

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:

Соціальна мережа "FanPulse" для геймерських спільнот

Виконав студент групи ПЗ-22-1

ЦЕЛУЄВ Віталій Олегович

Керівник роботи:

САВІЦЬКИЙ Роман Святославович

Рецензент:

КУШНІР Надія Олександрівна

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 2026р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ЦЕЛУЄВ Віталій Олегович**

Керівник роботи: **САВІЦЬКИЙ Роман Святославович**

Тема роботи: **«Соціальна мережа "FanPulse" для геймерських спільнот»**,
затверджена Наказом закладу вищої освіти від **«27» лютого 2026р., №92.с**

Вихідні дані для роботи: **взаємодія користувачів у межах геймерських спільнот, що включає в себе публікацію новин, обговорення, створення тематичних спільнот, систему ролей користувачів, адміністрування контенту з використанням сучасних технологій.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Савіцький Р.С.	05.03.2026	05.03.2026
2	Савіцький Р.С.	18.03.2026	18.03.2026
3	Савіцький Р.С.	01.04.2026	01.04.2026

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра присвячена розробці соціальної мережі для фанатів комп'ютерних ігор. Пояснювальна записка до випускної роботи містить 104 сторінки, 28 ілюстрацій, 48 таблиць та 31 джерел.

Метою роботи є проєктування та розробка веборієнтованої соціальної мережі, призначеної для об'єднання користувачів за інтересами у сфері комп'ютерних ігор, забезпечення їхньої взаємодії, обміну інформацією та створення тематичних спільнот.

У першому розділі виконано аналіз предметної області та досліджено сучасні рішення у сфері ігрових соціальних платформ і спільнот. Розглянуто функціональні можливості популярних сервісів, їхні переваги та недоліки. На основі проведеного аналізу визначено основні вимоги до майбутньої системи та сформовано перелік її ключових функцій.

У другому розділі здійснено проєктування програмного продукту. Визначено архітектуру системи, побудовано сценарії використання та описано взаємодію основних компонентів. Обґрунтовано вибір технологічного стеку, який забезпечує надійність, продуктивність і можливість подальшого масштабування системи відповідно до зростання кількості користувачів.

У третьому розділі наведено процес реалізації соціальної мережі та описано розроблені програмні модулі. Реалізовано систему реєстрації та авторизації користувачів, керування ролями та правами доступу, створення й редагування публікацій, тематичні форуми та спільноти, механізми пошуку контенту, інформаційні розсилки та засоби адміністрування.

Особливу увагу приділено створенню зручного користувацького інтерфейсу, забезпеченню безпеки даних і підтримці ефективної взаємодії між користувачами. Розроблена система дозволяє публікувати новини, обговорювати ігрові події, об'єднуватися в тематичні групи та отримувати актуальну інформацію

					ІПЗ.КР.Б-121-26-ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата					
Розроб.		Целусєв В.О.			Соціальна мережа "FanPulse" для геймерських спільнот		Літ.	Арк.	Аркуші
Керівник.		Савицький Р.С.						3	104
							Житомирська політехніка, група ІПЗ-22-1		
Рецензент		Кушнір Н.О.							
Зав. каф.		Вакалюк, Т.А.							

про улюблені комп'ютерні ігри.

Результатом роботи є функціональна соціальна мережа для фанатів комп'ютерних ігор, яка забезпечує комфортне спілкування користувачів, підтримує створення та розвиток спільнот, а також може бути розширена додатковими сервісами в майбутньому.

КЛЮЧОВІ СЛОВА: СОЦІАЛЬНА МЕРЕЖА, ВЕБЗАСТОСУНОК, ІГРОВА СПІЛЬНОТА, ФОРУМ, КОРИСТУВАЧ, АВТОРИЗАЦІЯ, АУТЕНТИФІКАЦІЯ, ВЕБСЕРВІС, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, БАЗА ДАНИХ, REST API.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ABSTRACT

This bachelor's thesis is dedicated to the development of a social network for computer game enthusiasts. The thesis includes 104 pages, 28 illustrations, 48 tables, and 31 references.

The aim of the project is to design and develop a web-based social network intended to bring together users with shared interests in the field of computer games, facilitate their interaction and information exchange, and create thematic communities.

The first chapter analyzes the subject area and examines current solutions in the field of gaming social platforms and communities. It reviews the functional capabilities of popular services, along with their advantages and disadvantages. Based on this analysis, the main requirements for the future system were identified, and a list of its key functions was compiled.

The second chapter presents the design of the software product. The system architecture is defined, use cases are developed, and the interaction of the main components is described. The choice of the technology stack is justified, ensuring reliability, performance, and the ability to scale the system as the number of users grows.

Chapter 3 outlines the implementation process of the social network and describes the developed software modules. The system includes user registration and authorization, role and access rights management, the creation and editing of posts, thematic forums and communities, content search mechanisms, newsletters, and administrative tools.

Particular attention was paid to creating a user-friendly interface, ensuring data security, and supporting effective interaction among users. The developed system allows users to post news, discuss gaming events, form thematic groups, and receive up-to-date information about their favorite computer games.

The result of this work is a functional social network for computer game fans that facilitates comfortable user communication, supports the creation and development of communities, and can be expanded with additional services in the future.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савицький Р.С.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

KEYWORDS: SOCIAL NETWORK, WEB APPLICATION, GAMING COMMUNITY, FORUM, USER, AUTHORIZATION, AUTHENTICATION, WEB SERVICE, CLIENT-SERVER ARCHITECTURE, DATABASE, REST API.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

Зміст

РЕФЕРАТ.....	3
ABSTRACT	5
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	8
ВСТУП.....	9
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ.....	11
1.1 Аналіз аналогів програмного забезпечення продукту	11
1.2 Постановка задачі	15
1.3 Вибір архітектури та технологій розробки	19
1.4 Обґрунтування вибору програмних засобів розробки.....	25
Висновки до першого розділу	30
РОЗДІЛ 2. ПРОЄКТУВАННЯ ЗАСТОСУНКУ ТА ВИМОГИ ДО ФУНКЦІОНАЛУ	32
2.1 Вимоги до системи	32
2.2 Моделювання архітектури та функціональних можливостей системи ..	37
2.3 Розроблення структури бази даних	41
2.4 Організація взаємодії користувача із системою та безпекою	59
Висновки до другого розділу.....	63
РОЗДІЛ 3. РЕАЛІЗАЦІЯ СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ ФАНАТІВ КОМП'ЮТЕРНИХ ІГОР.....	64
3.1 Реалізація функціоналу для неавторизованих користувачів.....	64
3.2 Реалізація адміністративної частини системи	76
Висновки до третього розділу	82
ВИСНОВКИ	84
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	86
ДОДАТКИ.....	89

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

СУБД – Система управління базами даних

API (Application Programming Interface) – Інтерфейс прикладного програмування

ASGI (Asynchronous Server Gateway Interface) – Асинхронний інтерфейс шлюзу сервера

CDN (Content Delivery Network) – Мережа доставки контенту

CI/CD (Continuous Integration / Continuous Deployment) – Безперервна інтеграція та доставка

DOM (Document Object Model) – Об'єктна модель документа

HTTP (HyperText Transfer Protocol) – Протокол передачі гіпертексту

JSON (JavaScript Object Notation) – Текстовий формат обміну даними

JWT (JSON Web Token) – Відкритий стандарт для створення токенів доступу

ORM (Object-Relational Mapping) – Об'єктно-реляційне відображення

REST (Representational State Transfer) – Архітектурний стиль для розробки розподілених систем

SPA (Single Page Application) – Односторінковий вебзастосунок

SQL (Structured Query Language) – Мова структурованих запитів

UGC (User-Generated Content) – Контент, створений користувачами

URL (Uniform Resource Locator) – Уніфікований локатор ресурсів

UX (User Experience) – Користувацький досвід

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Сучасна ігрова індустрія стрімко розвивається та об'єднує мільйони користувачів по всьому світу. Разом із зростанням популярності комп'ютерних ігор зростає потреба у спеціалізованих платформах, які забезпечують не лише доступ до актуальних ігрових новин, а й можливість спілкування, обміну досвідом, створення тематичних спільнот та взаємодії між користувачами. Існуючі соціальні мережі та інформаційні ресурси здебільшого орієнтовані на широке коло інтересів або реалізують лише окремі функції, що змушує користувачів використовувати декілька різних сервісів для отримання необхідної інформації та комунікації. Використання сучасних вебтехнологій дає змогу створити єдину інтегровану платформу, яка об'єднає основні потреби ігрової спільноти та забезпечить зручний обмін контентом і взаємодію між її учасниками. Тому розробка соціальної мережі для фанатів комп'ютерних ігор є актуальним завданням, що відповідає сучасним тенденціям розвитку інформаційних технологій та потребам користувачів.

Метою кваліфікаційної роботи є проектування та розробка застосунку соціальної мережі для фанатів комп'ютерних ігор, що забезпечує взаємодію користувачів, публікацію новин, створення спільнот, участь у форумах та адміністрування контенту.

Для досягнення поставленої мети необхідно виконати такі **завдання**:

- провести аналіз існуючих аналогів та сучасних рішень у сфері ігрових спільнот;
- визначити функціональні та нефункціональні вимоги до системи;
- спроектувати архітектуру програмного забезпечення;
- розробити структуру бази даних;
- реалізувати механізми реєстрації, авторизації та управління обліковими записами користувачів;
- реалізувати систему новин, форумів та спільнот;

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

- розробити адміністративну панель для керування контентом та користувачами;
- провести тестування та оцінити працездатність програмного продукту.

Об’єктом дослідження є процес організації взаємодії користувачів у межах онлайн-спільноти фанатів комп’ютерних ігор.

Предметом дослідження є методи, технології та програмні засоби розробки застосунків соціальних мереж для забезпечення комунікації, обміну інформацією та управління контентом.

Практичне застосування роботи полягає у створенні функціональної платформи для об’єднання користувачів за спільними інтересами у сфері комп’ютерних ігор, публікації новин, організації тематичних обговорень та взаємодії між учасниками спільноти. Система надає можливість створювати профілі, приєднуватися до груп, публікувати дописи, брати участь у форумах і знаходити однодумців для спільної гри. Завдяки модульній архітектурі платформа може бути розширена додатковими функціями, що забезпечує її подальший розвиток і практичне використання для формування та підтримки активних ігрових спільнот.

