throw new ValidationException(new Dictionary<string, string[]> { {
                    nameof(request.ChapterNumber), new[] { $"Глава з номером {request.ChapterNumber} вже
                    існує" } } });
        }

        chapter.Number = request.ChapterNumber;
        chapter.Name = request.Name;
        chapter.Content = PrepareTextForDb(request.Content);
        chapter.IsPremium = request.IsPremium;
        chapter.Price = request.Price;

        await _unitOfWork.Chapters.UpdateAsync(chapter);
        await _unitOfWork.SaveChangesAsync();

        return await BuildChapterContentDto(chapter);
    }

    public async Task DeleteAsync(Guid userId, Guid chapterId)
    {
        var chapter = await _unitOfWork.Chapters.GetByIdAsync(chapterId);
        if (chapter == null)
            throw new NotFoundException("Chapter", chapterId);

        var title = await _unitOfWork.Titles.GetByIdAsync(chapter.TitleId);
        if (title == null)
            throw new NotFoundException("Title", chapter.TitleId);

        var user = await _unitOfWork.Users.GetByIdAsync(userId);
        if (user == null)
            throw new UnauthorizedException();

        var canManage = title.PublisherId == userId || user.Role ==
            UserRole.Administrator;

        if (!canManage && chapter.TranslationTeamId.HasValue)
        {
            canManage = await _unitOfWork.TranslationTeamMembers.Query()
                .AnyAsync(m => m.TeamId == chapter.TranslationTeamId.Value && m.UserId
                    == userId);
        }
        if (!canManage)
        {
            canManage = await _unitOfWork.TranslationTeamMembers.Query()
                .AnyAsync(m => m.UserId == userId);
        }

        if (!canManage)
            throw new ForbiddenException();

        await _unitOfWork.Chapters.DeleteAsync(chapter);
        await _unitOfWork.SaveChangesAsync();
    }

    public async Task<IEnumerable<LatestChapterDto>> GetLatestChaptersAsync(int count)
    {
        var chapters = await _unitOfWork.Chapters.GetAllAsync();

        return chapters
            .Where(c => c.Title != null)
    }

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Сфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		118

```

        .OrderByDescending(c => c.PublishedAt)
        .Take(count)
        .Select(c => new LatestChapterDto
        {
            Id = c.Id,
            Number = c.Number,
            CreatedAt = c.CreatedAt,
            Title = new TitleDto(
                c.Title.Id,
                c.Title.Name,
                c.Title.Author ?? "",
                c.Title.CoverImageUrl ?? "",
                c.Title.Description ?? "",
                c.Title.CountryOfOrigin ?? string.Empty,
                c.Title.Status,
                0
            )
        })
    };
}

public async Task UpdateReadingProgressAsync(Guid userId, Guid titleId, int
chapterNumber, int page)
{
    var progress = await _unitOfWork.UserReadingProgresses
        .Query()
        .FirstOrDefaultAsync(p => p.UserId == userId && p.TitleId == titleId);

    if (progress == null)
    {
        progress = new UserReadingProgress
        {
            Id = Guid.NewGuid(),
            UserId = userId,
            TitleId = titleId
        };
        await _unitOfWork.UserReadingProgresses.AddAsync(progress);
    }

    progress.ChapterNumber = chapterNumber;
    progress.CurrentPage = page;
    progress.LastReadAt = DateTime.UtcNow;

    await _unitOfWork.SaveChangesAsync();
}

private string PrepareTextForDb(string input)
{
    if (string.IsNullOrEmpty(input)) return input;

    return input
        .Replace("\r\n", "\n")
        .Replace("\r", "\n")
        .Replace(" - ", " - ")
        .Replace("...", "...")
        .Trim();
}

public async Task<IEnumerable<ChapterCommentResponse>> GetCommentsAsync(Guid
chapterId, Guid? currentUserId)
{
    var chapter = await _unitOfWork.Chapters.GetByIdAsync(chapterId);
    if (chapter == null)
        throw new NotFoundException("Chapter", chapterId);

    var comments = await
        _unitOfWork.Chapters.GetCommentsByChapterIdAsync(chapterId);
}

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		119

```

        if (currentUserId.HasValue)
        {
            var blockedIds = (await
_userBlockService.GetBlockedUserIdsAsync(currentUserId.Value)).ToHashSet();
            comments = comments.Where(c => !blockedIds.Contains(c.UserId)).ToList();
        }

        var byId = comments.ToDictionary(c => c.Id);
        var roots = new List<ChapterComment>();
        foreach (var c in comments)
        {
            if (c.ParentCommentId.HasValue && byId.TryGetValue(c.ParentCommentId.Value,
out var parent))
            {
                parent.Replies.Add(c);
            }
            else
            {
                roots.Add(c);
            }
        }