Апробація результатів роботи. Основні результати випускної кваліфікаційної роботи, пов’язані з дослідженням та розробкою соціальної мережі для фанатів комп’ютерних ігор, були апробовані на XI Міжнародній науково-технічній конференції «Інформаційно-комп’ютерні технології» (м. Житомир, 02–03 квітня 2026 р.) та опубліковані у вигляді двох тез доповідей [1, 2].

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1 Аналіз аналогів програмного забезпечення продукту

Для визначення ролі розробленого програмного забезпечення на ринку було проведено аналіз існуючих програмних рішень у сфері соціальних мереж для фанатів комп'ютерних ігор. Було розглянуто два найбільш популярні сервіси для об'єднання спільнот комп'ютерних ігор.

Для аналізу було обрано платформи Twitch [28] та Discord [6]. Обидва сервіси є популярними серед гравців і надають інструменти для комунікації та взаємодії користувачів. Аналіз їхнього функціоналу дозволив визначити основні можливості, які доцільно реалізувати у розроблюваній системі, а також врахувати переваги існуючих рішень.

Аналоги програмного забезпечення оцінювалися за такими параметрами: призначення та сфера застосування, функціональні можливості, аудиторія та популярність, модель монетизації, зручність використання, інтеграція з іншими сервісами, рівень безпеки та конфіденційності, а також переваги й недоліки кожної платформи. Discord розглядався як сервіс для комунікації та створення спільнот, а Twitch як платформа для проведення прямих трансляцій. Такий підхід дозволяє комплексно порівняти існуючі рішення, визначити їх сильні та слабкі сторони і сформулювати перелік функцій, які доцільно реалізувати у власному програмному продукті.

Discord – це багатофункціональна комунікаційна платформа, яка працює за принципом тематичних спільнот (серверів). Вона була створена у 2015 році як інструмент для геймерів, але з часом перетворилася на універсальний простір для навчання, роботи, творчості та дружнього спілкування. Кожен сервер у Discord має власну структуру каналів – текстових, голосових та відео, що дозволяє організовувати спілкування за темами, інтересами чи проєктами. Користувачі можуть створювати приватні або публічні сервери, налаштовувати ролі та права доступу, що робить платформу гнучкою для управління спільнотами.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

Інтерфейс Discord реалізований як багатопанельна система, яка наведена на рисунку 1.1, що забезпечує логічну навігацію та масштабованість для різних сценаріїв використання. Ліва панель відповідає за швидкий доступ до серверів, кожен із яких має власну ієрархію каналів. Центральна область є основним робочим простором, де відображаються повідомлення, мультимедійний контент та інтеграції, а права панель демонструє учасників із їхніми ролями та статусами. Голосові та відеоканали інтегровані як окремий функціональний блок, що підтримує аудіо- та відеоконференції, а також трансляцію екрану. Нижня панель забезпечує введення повідомлень і доступ до налаштувань користувача. Візуально інтерфейс побудований у мінімалістичному стилі з темним фоном і чіткими акцентами на інтерактивних елементах, що знижує когнітивне навантаження при тривалому використанні. Така архітектура відповідає принципам UX-дизайну: модульність, зрозуміла навігація, персоналізація та можливість масштабування, що робить Discord придатним як для невеликих груп, так і для великих професійних спільнот.

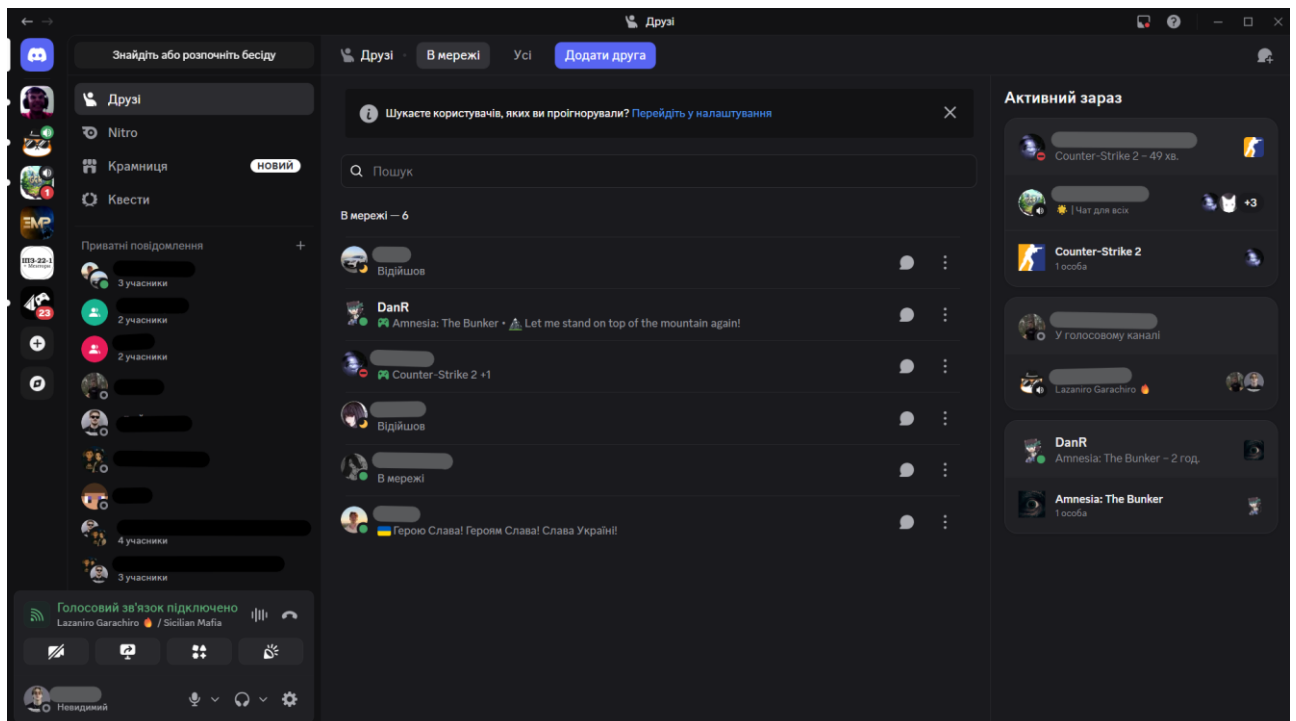


Рис. 1.1. Головний екран застосунку Discord

Twitch – це популярна платформа для відеотрансляцій, яка належить компанії Amazon і функціонує як інтерактивний медіасервіс у режимі реального часу. Її основна особливість полягає у поєднанні потокового відео з живим

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

чатом, що забезпечує двосторонню комунікацію між автором трансляції та аудиторією. Інтерфейс платформи побудований за принципом «контент у центрі»: головна сторінка містить рекомендовані трансляції, категорії за іграми та темами, а також панель пошуку для швидкої навігації. Сторінка трансляції включає відеопотік, інтерактивний чат праворуч, панель підписок і донатів, а також інформацію про стрімера. Для авторів контенту передбачено панель керування стрімом, де можна налаштовувати якість трансляції, інтеграції з іншими сервісами та інструменти монетизації.

Завдяки такій архітектурі Twitch забезпечує масштабованість і високу інтерактивність: користувачі не лише споживають контент, а й активно беруть участь у його створенні через коментарі, реакції та фінансову підтримку. Візуально інтерфейс орієнтований на простоту та швидкий доступ до трансляцій, що робить платформу ефективною як для геймерів, так і для музикантів, творчих авторів чи тих, хто просто прагне спілкування з аудиторією.

Інтерфейс Twitch побудований за принципом максимальної концентрації користувача на контенті та інтерактивності зображеному на рисунку 1.2. Головна сторінка містить блоки рекомендованих трансляцій, категорії за іграми чи темами, а також панель пошуку для швидкої навігації. При переході на сторінку конкретного стріму центральне місце займає відеопотік, праворуч розташований інтерактивний чат, який забезпечує безпосередню комунікацію між стрімером та аудиторією. Під відео знаходяться елементи взаємодії, такі як кнопки підписки, донатів, «бітів», а також інформація про канал і стрімера. Для авторів контенту передбачено окрему панель керування трансляціями, де можна налаштовувати параметри стріму, інтеграції з іншими сервісами та інструменти монетизації.

З інженерної точки зору інтерфейс Twitch реалізує принципи UX-дизайну для потокового відео: контент у центрі, мінімум відволікаючих елементів, швидкий доступ до інтерактивних функцій. Така архітектура забезпечує масштабованість платформи та високу залученість користувачів, адже вони не лише споживають контент, а й активно взаємодіють із ним у реальному часі.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

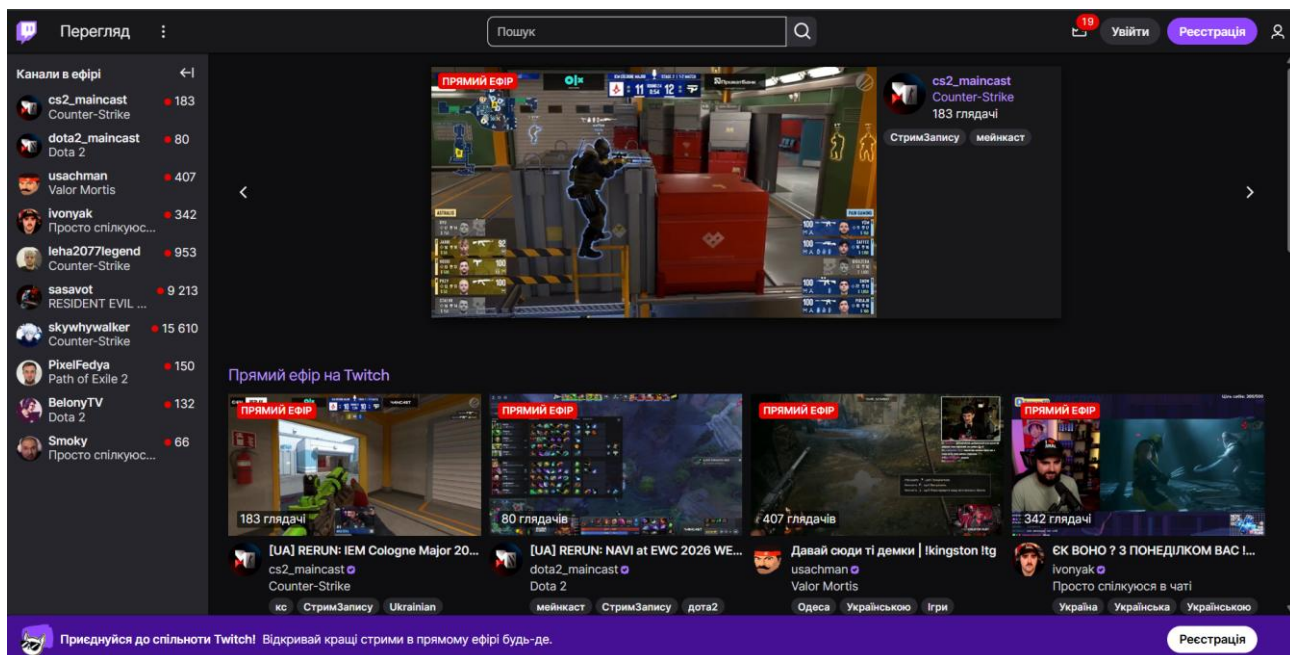


Рис. 1.2. Головний екран Twitch

У таблиці 1.1 наведено порівняння характеристик описаних аналогів із розроблюваним сервісом.

Таблиця 1.1

Порівняння аналогів із розроблюваним сервісом

Функціональна можливість / Характеристика	Discord	Twitch	FanPulse
Орієнтація виключно на ігрову тематику	+/-	+	+
Глобальна стрічка новин та публікацій	—	—	+
Модуль тематичних форумів та обговорень	+/-	—	+
Прямі відеотрансляції (стрімінг)	+	+	—
Система текстових коментарів та лайків до постів	—	—	+
Мінімалістичний інтерфейс із низьким навантаженням	—	+/-	+
Багатоканальні голосові кімнати (голосовий зв'язок)	+	-	-
Профільна взаємодія користувачів та розробників ігор	+/-	-	+

Отже, проведений порівняльний аналіз наочно демонструє, що розроблюваний сервіс FanPulse займає унікальну ринкову позицію, успішно усуваючи функціональні прогалини та архітектурні обмеження існуючих аналогів. У той час як Discord змушує геймерів існувати в ізольованих приватних серверах без можливості структурування публікацій, а Twitch фокусується лише на пасивному

перегляді трансляцій та швидкоплинному чаті, FanPulse пропонує єдину відкриту екосистему з глобальною стрічкою новин, тематичними форумами та класичною системою лайків і коментарів. Відмова від складного та ресурсомісткого функціоналу відеотрансляцій і голосових кімнат дозволила спростити програмну архітектуру та створити мінімалістичний, інтуїтивно зрозумілий інтерфейс, що мінімізує навантаження на користувача й робить розробку даного продукту технічно та практично доцільною.

1.2 Постановка задачі

У межах дипломного проєкту здійснюється розробка платформи FanPulse, призначеної для об'єднання користувачів, які цікавляться комп'ютерними іграми та діяльністю ігрових спільнот. Основним призначенням системи є створення єдиного інформаційного простору для користувачів, де вони можуть отримувати актуальні новини, створювати та приєднуватися до тематичних спільнот, брати участь в обговореннях, спілкуватися в чатах і відстежувати анонси та дати виходу нових ігор. Для підвищення залученості користувачів передбачено персоналізацію контенту відповідно до їхніх інтересів, а також використання елементів гейміфікації, зокрема системи рейтингів, балів і статусів за активність.

Платформа FanPulse розробляється як сучасний вебзастосунок, який забезпечує централізований доступ до інформації та засобів комунікації для представників ігрової спільноти. Основною метою проєкту є створення програмного продукту, який дозволить користувачам не лише отримувати інформацію про світ комп'ютерних ігор, але й активно взаємодіяти між собою, формуючи власні тематичні спільноти та беручи участь у розвитку платформи. При цьому система повинна забезпечувати високу швидкодію, зручний інтерфейс та можливість подальшого розширення функціональних можливостей.

Весь сайт будується за класичним принципом, де є серверна частина, яка відповідає за обробку запитів, реалізацію бізнес-логіки та збереження інформації в базі даних, і клієнтська частина, що формує зовнішній вигляд системи та забезпечує взаємодію користувача з функціоналом платформи через веббраузер. Такий підхід дозволяє розділити функції між окремими компонентами системи,

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

спростити її супровід та модернізацію, а також забезпечити можливість незалежного розвитку серверної та клієнтської частин.

Однією з важливих вимог до системи є реалізація повноцінного гостьового режиму. Будь-який відвідувач без проходження реєстрації повинен мати змогу користуватися основними інформаційними можливостями платформи. До таких можливостей належать перегляд новин, ознайомлення з календарем майбутніх релізів, перегляд інформації про ігри, читання публікацій користувачів, ознайомлення зі спільнотами та перегляд результатів опитувань. При цьому функції, пов'язані з активною взаємодією із системою, такі як написання коментарів, створення власного контенту, участь у голосуваннях та використання чатів, повинні бути доступними лише авторизованим користувачам. Такий підхід дозволяє збільшити аудиторію платформи та одночасно забезпечити контроль за активністю учасників спільноти.

Одним із базових модулів системи є модуль користувачів та авторизації. Його основним призначенням є забезпечення безпечної реєстрації нових користувачів та їхньої автентифікації під час входу в систему. Після успішної реєстрації кожен користувач отримує власний профіль, який містить основну інформацію про нього та може бути персоналізований. Передбачається можливість завантаження та зміни аватара, редагування особистих даних, зміни пароля та перегляду власної активності на платформі. Крім того, система повинна підтримувати механізми відновлення доступу до облікового запису та захисту персональних даних користувачів.

Наступним важливим компонентом є модуль новин. Його призначення полягає в інформуванні користувачів про події ігрової індустрії, анонси нових проєктів, оновлення популярних ігор та інші тематичні матеріали. Новини повинні відображатися у вигляді зручної стрічки з автоматичним підвантаженням нових записів під час прокручування сторінки. Такий механізм дозволяє покращити користувацький досвід та забезпечити швидкий доступ до великої кількості інформації. Користувачі можуть взаємодіяти з новинами за допомогою реакцій та коментарів, що сприяє формуванню активної спільноти. Для підтримання

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

належного рівня культури спілкування всі коментарі повинні перевірятися автоматичною системою фільтрації нецензурної лексики, а також бути доступними для модерації адміністраторами платформи.

Окремий функціональний модуль присвячений календарю релізів комп'ютерних ігор. Його головною задачею є накопичення та відображення інформації про майбутні та вже випущені ігрові проєкти. Для кожної гри формується окрема сторінка, де містяться її основні характеристики: назва, дата виходу, підтримувані платформи, жанр, короткий опис та інші важливі відомості. Користувачі повинні мати можливість переглядати список ігор відповідно до їхнього статусу, наприклад, ті, що вже вийшли, перебувають у розробці або були скасовані. Для полегшення пошуку інформації передбачено систему фільтрації та сортування за жанрами, платформами та іншими параметрами.

Важливою складовою платформи є засоби комунікації між користувачами. Для цього в системі передбачено реалізацію модулів чатів, форумів та спільнот. Модуль чатів забезпечує швидкий обмін повідомленнями між учасниками платформи. Передбачено підтримку приватних діалогів між двома користувачами та групових чатів, присвячених окремим ігровим жанрам або тематикам. Адміністрація системи має можливість створювати та модерувати такі чати, забезпечуючи дотримання встановлених правил поведінки.

Модуль форумів орієнтований на проведення тривалих тематичних обговорень. Адміністратори створюють основні категорії, що відповідають різним напрямкам ігрової тематики, після чого користувачі можуть створювати власні теми, відповідати на повідомлення інших учасників та брати участь у дискусіях. Для підвищення зручності роботи з форумом передбачено систему тегів, можливість пошуку повідомлень, сортування тем за популярністю та підписку на цікаві обговорення.

Модуль спільнот реалізує механізм створення користувацьких об'єднань за інтересами. Користувачі можуть створювати або приєднуватися до кланів, публікувати власний контент, спілкуватися з іншими учасниками та брати участь у внутрішніх активностях. Наявність такого функціоналу сприяє формуванню

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

стійких соціальних зв'язків між користувачами та підвищує рівень їхньої активності на платформі.

Для збільшення зацікавленості користувачів у використанні платформи передбачено впровадження елементів гейміфікації. За виконання різних дій, таких як публікація повідомлень, написання коментарів, участь в обговореннях або створення контенту, користувачі отримують спеціальні бали активності. На основі накопичених балів формується рейтинг користувача та присвоюються певні статуси, які відображають його внесок у розвиток спільноти. Такий механізм мотивує користувачів бути більш активними та регулярно відвідувати платформу.

Особливу увагу під час розробки системи приділено забезпеченню інформаційної безпеки та контролю доступу до функціональних можливостей. Передбачено використання системи ролей, яка визначає права доступу різних категорій користувачів. Звичайні користувачі отримують доступ до основного функціоналу платформи, модератори можуть контролювати окремі розділи та контент, а адміністратори мають повний доступ до всіх компонентів системи та її налаштувань.

Найважливішим та технічно найскладнішим компонентом системи є панель адміністратора. Вона реалізується у вигляді інтерактивного робочого середовища, яке дозволяє контролювати стан усієї платформи. На головній сторінці панелі повинна відображатися актуальна статистична інформація, зокрема кількість зареєстрованих користувачів, нових публікацій, коментарів, створених тем форуму та інших показників активності.

Через адміністративну панель забезпечується повний контроль над інформаційним наповненням платформи. Адміністратор може створювати, редагувати та видаляти новини, керувати інформацією про ігри в календарі релізів, модерувати повідомлення в чатах, форумах і спільнотах, а також здійснювати контроль за коментарями користувачів. Окремий набір функцій призначений для управління обліковими записами користувачів. Передбачено можливість перегляду профілів, блокування порушників, тимчасового обмеження їхньої активності,

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

призначення або відкликання прав модератора та налаштування параметрів роботи системи.

Під час розробки платформи необхідно забезпечити зручність користування інтерфейсом, високу швидкість обробки запитів, цілісність та безпечне зберігання даних, а також можливість масштабування системи в майбутньому. Усі функціональні модулі повинні взаємодіяти між собою як єдина інформаційна система, забезпечуючи користувачам комфортний доступ до необхідних сервісів та стабільну роботу платформи.