        ChapterCommentResponse Map(ChapterComment c)
        {
            var likes = c.Reactions.Count(r => r.Type == ReactionType.Like);
            var dislikes = c.Reactions.Count(r => r.Type == ReactionType.Dislike);
            var userReaction = currentUserId.HasValue
                ? c.Reactions.FirstOrDefault(r => r.UserId == currentUserId.Value)?.Type
                : null;

            return new ChapterCommentResponse(
                c.Id,
                c.UserId,
                c.User.Username,
                c.User.AvatarUrl,
                c.Content,
                c.CreatedAt,
                likes,
                dislikes,
                userReaction,
                c.ParentCommentId,
                c.Replies.OrderBy(r => r.CreatedAt).Select(Map)
            );
        }

        return roots.OrderBy(r => r.CreatedAt).Select(Map);
    }

    public async Task<ChapterCommentResponse> AddCommentAsync(Guid userId, Guid
chapterId, CreateChapterCommentRequest request)
    {
        var chapter = await _unitOfWork.Chapters.GetByIdAsync(chapterId);
        if (chapter == null)
            throw new NotFoundException("Chapter", chapterId);

        var user = await _unitOfWork.Users.GetByIdAsync(userId);
        if (user == null)
            throw new UnauthorizedException();

        if (request.ParentCommentId.HasValue)
        {
            var parent = await
_unitOfWork.Chapters.GetCommentByIdAsync(request.ParentCommentId.Value);
            if (parent == null)

```

```

        throw new NotFoundException("ChapterComment",
request.ParentCommentId.Value);

        if (await _userService.IsBlockedAsync(parent.UserId, userId))
            throw new ForbiddenException("Ви не можете відповідати на коментарі
цього користувача.");
    }

    var comment = new ChapterComment
    {
        Id = Guid.NewGuid(),
        ChapterId = chapterId,
        UserId = userId,
        Content = request.Content,
        CreatedAt = DateTime.UtcNow,
        ParentCommentId = request.ParentCommentId
    };

    await _unitOfWork.Chapters.AddCommentAsync(comment);
    await _unitOfWork.SaveChangesAsync();

    if (comment.ParentCommentId.HasValue)
    {
        var parent = await
        _unitOfWork.Chapters.GetCommentByIdAsync(comment.ParentCommentId.Value);
        if (parent != null && parent.UserId != userId)
        {
            await _notifications.CreateAsync(new CreateNotificationRequest(
                parent.UserId,
                NotificationType.CommentReply,
                "Відповідь на коментар",
                $"{user.Username} відповів(ла) на ваш коментар до глави
{chapter.Number}",
                $"/titles/{chapter.TitleId}/chapters/{chapter.Number}#comment-
{comment.Id}",
                userId,
                chapter.TitleId,
                chapter.Id
            ));
        }
    }

    return new ChapterCommentResponse(
        comment.Id,
        userId,
        user.Username,
        user.AvatarUrl,
        comment.Content,
        comment.CreatedAt,
        0,
        0,
        null,
        comment.ParentCommentId,
        Enumerable.Empty<ChapterCommentResponse>())
    };

    public async Task<ChapterCommentResponse> UpdateCommentAsync(Guid userId, Guid
commentId, UpdateChapterCommentRequest request)
    {
        var comment = await _unitOfWork.Chapters.GetCommentByIdAsync(commentId);
        if (comment == null)
            throw new NotFoundException("ChapterComment", commentId);
        if (comment.UserId != userId)
            throw new ForbiddenException();
    }

```

```

        comment.Content = request.Content;
        await _unitOfWork.SaveChangesAsync();

        return await BuildCommentResponse(commentId, userId);
    }

    public async Task<ChapterCommentResponse> SetCommentReactionAsync(Guid userId,
        Guid commentId, ReactionType type)
    {
        var existing = await _unitOfWork.Chapters.GetCommentReactionAsync(userId,
            commentId);
        if (existing != null)
        {
            if (existing.Type == type)
            {
                return await BuildCommentResponse(commentId, userId);
            }

            existing.Type = type;
            await _unitOfWork.SaveChangesAsync();
            return await BuildCommentResponse(commentId, userId);
        }

        var reaction = new ChapterCommentReaction
        {
            Id = Guid.NewGuid(),
            CommentId = commentId,
            UserId = userId,
            Type = type,
            CreatedAt = DateTime.UtcNow
        };