Таким чином, у межах дипломного проєкту необхідно розробити багатофункціональну вебплатформу FanPulse, яка поєднуватиме можливості соціальної мережі, інформаційного порталу та засобу комунікації для фанатів комп'ютерних ігор, забезпечуючи зручний доступ до тематичного контенту, сучасні засоби взаємодії між користувачами та ефективні інструменти адміністрування й модерації.

1.3 Вибір архітектури та технологій розробки

Для створення соціальної мережі FanPulse з високим рівнем інтерактивності та швидким обміном даними було обрано класичну трирівневу клієнт-серверну архітектуру, вона розділяється на три незалежні частини: базу даних для надійного зберігання інформації, серверну частину для обробки логіки та запитів, а також клієнтську частину, яка відповідає за відображення інтерфейсу в браузері користувача. Кожен із цих рівнів може розроблятися, тестуватися та масштабуватися окремо, що значно полегшує командну роботу та подальшу підтримку системи. Розділення відповідальностей також дозволяє у майбутньому замінити будь-який із компонентів без необхідності переробляти решту системи, за умови дотримання узгодженого контракту взаємодії між рівнями.

Взаємодія між клієнтом і сервером відбувається за допомогою архітектурного стилю REST API, де фронтенд надсилає стандартизовані HTTP-запити, а бекенд повертає структуровані дані у форматі JSON [10, 13]. Кожен ресурс системи, такий як користувач, повідомлення форуму, спільнота, новина чи реліз гри, має власну унікальну адресу, а дії над ним виконуються за допомогою

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

стандартних HTTP-методів: GET для отримання даних, POST для створення нових записів, PATCH або PUT для оновлення існуючих, DELETE для видалення. Такий підхід забезпечує передбачуваність та однорідність API, а також дозволяє використовувати стандартні інструменти для тестування і документування зокрема автоматично згенеровану специфікацію OpenAPI [18] та інтерфейс Swagger UI, який дозволяє переглядати всі доступні ендпоінти, їхні параметри та формати відповідей безпосередньо в браузері.

Для модуля чатів, де користувачі обмінюються повідомленнями у приватних та групових діалогах, взаємодія також реалізована через REST API: клієнт надсилає повідомлення стандартним HTTP-запитом, а отримання нових повідомлень відбувається через періодичний запит історії чату, що дозволяє уніфікувати весь обмін даними в межах єдиного архітектурного стилю. Усі повідомлення зберігаються в базі даних, що гарантує збереження повної історії листування навіть після завершення сесії користувача чи перезапуску сервера. При вході в чат клієнтська частина завантажує останні повідомлення одним запитом, а нові повідомлення з'являються при оновленні сторінки або повторному запиті, що є достатнім для соціальної мережі такого формату.

Загалом програмний інтерфейс системи реалізовано у вигляді 95 REST-ендпоінтів, розподілених між десятьма функціональними модулями: автентифікації, новин, форуму, користувачів, релізів, чатів (як у клієнтській, так і в адміністративній частинах), спільнот, постів спільнот та службового модуля перевірки працездатності. Зведену структуру API за модулями наведено у таблиці В.11 [Додаток В], а детальний перелік усіх ендпоінтів із зазначенням методу, шляху, опису та рівня доступу у таблицях В.1-В.9, В.11 [Додаток В]. Кожен ендпоінт документований автоматично на основі специфікації OpenAPI [18], що генерується безпосередньо з коду FastAPI, а інтерактивний перегляд та тестування доступні через інтерфейс Swagger UI [24] за адресою /docs.

Для розробки серверної частини було обрано мову розробки Python, тому що його перевагами є: швидкість написання коду, розвиненість екосистеми готових бібліотек та відмінною підтримкою асинхронного програмування, що є

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				20
Змн.	Арк.	№ докум.	Підпис	Дата		

критичним для додатків із багатьма одночасними користувачами. Синтаксис Python дозволяє писати компактний та читабельний код, що особливо важливо для дипломного проєкту, де частина функціоналу розробляється та підтримується однією людиною в обмежені терміни. Крім того, Python має величезну кількість готових бібліотек практично для будь-якої задачі від роботи з базами даних до криптографії та надсилання електронних листів, що дозволяє зосередитися на бізнес-логіці застосунку, а не на написанні допоміжного коду з нуля.

Для розробки серверної частини проєкту було використано вебфреймворк FastAPI, який забезпечує високу швидкість роботи та підтримує асинхронну обробку запитів. Він автоматично перевіряє коректність вхідних даних за допомогою бібліотеки Pydantic і створює документацію для API, що спрощує розробку та тестування. Завдяки підтримці сучасних можливостей Python, FastAPI є зручним рішенням для створення продуктивних вебзастосунків [7].

Доступ до захищених ендпоінтів регулюється через пару JWT-токенів access та refresh [14], які видаються користувачу після успішної реєстрації або входу. Access-токен передається в заголовок кожного захищеного запиту і має обмежений термін дії, тоді як refresh-токен дозволяє отримати новий access-токен без повторного введення облікових даних. Рівень доступу до кожного ендпоінта публічний, для авторизованого користувача, для автора ресурсу, для власника спільноти чи модератора, або для адміністратора визначається на рівні залежностей FastAPI та однозначно зафіксований у відповідному стовпці таблиць В.1-В.9, В.11 [Додаток В] для кожного модуля окремо.

Усі бізнес-винятки, наприклад, спроба зареєструватися з email, який вже використовується, або спроба видалити чужий пост, описані як окремі класи винятків Python, незалежні від HTTP-протоколу. Окремий шар відповідає за перетворення цих винятків у відповідні HTTP-статус-коди та повідомлення для користувача. Такий підхід дозволяє тестувати бізнес-логіку без необхідності піднімати весь веб-сервер, а також гарантує, що користувач завжди отримає зрозуміле повідомлення про помилку рідною мовою, незалежно від того, в якому модулі системи вона виникла.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

Для запуску та стабільної роботи сервера використовується асинхронний ASGI-сервер Uvicorn, який здатний ефективно обробляти тисячі одночасних підключень користувачів. На відміну від традиційних WSGI-серверів, де кожен запит обробляється в окремому потоці або процесі, ASGI-сервер дозволяє обробляти багато запитів у межах одного потоку завдяки асинхронній моделі виконання поки один запит очікує відповіді від бази даних, сервер вже може почати обробку наступного запиту. Це особливо важливо для соціальної мережі, де велика кількість користувачів одночасно надсилає короткі запити (лайки, перегляди новин, повідомлення в чатах), переважна більшість яких задокументована у модулях форуму та новин таблиці В.2 і В.3 [Додаток В] і саме там реалізує операції типу "лайк/дизлайк".

Для зберігання всієї інформації сайту від профілів користувачів до постів на форумах і повідомлень у чатах обрано об'єктно-реляційну систему керування базами даних PostgreSQL [21, 26]. Вона є безкоштовним інструментом промислового рівня, який гарантує повну цілісність даних та високу швидкість виконання складних пошукових запитів завдяки підтримці індексів [4, 20]. PostgreSQL також підтримує розширені типи даних, зокрема перелічувані типи ENUM, які використовуються в проєкті для зберігання ролей користувачів, статусів тем форуму та категорій новин, що забезпечує додатковий рівень контролю цілісності даних на рівні самої бази, а не лише на рівні коду застосунку.

Щоб пов'язати код на Python із базою даних без написання складних ручних SQL-запитів, застосовано бібліотеку SQLAlchemy. Вона виконує роль ORM-системи, тобто перетворює таблиці бази даних у звичайні об'єкти та класи у коді [27]. Кожна таблиця бази даних описується як окремий клас-модель з атрибутами, що відповідають колонкам таблиці, а зв'язки між таблицями, наприклад, один користувач може мати багато повідомлень, описуються декларативно через відношення між класами, що робить структуру даних наочною і легкою для розуміння навіть без постійного звернення до самої бази даних.

Для того щоб робота з базою даних не гальмувала загальну роботу сервера, SQLAlchemy налаштована в асинхронному режимі спільно зі спеціалізованим

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

драйвером `asynsrg`. Це дозволяє серверу відправляти запит до бази даних і, не чекаючи відповіді, одразу обробляти запити інших користувачів, що суттєво підвищує загальну швидкість роботи сайту під навантаженням. Для контролю змін структури бази даних у проєкті використовується інструмент Alembic [30], який дозволяє описувати кожну зміну схеми як окрему версійну міграцію. Це гарантує, що структура бази даних на сервері розробки, тестовому та робочому серверах завжди узгоджена між собою, а будь-яку зміну можна як застосувати, так і скасувати в контрольований спосіб.

Клієнтська частина соціальної мережі будується на основі фреймворку Next.js, що базується на бібліотеці React і дозволяє створювати швидкі та динамічні інтерфейси з модульною структурою [15, 23]. Весь вигляд сайту розбивається на окремі компоненти (наприклад, картка новини, кнопка лайка, вікно чату), які можна використовувати повторно в різних частинах проєкту, що спрощує підтримку та оновлення коду. Завдяки компонентному підходу зміни в одному місці інтерфейсу (наприклад, у вигляді кнопки лайка) автоматично застосовуються на всіх сторінках, де цей компонент використовується, без необхідності вручну редагувати кожну сторінку окремо.

Для взаємодії з 95 ендпоінтами бекенду, описаними у Додатку В, на клієнтській частині використовується єдиний налаштований екземпляр HTTP-клієнта `axios` [3], який автоматично додає до кожного запиту JWT access-токен та виконує його оновлення через refresh-токен у разі завершення терміну дії, відповідно до маршрутів модуля «Автентифікація» таблиці В.1 [Додаток В]. Next.js забезпечує вбудовану систему маршрутизації на основі файлової структури, оптимізацію зображень та статичних ресурсів, а також ефективну збірку готового коду перед публікацією на сервер, що дозволяє сторінкам FanPulse завантажуватися швидко навіть на мобільних пристроях чи при слабкому інтернет-з'єднанні.

Для написання клієнтської частини використовується мова TypeScript [29], що додає до JavaScript статичну типізацію кожної змінної, функції та об'єкта даних мають заздалегідь визначений тип, що дозволяє виявляти значну частину помилок ще на етапі написання коду, до запуску застосунку в браузері. Зокрема, типи

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		23

відповідей кожного з ендпоінтів, перелічених у Додатку В [Додаток В], описані у клієнтській частині як TypeScript-інтерфейси, що гарантує відповідність між очікуваною структурою даних на фронтенді та фактичною структурою JSON-відповіді бекенду.

Для зберігання глобального стану застосунку, зокрема даних про поточного авторизованого користувача, використовується бібліотека Zustand [12], яка є простою у використанні та є альтернативою більш складним рішенням для управління станом та дозволяє будь-якому компоненту інтерфейсу отримати доступ до даних користувача без передачі їх через довгі ланцюжки властивостей компонентів. Дані про користувача, що зберігаються у глобальному стані, надходять з ендпоінта GET /auth/me таблиці В.1 [Додаток В], який викликається одразу після відновлення сесії за збереженим токеном.

Для стилізації користувацького інтерфейсу було використано CSS-фреймворк Tailwind CSS [11], який дозволяє стилізувати елементи інтерфейсу безпосередньо в розмітці компонентів за допомогою готових класів, без необхідності писати окремі файли стилів для кожного елемента. Такий підхід прискорює розробку інтерфейсу та забезпечує візуальну узгодженість усіх сторінок застосунку, оскільки всі розробники користуються тим самим обмеженим набором значень для відступів, кольорів, розмірів шрифтів тощо.

Важливим аспектом проєкту є підхід до архітектури коду всередині самого бекенду, де реалізовано патерн «Репозиторій». Замість того, щоб писати логіку роботи з базою даних прямо всередині обробників маршрутів FastAPI, всі операції з таблицями винесені в окремі класи-репозиторії. Це спрощує структуру коду, полегшує тестування та підтримку окремих модулів, ізолює бізнес-логіку від технічних деталей збереження даних та дозволяє легко змінювати або тестувати окремі модулі без ризику зламати роботу всього сайту.

Поверх шару репозиторіїв розташований шар сервісів, який відповідає за бізнес-логіку застосунку, а саме: перевірку прав доступу користувача, валідацію бізнес-правил, а також координацію кількох операцій з різними репозиторіями в межах однієї транзакції. Обробники маршрутів FastAPI, у свою чергу,

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

відповідають лише за прийом запиту, виклик відповідного методу сервісу та формування HTTP-відповіді, не містячи жодної бізнес-логіки самостійно. Таке Трирівневе розділення (маршрути – сервіси – репозиторії) всередині самого бекенду є додатковим прикладом застосування принципу єдиної відповідальності та значно полегшує читання та супровід коду великого проєкту. Саме на цьому рівні реалізується перевірка прав доступу, зафіксована у стовпці «Доступ» для кожного з 95 ендпоінтів наприклад, відмінність між роллю «Автор» та «Адмін» під час видалення поста форуму таблиці Г.3.

Для управління сесіями та транзакціями бази даних використовується механізм впровадження залежностей, вбудований у FastAPI. Він автоматично створює підключення до бази даних на початку кожного запиту користувача, передає його в потрібний репозиторій, а після успішного виконання запиту закриває з'єднання, запобігаючи витoku пам'яті та забезпечуючи стабільну автономну роботу всієї системи. Той самий механізм впровадження залежностей використовується для перевірки авторизації користувача: залежність отримує JWT-токен із заголовка запиту, перевіряє його дійсність та повертає об'єкт поточного користувача, який потім доступний у будь-якому обробнику маршруту, що його потребує, без дублювання коду перевірки токена в кожному окремому ендпоінті. Завдяки такій уніфікації один і той самий механізм перевірки застосовується для всіх 95 ендпоінтів, описаних у [Додатку В] від публічних маршрутів модуля новин до повністю захищених маршрутів адміністрування чатів.

1.4 Обґрунтування вибору програмних засобів розробки

Вибір програмних засобів для реалізації інформаційної системи є одним із ключових етапів проєктування, оскільки саме технологічний стек визначає продуктивність, масштабованість, надійність та подальші можливості розвитку програмного продукту. Під час розробки соціальної платформи FanPulse особлива увага приділялася таким критеріям, як швидкість обробки запитів, підтримка великої кількості одночасних користувачів, безпека зберігання даних, зручність подальшого супроводу та відповідність сучасним тенденціям веброзробки. Враховуючи ці вимоги, для реалізації серверної частини було обрано мову

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				25
Змн.	Арк.	№ докум.	Підпис	Дата		

програмування Python та вебфреймворк FastAPI, для зберігання даних – систему керування базами даних PostgreSQL, а для створення клієнтської частини – бібліотеку React у поєднанні з фреймворком Next.js [8, 16].

Мова програмування Python була обрана як основний інструмент реалізації серверної логіки завдяки поєднанню високої продуктивності розробки, простоти підтримки коду та наявності великої кількості готових бібліотек. На відміну від мов програмування нижчого рівня, Python характеризується лаконічним синтаксисом, що дозволяє скоротити час розробки та зменшити кількість потенційних помилок у програмному коді. Важливою перевагою є також підтримка асинхронного програмування через механізми `async/await`, які дають змогу ефективно працювати з великою кількістю одночасних мережових запитів.

Для соціальної мережі така можливість є особливо важливою, оскільки користувачі постійно виконують різні дії: переглядають стрічку новин, публікують повідомлення, ставлять вподобання, створюють коментарі, надсилають повідомлення в чатах та отримують сповіщення. У разі використання традиційного синхронного підходу сервер був би змушений послідовно обробляти кожен запит, що призводило б до збільшення затримок під час високого навантаження. Асинхронна модель Python дозволяє виконувати операції введення-виведення паралельно, значно підвищуючи пропускну здатність системи без необхідності суттєвого збільшення апаратних ресурсів.

Додатковою перевагою Python є його популярність у сфері веброзробки та велика спільнота розробників. Завдяки цьому існує значна кількість перевірених бібліотек для реалізації аутентифікації, роботи з базами даних, шифрування інформації, тестування та автоматизації процесів розгортання. Наявність таких інструментів дозволяє прискорити розробку програмного продукту та зосередитися на реалізації бізнес-логіки платформи замість створення допоміжних механізмів з нуля.

Для побудови серверної частини системи було обрано вебфреймворк FastAPI. Дане рішення пояснюється тим, що FastAPI поєднує високу швидкодію, підтримку асинхронної архітектури та сучасний підхід до проектування

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				26
Змн.	Арк.	№ докум.	Підпис	Дата		

вебсервісів. На відміну від фреймворку Django, який містить велику кількість вбудованих модулів та орієнтований на створення монолітних вебзастосунків, FastAPI є більш легковаговим і краще підходить для реалізації серверів, побудованих за принципом REST API [24, 31].

Порівняно з Flask, який також є популярним рішенням для створення вебзастосунків, FastAPI має ряд суттєвих переваг. Насамперед це повноцінна підтримка асинхронності на рівні архітектури фреймворку. Крім того, FastAPI активно використовує механізми типізації Python, що дозволяє автоматично перевіряти правильність даних, які надходять від клієнта. Це значно підвищує надійність роботи системи та зменшує ризик виникнення помилок через некоректні вхідні параметри.

Однією з ключових особливостей FastAPI є інтеграція з бібліотекою Pydantic. Вона забезпечує автоматичну валідацію, серіалізацію та десеріалізацію даних відповідно до визначених моделей [22]. Кожен запит користувача перевіряється на відповідність заданій структурі ще до початку виконання бізнес-логіки. У результаті система автоматично відхиляє некоректні або потенційно небезпечні дані, що позитивно впливає на стабільність роботи та рівень безпеки програмного забезпечення.

Додатковою перевагою FastAPI є автоматична генерація документації API на основі специфікації OpenAPI із використанням інтерфейсу Swagger UI. Це значно спрощує процес тестування та інтеграції клієнтської частини із сервером. Розробники можуть переглядати доступні маршрути, структури запитів і відповідей, а також виконувати тестові виклики безпосередньо через браузер. Такий підхід прискорює процес розробки та спрощує супровід системи на наступних етапах її життєвого циклу.

Для зберігання інформації було обрано систему керування базами даних PostgreSQL. Основною причиною такого вибору є необхідність забезпечення цілісності складної структури взаємопов'язаних даних. Соціальна мережа FanPulse оперує великою кількістю сутностей, серед яких користувачі, профілі, публікації, коментарі, лайки, клани, форуми, повідомлення та інші об'єкти. Між цими

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

сутностями існують численні зв'язки типу «один до багатьох» та «багато до багатьох», які потребують надійного механізму контролю цілісності даних.

PostgreSQL є однією з найпоширеніших реляційних СУБД у сучасних інформаційних системах. Вона підтримує транзакції, механізми забезпечення цілісності даних, індексування, складні SQL-запити та ефективну роботу з великими обсягами інформації. Завдяки відкритому вихідному коду система активно розвивається та використовується як у невеликих проєктах, так і в корпоративних інформаційних системах.

Важливою перевагою PostgreSQL є можливість масштабування та підтримка розширених механізмів оптимізації запитів. Це дозволяє ефективно працювати навіть за умов значного зростання кількості користувачів та накопичення великих обсягів даних. Крім того, PostgreSQL забезпечує високий рівень надійності завдяки використанню журналювання транзакцій та механізмів резервного копіювання.

Для взаємодії між серверною частиною та базою даних використовується ORM-бібліотека SQLAlchemy. Її застосування дозволяє працювати з таблицями бази даних через об'єкти Python, не використовуючи безпосередньо SQL-запити в кожній частині програми. Такий підхід підвищує читабельність коду, зменшує кількість повторюваних конструкцій та спрощує супровід системи.

Додатково використовується драйвер asynpcpg, який забезпечує асинхронну взаємодію з PostgreSQL. Завдяки цьому операції читання та запису даних не блокують виконання інших серверних процесів. У результаті система може одночасно обслуговувати значно більшу кількість користувачів без суттєвого зниження продуктивності.