        await _unitOfWork.Chapters.AddCommentReactionAsync(reaction);
        await _unitOfWork.SaveChangesAsync();

        return await BuildCommentResponse(commentId, userId);
    }

    public async Task RemoveCommentReactionAsync(Guid userId, Guid commentId)
    {
        var reaction = await _unitOfWork.Chapters.GetCommentReactionAsync(userId,
            commentId);
        if (reaction == null)
            throw new NotFoundException("ChapterCommentReaction", commentId);

        await _unitOfWork.Chapters.RemoveCommentReactionAsync(reaction);
        await _unitOfWork.SaveChangesAsync();
    }

    public async Task DeleteCommentAsync(Guid userId, Guid commentId)
    {
        var comment = await _unitOfWork.Chapters.GetCommentByIdAsync(commentId);
        if (comment == null)
            throw new NotFoundException("ChapterComment", commentId);

        if (comment.UserId != userId)
            throw new ForbiddenException();

        await _unitOfWork.Chapters.RemoveCommentAsync(comment);
        await _unitOfWork.SaveChangesAsync();
    }

    private async Task<ChapterCommentResponse> BuildCommentResponse(Guid commentId,
        Guid currentUserId)
    {
        var all = await _unitOfWork.Chapters.GetCommentsByChapterIdAsync(

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		122

```

        await GetChapterIdByCommentId(commentId)
    };

    var blockedIds = (await
_userBlockService.GetBlockedUserIdsAsync(currentUserId)).ToHashSet();
    all = all.Where(c => !blockedIds.Contains(c.UserId)).ToList();

    var dict = all.ToDictionary(c => c.Id);
    foreach (var c in all)
    {
        if (c.ParentCommentId.HasValue &&
dict.TryGetValue(c.ParentCommentId.Value, out var parent))
            parent.Replies.Add(c);
    }

    ChapterCommentResponse Map(ChapterComment c)
    {
        var likes = c.Reactions.Count(r => r.Type == ReactionType.Like);
        var dislikes = c.Reactions.Count(r => r.Type == ReactionType.Dislike);
        var userReaction = c.Reactions.FirstOrDefault(r => r.UserId ==
currentUserId)?.Type;
        return new ChapterCommentResponse(
            c.Id,
            c.UserId,
            c.User.Username,
            c.User.AvatarUrl,
            c.Content,
            c.CreatedAt,
            likes,
            dislikes,
            userReaction,
            c.ParentCommentId,
            c.Replies.OrderBy(r => r.CreatedAt).Select(Map)
        );
    }

    if (!dict.TryGetValue(commentId, out var target))
        throw new NotFoundException("ChapterComment", commentId);

    return Map(target);
}

public async Task<int> RecordViewAsync(Guid chapterId, Guid? userId, string
ipAddress)
{
    var chapter = await _unitOfWork.Chapters.GetByIdAsync(chapterId);
    if (chapter == null)
        throw new NotFoundException("Chapter", chapterId);

    var isNew = await _viewTracking.TryRecordViewAsync("chapter", chapterId,
userId, ipAddress);
    if (!isNew)
        return chapter.ViewCount;

    var ipHash = HashIp(ipAddress);
    await _unitOfWork.ChapterViews.AddAsync(new ChapterView
    {
        Id = Guid.NewGuid(),
        ChapterId = chapterId,
        UserId = userId,
        IpHash = ipHash,
        ViewedAt = DateTime.UtcNow
    });

    chapter.ViewCount++;
    await _unitOfWork.Chapters.UpdateAsync(chapter);
}

```

		Петров Д.В.			ІІЗ.КР.Б – 121 – 26 – ІІЗ	Арк.
		Єфремов Ю.М.				
Змн.	Арк.	№ докум.	Підпис	Дата		123

```

        await _unitOfWork.SaveChangesAsync();

        return chapter.ViewCount;
    }

    private static string HashIp(string ip)
    {
        var bytes = SHA256.HashData(Encoding.UTF8.GetBytes(ip));
        return Convert.ToString(bytes, 16);
    }

    private async Task<Guid> GetChapterIdByCommentId(Guid commentId)
    {
        var comment = await _unitOfWork.Chapters.GetCommentByIdAsync(commentId);
        if (comment == null)
            throw new NotFoundException("ChapterComment", commentId);
        return comment.ChapterId;
    }
}

```

		Петров Д.В.			ІПЗ.КР.Б – 121 – 26 – ІЗ	Арк.
		Єфремов Ю.М.				124
Змн.	Арк.	№ докум.	Підпис	Дата		