У бекенді FanPulse використовується патерн проєктування «Репозиторій». Патерн створює окремий шар доступу до даних, який відокремлює бізнес-логіку від роботи зі сховищем інформації, який ізолює бізнес-логіку від конкретної реалізації сховища інформації. Усі операції взаємодії з базою даних зосереджуються всередині спеціалізованих класів-репозиторіїв.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

Використання такого підходу забезпечує декілька важливих переваг. По-перше, покращується модульність системи, оскільки бізнес-логіка не залежить від конкретної структури бази даних. По-друге, значно спрощується тестування програмного забезпечення завдяки можливості підміняти реальні репозиторії тестовими реалізаціями. По-третє, зменшується дублювання коду, оскільки всі операції доступу до даних реалізуються централізовано. Це позитивно впливає на підтримуваність та розширюваність програмного продукту.

Для реалізації клієнтської частини платформи було обрано бібліотеку React. Дане рішення обґрунтовується її компонентною архітектурою, високою продуктивністю та широким використанням у сучасній веброботі. React дозволяє розділяти інтерфейс на незалежні компоненти, кожен з яких відповідає за власну логіку та відображення інформації. Такий підхід спрощує повторне використання коду та пришвидшує розробку нових функціональних можливостей.

Virtual DOM оновлює лише ті елементи інтерфейсу, дані яких змінилися, без повного перезавантаження сторінки після кожної дії користувача. За додавання нового коментаря або оновлення лічильника вподобань перерисовується відповідний компонент, що забезпечує швидкий та плавний інтерфейс, подібний до роботи настільних або мобільних застосунків.

Для подальшого розвитку клієнтської частини використовується фреймворк Next.js, який базується на React та розширює його можливості. Використання Next.js дозволяє автоматизувати процес маршрутизації, організувати структуру проєкту відповідно до сучасних стандартів та підвищити продуктивність вебзастосунку. Вбудована система маршрутизації App Router забезпечує логічний поділ сторінок та спрощує навігацію між різними розділами платформи.

Серед додаткових переваг Next.js слід відзначити оптимізацію процесу збірки застосунку, підтримку серверного рендерингу та покращене завантаження сторінок. Це позитивно впливає як на швидкість роботи вебресурсу, так і на зручність його використання кінцевими користувачами.

Для взаємодії між клієнтською та серверною частинами використовується архітектурний стиль REST API. Його застосування дозволяє стандартизувати

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

обмін даними між усіма модулями системи та забезпечити незалежність клієнтської частини від внутрішньої реалізації серверної логіки. Кожен ресурс системи має власний набір HTTP-методів, що відповідають операціям створення, читання, оновлення та видалення даних.

Для виконання HTTP-запитів у клієнтській частині використовується бібліотека Axios. Її застосування забезпечує централізовану обробку помилок, автоматичне додавання токенів авторизації та можливість реалізації механізму оновлення JWT-токенів. Завдяки цьому спрощується взаємодія між клієнтом і сервером та підвищується рівень безпеки системи.

Механізм автентифікації реалізовано на основі технології JSON Web Token (JWT). Після успішного входу в систему користувач отримує спеціальний токен, який передається разом із кожним захищеним запитом. Сервер перевіряє дійсність токена та визначає права доступу користувача до відповідних ресурсів. Такий підхід дозволяє реалізувати безпечну та масштабовану систему контролю доступу без необхідності постійного зберігання інформації про активні сесії на сервері.

Отже, обраний технологічний стек повністю відповідає вимогам, поставленим до соціальної платформи FanPulse. Поєднання Python, FastAPI, PostgreSQL, SQLAlchemy, React та Next.js забезпечує високу продуктивність, масштабованість, безпеку та зручність подальшого розвитку програмного забезпечення. Використання сучасних архітектурних підходів і перевірених технологій створює надійну основу для стабільної роботи системи навіть за умов значного навантаження та подальшого розширення функціональних можливостей платформи.

Висновки до першого розділу

У першому розділі було проведено комплексний аналіз предметної області, визначено ключові функціональні вимоги до системи, а також обґрунтовано вибір архітектурних паттернів та програмних засобів для реалізації спеціалізованої соціальної мережі для геймерських спільнот FanPulse.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

На основі проведеного порівняльного аналізу існуючих ринкових аналогів, таких як Discord та Twitch, було виявлено архітектурні та функціональні обмеження у сфері структурування публічного контенту та організації відкритого інформаційного простору. Це дозволило чітко сформулювати постановку задачі на проєктування сервісу FanPulse як унікального програмного продукту, що поєднує в собі переваги тематичних форумів, загальної новинної стрічки з гнучкою системою фільтрації та унікального інструменту внутрішньокланової активності користувачів.

Для забезпечення високої продуктивності, масштабованості та відмовостійкості платформи було обрано трирівневу клієнт-серверну архітектуру. В межах проєктування серверної частини обґрунтовано доцільність застосування мови програмування Python та сучасного асинхронного вебфреймворку FastAPI, який надає інструменти автоматичної валідації даних та генерації документації. З метою забезпечення цілісності великого масиву взаємопов'язаних даних та високої швидкості обробки складних транзакцій обрано реляційну СУБД PostgreSQL, робота з якою реалізується через асинхронний режим ORM-системи SQLAlchemy.

Для забезпечення взаємодії між клієнтом і сервером, включно з модулем чатів, обрано єдиний архітектурний стиль REST API, що забезпечує уніфіковану модель обробки запитів і спрощує підтримку системи. На рівні внутрішньої архітектури коду впроваджено патерн «Репозиторій», який дозволяє ізолювати бізнес-логіку застосунку від безпосереднього рівня збереження даних, що підвищує чистоту коду, полегшує тестування та супровід. Клієнтську частину реалізовано на базі фреймворку Next.js, який забезпечує створення швидкого та адаптивного інтерфейсу з підтримкою серверного рендерингу та SEO-оптимізації, а також зменшує навантаження на користувача завдяки ефективному оновленню сторінок.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ЗАСТОСУНКУ ТА ВИМОГИ ДО ФУНКЦІОНАЛУ

2.1 Вимоги до системи

Проектування соціальної мережі для геймерів FanPulse вимагає чіткого визначення та деталізації системних вимог, які поділяються на функціональні та нефункціональні. Функціональні вимоги описують безпосередні можливості користувачів та адміністраторів у межах платформи, визначаючи логіку поведінки програмного забезпечення у відповідь на їхні дії. Нефункціональні вимоги задають технічні обмеження та стандарти якості, яким повинна відповідати система, включаючи рівень продуктивності, безпеки, доступності та зручності використання інтерфейсу. Усі ці критерії закладаються на початковому етапі проектування, щоб забезпечити стабільну роботу сервісу при одночасному підключенні великої кількості активних користувачів.

Головна функціональна вимога до системи полягає в реалізації дворівневої моделі доступу до вмісту сайту, яка включає гостьовий режим та режим авторизованого користувача. Неавторизовані гості платформи повинні мати право на ознайомлення з публічною інформацією. Програмна логіка має забезпечувати для них можливість перегляду стрічки новин, вивчення інтерактивного календаря релізів відеоігор із використанням фільтрів, а також читання форумних гілок обговорення та публічних коментарів. Водночас система повинна блокувати будь-які спроби змінити дані в базі з боку гостей, обмежуючи для них можливість писати повідомлення, виставляти реакції, створювати нові теми, вступати до спільнот та голосувати в опитуваннях до моменту успішного проходження реєстрації. Такий підхід забезпечує відкритість платформи для нових відвідувачів і водночас захищає цілісність даних від несанкціонованого запису.

Для зареєстрованих користувачів система має надавати повний набір інструментів для взаємодії з платформою та іншими учасниками. Насамперед, система повинна підтримувати функціонал керування особистим кабінетом, де користувач може завантажити нове зображення для профілю, відредагувати текстову інформацію про себе та змінити поточний пароль. Реєстрація нового

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

акаунту здійснюється через форму з перевіркою електронної пошти, на яку надсилається лист з підтверджуючим посиланням. До отримання підтвердження доступ до функцій платформи залишається обмеженим. Скидання забутого пароля реалізоване аналогічно через електронну пошту з одноразовим токеном, що має обмежений термін дії.

У межах модуля новин та форумів система повинна коректно обробляти додавання нових публікацій, коментарів та виставлення реакцій. Коментарі на форумі підтримують вкладену структуру відповідей, тобто кожне повідомлення може стати початком окремого під-обговорення. Користувачі повинні мати змогу підписуватися на оновлення конкретних тем форуму, щоб отримувати сповіщення про нові відповіді. Підписка на тему зберігається в базі даних і може бути скасована у будь-який момент. Крім того, система повинна підтримувати можливість поскаржитися на конкретне повідомлення, після чого скарга потрапляє до черги модераторів.

Функціонал спільнот дозволяє будь-якому зареєстрованому користувачу створити власну тематичну групу з назвою, описом та зображеннями. Всередині кожної спільноти діє власна ієрархія ролей: власник, адміністратор, модератор і учасник. Кожна роль має строго визначені права в межах спільноти. Учасники можуть публікувати пости та коментарі всередині спільноти, ставити вподобання та взаємодіяти між собою. Система повинна підтримувати механізм банів на рівні спільноти, що не залежить від глобальних обмежень на рівні платформи. Для спілкування передбачено чати, де авторизовані користувачі можуть вести групові бесіди.

Окремий деталізований набір функціональних вимог висувається до адміністраторського інтерфейсу. Панель адміністрування повинна функціонувати як центральний вузол управління сайтом. В першу чергу тут реалізується аналітичний дашборд для моніторингу загальної статистики активності: кількості зареєстрованих користувачів, нових публікацій за день, активних чатів та переглядів новин. Ці дані надають адміністратору поточний зріз стану платформи без необхідності формувати складні ручні запити до бази даних.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

Система має надавати адміністратору повні права на модерацію вмісту: додавання, редагування та видалення новинних матеріалів, записів у календарі релізів, форумних тем та окремих коментарів. На рівні контролю за порядком у спільноті система повинна підтримувати застосування дисциплінарних заходів: видалення облікових записів, повна заборона на вхід (бан), обмеження права надсилання повідомлень (мут), а також зміна ролей користувачів. Для підтримки чистоти контенту адміністратор управляє динамічним словником заборонених слів. Кожна нова публікація на платформі автоматично перевіряється за цим словником ще до запису в базу даних, і при виявленні забороненого вмісту публікація блокується з поверненням відповідного повідомлення.

Вказані функціональні можливості адміністратора, а також сценарії взаємодії гостей і зареєстрованих користувачів із системою, узагальнено на діаграмі варіантів використання, наведеній на рисунку 2.1. Вона наочно демонструє основні ролі в системі FanPulse та їхні ключові дії, що забезпечує цілісне розуміння архітектури платформи.

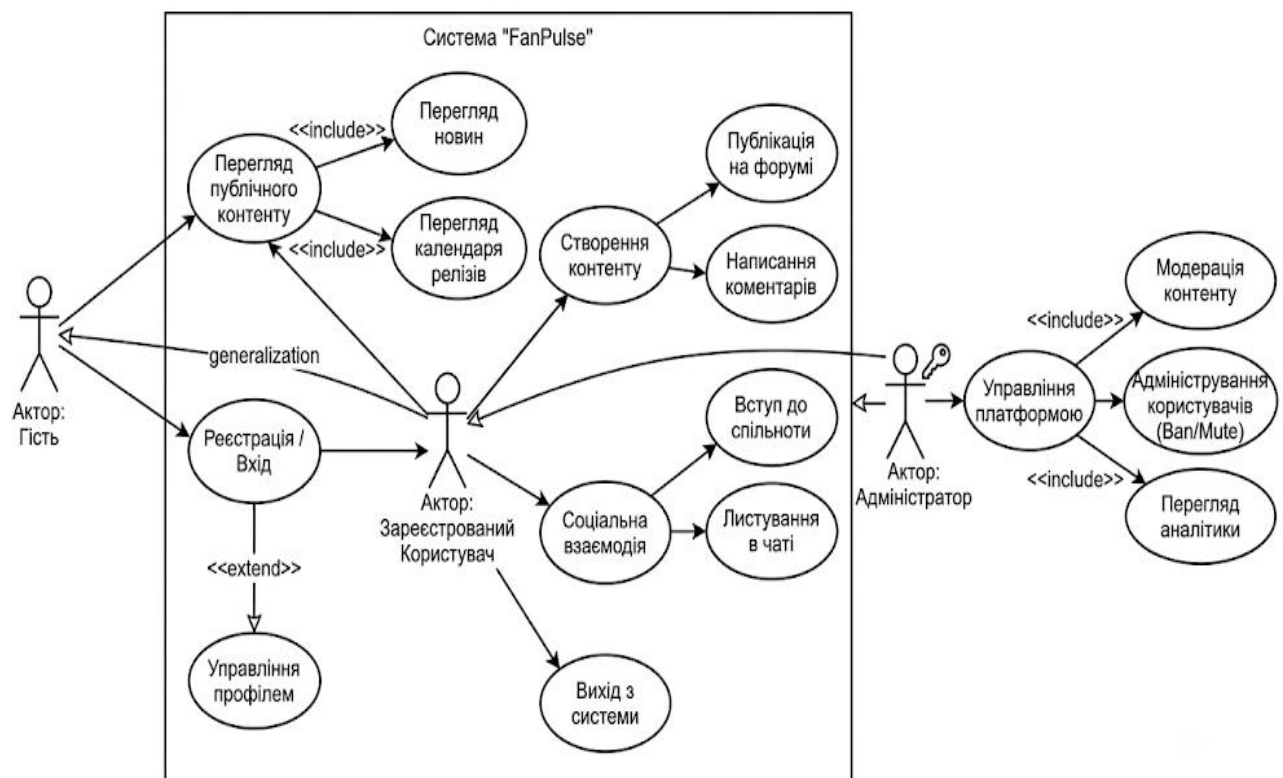


Рис. 2.1. Діаграма використання системи

Для деталізації основних сценаріїв взаємодії користувачів із системою FanPulse було сформовано набір прецедентів використання. Кожен прецедент описує конкретний функціональний сценарій роботи платформи, визначає учасників процесу та коротко характеризує їхні дії в межах системи. Найважливіші варіанти використання, що відображають базову функціональність платформи, наведено в таблицях 2.1–2.3.

У таблиці 2.1 представлено опис прецеденту «Перегляд публічного контенту», який характеризує можливості доступу до відкритої інформації для всіх категорій користувачів.

Таблиця 2.1

Опис прецеденту «Перегляд публічного контенту»

Назва варіанту використання	Перегляд публічного контенту
Основні актори	Гість, Зареєстрований користувач, Адміністратор
Короткий опис	Гість, зареєстрований користувач та адміністратор мають можливість переглядати стрічку новин та календар релізів для ознайомлення з публічною інформацією системи.

У таблиці 2.2 наведено опис прецеденту «Управління профілем», що визначає функції персоналізації облікового запису та керування особистими даними користувача.

Таблиця 2.2

Опис прецеденту «Управління профілем»

Назва варіанту використання	Управління профілем
Основні актори	Зареєстрований користувач, Адміністратор
Короткий опис	Зареєстрований користувач або адміністратор має право редагувати дані свого профілю, зокрема змінювати нікнейм, пароль та фотографію (аватар) для персоналізації свого облікового запису.

У таблиці 2.3 подано опис прецеденту «Публікація на форумі», який відображає процес створення нових обговорень і взаємодії учасників через систему коментарів.

Опис прецеденту «Публікація на форумі»

Назва варіанту використання	Публікація на форумі
Основні актори	Зареєстрований користувач, Адміністратор
Короткий опис	Зареєстрований користувач або адміністратор має право створювати нові обговорення та публікувати коментарі на форумі для взаємодії з іншими учасниками.

З технічної точки зору, до вебплатформи висуваються жорсткі нефункціональні вимоги щодо продуктивності та швидкодії. Оскільки серверна частина проєкту базується на асинхронній архітектурі FastAPI з ASGI-сервером Uvicorn, час відповіді сервера на стандартні HTTP-запити не повинен перевищувати кількості мілісекунд, що гарантує плавне відображення даних у браузері. Модуль чатів повинен забезпечувати оперативну доставку повідомлень учасникам діалогу та відповідати тим самим вимогам до часу відповіді сервера, що й решта REST-ендпоінтів платформи. Система повинна демонструвати стійкість до відмов та здатність обробляти одночасні запити від великої кількості відвідувачів без падіння продуктивності, що досягається оптимізацією PostgreSQL та асинхронним драйвером asyncpg.

Вимоги до надійності роботи включають обов'язкове збереження хронологічної історії повідомлень у чатах, щоб після повторного входу в систему користувач завжди міг переглянути попереднє листування. Усі операції з базою даних виконуються в межах транзакцій, що гарантує відсутність частково записаних або пошкоджених даних у разі помилки під час виконання запиту. Система управління міграціями Alembic гарантує, що схема бази даних завжди відповідає поточній версії програмного коду, що особливо важливо при оновленні розгорнутого продукту.

Вимоги до безпеки та надійності зберігання інформації є пріоритетними при проєктуванні архітектури FanPulse. Усі паролі користувачів перед збереженням у базу даних підлягають обов'язковому криптографічному хешуванню за алгоритмом bcrypt на стороні сервера, що робить неможливим їхнє відновлення у

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

відкритому вигляді навіть у разі компрометації сховища. Сесії авторизованих користувачів реалізовані через пару JWT-токенів: короткоживучий access token та довгоживучий refresh token, що зберігається у базі даних і може бути відкликаний адміністратором у будь-який момент [14]. Система повинна забезпечувати захист від поширених мережесих вразливостей та виконувати суворе валідування всіх вхідних даних на рівні фреймворку FastAPI за допомогою Pydantic-схем [22, 25]. Будь-які дані, що надходять від клієнта, перевіряються на відповідність очікуваному формату ще до передачі в бізнес-логіку.

Вимоги до зручності використання передбачають, що інтерфейс платформи повинен коректно відображатися як на настільних комп'ютерах, так і на мобільних пристроях. Усі форми на сайті повинні повертати зрозумілі повідомлення про помилки при некоректному заповненні. Операції, що потребують тривалого часу обробки, наприклад завантаження та обробка зображень, не повинні блокувати інтерфейс користувача. Загальна навігація по платформі має бути інтуїтивно зрозумілою і не вимагати попереднього знайомства з документацією.

Підсистема автоматичної фільтрації контенту повинна сканувати кожен новий коментар або пост на наявність забороненої лексики зі словника ще до запису в базу даних. Якщо текст містить заборонені слова, система повертає помилку з відповідним повідомленням і не зберігає публікацію. Адміністратор може у будь-який час додати нові слова до словника або видалити застарілі записи, і ці зміни одразу набирають чинності без перезапуску сервера, оскільки перевірка виконується динамічно через запит до бази даних при кожній публікації.

2.2 Моделювання архітектури та функціональних можливостей системи

Проектування архітектури соціальної мережі FanPulse стало одним із ключових етапів розробки, адже саме на цьому рівні визначається логіка взаємодії між усіма компонентами системи та формується основа для подальшої реалізації інтерфейсу й механізмів безпеки. Архітектура будувалася з урахуванням принципів модульності, гнучкості та продуктивності, що дозволяє платформі

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

адаптуватися до зростання аудиторії та стабільно працювати під пікових навантажень.

Для реалізації поставлених завдань обрано архітектурний шаблон «Клієнт-Сервер», який забезпечує чітке розділення відповідальності між клієнтським застосунком та серверним API. Клієнтська частина відповідає за відображення інтерфейсу та взаємодію з користувачем, а серверна реалізує бізнес-логіку, обробку запитів і роботу з базою даних. Такий підхід дозволяє незалежно масштабувати фронтенд і бекенд та підвищує стійкість системи до навантажень. Якщо кількість користувачів різко зростає, достатньо запустити ще один екземпляр бекенду без будь-яких змін у клієнтській частині.

Бекенд FanPulse побудований за принципом трирівневої архітектури, де кожен рівень відповідає за окремий набір завдань. Рівень представлення реалізований через API-маршрути FastAPI та відповідає за обробку HTTP-запитів. Рівень бізнес-логіки містить сервіси, які виконують основні операції та координують роботу застосунку. Рівень доступу до даних складається з репозиторіїв і моделей SQLAlchemy, які забезпечують взаємодію з базою даних PostgreSQL. Такий підхід дозволяє розділити відповідальність між компонентами системи, спростити підтримку коду та полегшити його тестування.

Основну бізнес-логіку платформи зосереджено на рівні сервісів, який функціонує над шаром репозиторіїв. Тут виконується перевірка повноважень користувачів, контроль бізнес-обмежень і координація операцій, що задіюють кілька репозиторіїв одночасно. Наприклад, при публікації теми на форумі сервісний шар перевіряє користувача на наявність блокувань, а текст на відповідність правилам модерації, і лише після цього ініціює запис у базу даних. Взаємодію із зовнішніми запитами забезпечують ендпоінти FastAPI, які виконують первинну автентифікацію та передають дані на сервіси. Таке архітектурне розділення мінімізує зв'язність компонентів, полегшує масштабування системи та дозволяє ізольовано змінювати логіку роботи застосунку.

Механізми автентифікації та авторизації у системі базуються на використанні двотокенної схеми стандарту JWT. У разі успішної автентифікації сервер

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

генерує короткочасний access-токен та довготривалий refresh-токен, хеш якого зберігається у таблиці refresh_tokens бази даних. Це дозволяє перевіряти дійсність токена без зберігання його у відкритому вигляді. Після завершення терміну дії access-токена клієнт надсилає refresh-токен для отримання нової пари токенів без повторної автентифікації. Для забезпечення безпеки реалізовано механізм відкликання refresh-токенів шляхом помітки запису в базі даних як неактивного. Валідація доступу до захищених ендпоінтів здійснюється через access-токен у заголовку Authorization, а розподіл прав визначається ролями користувачів.

Система спроектована таким чином, щоб задовольнити потреби різних категорій користувачів через чітко визначені функціональні модулі. Перший модуль відповідає за управління користувачами та включає реєстрацію з підтвердженням електронної пошти, вхід у систему, оновлення профілю, зміну та скидання пароля. Реєстрація не завершується одразу після заповнення форми, а вимагає переходу за посиланням у листі, що захищає від створення акаунтів на чужу адресу. Скидання пароля також реалізоване через електронну пошту з одноразовим токеном з обмеженим терміном дії.

Окремий модуль платформи призначений для роботи з новинами. Адміністратори можуть створювати та публікувати статті, додаючи до них заголовки, короткий опис, основний текст, обкладинку, категорію та теги. Для матеріалів передбачено систему вподобань і коментарів, а також збір статистики переглядів. Унікальні перегляди враховуються як для авторизованих користувачів за їхнім ідентифікатором, так і для гостей за допомогою поля viewer_key. Наявність поля is_published дозволяє підготувати матеріал заздалегідь і опублікувати його в потрібний момент.

Для організації спілкування користувачів у системі передбачено форум із багаторівневою структурою. Основу форуму складають категорії, всередині яких створюються окремі теми для обговорення. Кожна тема містить повідомлення, що підтримують вкладені відповіді через поле parent_id, завдяки чому можна створювати окремі гілки дискусій. Користувачі можуть оцінювати повідомлення та підписуватися на теми, а модератори й адміністратори мають додаткові

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

можливості для закріплення, блокування та модерування вмісту. Також реалізовано систему скарг, що дозволяє повідомляти про порушення правил спільноти.

Для об'єднання користувачів за інтересами на платформі реалізовано систему спільнот. Авторизований користувач може створити власну спільноту, додавши назву, опис, зображення та банер. У середині кожної спільноти діє окрема система ролей, що визначає права учасників. Наприклад, модератор може видаляти публікації, але не має доступу до зміни основних налаштувань. Публікації підтримують коментарі та вподобання, а також передбачено незалежну систему банів для окремих спільнот.

Засоби особистого спілкування представлені модулем чатів. Користувачі можуть створювати особисті та групові чати, причому кількість учасників групового чату не обмежується. Для кожного учасника можуть бути визначені окремі права доступу. Повідомлення зберігаються в базі даних, що забезпечує доступ до історії листування після повторного входу в систему. Обмін повідомленнями виконується через REST API за допомогою відповідних ендпоінтів.

Ще однією складовою платформи є інтерактивний календар ігрових релізів. Адміністратори додають інформацію про нові ігри, зазначаючи опис, дату виходу, статус, жанри та підтримувані платформи. Зареєстровані користувачі можуть оформити підписку на релізи, що їх цікавлять, для подальшого отримання сповіщень. Також реалізовано систему фільтрації за жанрами, платформами та статусом релізу.

Керування всією платформою здійснюється через адміністративний модуль. Він надає можливість переглядати список користувачів, змінювати їхні ролі, блокувати акаунти та контролювати наповнення системи. Додатково адміністратори можуть керувати категоріями форуму, розглядати скарги користувачів і працювати зі словником заборонених слів, який автоматично використовується під час перевірки опублікованого контенту.

Проектування функціоналу велось з урахуванням концепції користувацьких сценаріїв, де для кожного типу дії визначено очікувану реакцію системи. Наприклад, при створенні нової теми на форумі система перевіряє права доступу,

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				40
Змн.	Арк.	№ докум.	Підпис	Дата		

перевіряє текст на наявність забороненої лексики зі словника та зберігає запис у базі даних. При додаванні коментаря клієнт отримує підтвердження через асинхронну обробку запиту. При вході в систему після перевірки облікових даних видається пара JWT-токенів. Кожна дія, пов'язана з модерацією, фіксується у відповідних таблицях, що дозволяє відстежувати історію змін.

Важливим аспектом архітектурного рішення є підтримка горизонтального масштабування. Оскільки стан сесії зберігається у базі даних через refresh token, а не у пам'яті сервера, можна запустити кілька паралельних екземплярів бекенду за балансувальником навантаження без проблем із синхронізацією сесій. Для оптимізації швидкості відповідей на часто повторювані запити передбачено можливість підключення кешування на рівні сервісного шару. Асинхронна природа FastAPI та asynсpg дозволяє ефективно використовувати ресурси одного серверного вузла навіть при значному навантаженні.

Таким чином, спроектована архітектура FanPulse являє собою логічно впорядковану, модульну систему, де кожен компонент виконує одну чітко визначену роль. Шарова організація бекенду, розподіл на функціональні модулі та стандартизована REST API разом забезпечують платформу, яка є зручною у розробці, легко розширюється новим функціоналом та здатна стабільно працювати при зростанні аудиторії.

2.3 Розроблення структури бази даних

Під час розробки соціальної мережі для фанатів комп'ютерних ігор FanPulse особливу увагу було приділено проектуванню бази даних, оскільки саме вона є основним компонентом, що забезпечує надійне зберігання, структурування та оперативну обробку усіх відомостей про користувачів та їхню взаємодію з платформою. База даних виступає фундаментом системи, адже від коректності її архітектурних рішень безпосередньо залежить узгодженість роботи програмних модулів, швидкість виконання запитів та потенційна можливість подальшого горизонтального і вертикального масштабування. Для реалізації проєкту обрано класичну реляційну модель даних, а як систему керування базами даних використано PostgreSQL. Вибір цієї промислової СУБД обумовлений її високою

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				41
Змн.	Арк.	№ докум.	Підпис	Дата		

продуктивністю під навантаженням, підтримкою складних зв'язків між сутностями, гнучкими механізмами забезпечення цілісності даних, відповідністю вимогам ACID та широкою інтеграцією із сучасними засобами асинхронної веб-розробки.

Логічна структура бази даних платформи FanPulse сформована відповідно до функціональних вимог системи та поділена на окремі взаємопов'язані модулі. Такий підхід спрощує підтримку програмного забезпечення у стадії експлуатації та дозволяє розширювати систему шляхом додавання нових компонентів без порушення цілісності існуючих зв'язків. Кожен модуль відповідає за конкретну бізнес-логіку платформи і містить профільні таблиці, що забезпечують персистентне зберігання даних та їхню внутрішню консистентність. Загалом база даних містить двадцять шість таблиць, розподілених між сімома функціональними модулями: керування користувачами, новини, форум, чати, спільноти, календар релізів та модерація контенту.

Модуль керування користувачами є ядром усієї системи, оскільки він забезпечує процеси реєстрації, автентифікації, розмежування прав доступу та підтримки безпечних сесій. Таблиця users є центральною сутністю платформи і містить ключові ідентифікаційні дані про кожного зареєстрованого учасника: унікальний логін, криптографічно хешований пароль, підтверджену електронну адресу, часову мітку реєстрації та поточну системну роль. Поля is_banned, is_muted та content_blocked забезпечують гнучке управління обмеженнями для порушників правил спільноти без їхнього фізичного видалення з бази. Поле avatar_url зберігає посилання на зображення профілю користувача, яке зберігається у хмарному сховищі. Детальний опис структурних полів таблиці наведено в таблиці 2.4.

Таблиця 2.4

Структура таблиці users

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
email		character varying	електронна адреса користувача

продовж. табл. 2.4

username		character varying	ім'я користувача
hashed_password		text	хешований пароль
role		USER-DEFINED	роль користувача
is_active		boolean	статус активності
avatar_url		character varying	посилання на аватар
is_banned		boolean	статус блокування
is_muted		boolean	статус обмеження
content_blocked		boolean	статус блокування контенту
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця refresh_tokens відповідає за зберігання токенів оновлення, що дозволяють безпечно регенерувати короткострокові сесії доступу за протоколом JWT та підтримувати безперервну роботу користувача з платформою без повторного введення пароля. Кожен токен пов'язаний з конкретним користувачем через зовнішній ключ до таблиці users та має поля expires_at і revoked_at, що дозволяють реалізувати механізм примусового завершення сесій та автоматичного інвалідування прострочених токенів. Структура таблиці refresh_tokens наведена у таблиці 2.5.

Таблиця 2.5

Структура таблиці refresh_tokens

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
user_id	Foreign Key	uuid	ідентифікатор користувача
token		text	токен авторизації
expires_at		timestamp with time zone	дата завершення дії
revoked_at		timestamp with time zone	дата відкликання токена

продовж. табл. 2.5

created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Функціонал агрегації та публікації новин реалізовано у вигляді окремого модуля, що забезпечує створення, модерацію, перегляд та взаємодію користувачів із інформаційними матеріалами ігрової індустрії. Центральною таблицею цього модуля є news_articles, яка зберігає повний набір метаданих кожного новинного матеріалу. Використання поля is_published дозволяє адміністраторам готувати матеріали заздалегідь та публікувати їх у потрібний момент. Структуру таблиці news_articles наведено у таблиці 2.6.

Таблиця 2.6

Структура таблиці news_articles

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
title		character varying	заголовок
description		text	короткий опис
content		text	основний вміст
category		character varying	Категорія
image_url		character varying	посилання на зображення
author		character varying	ім'я автора
tags		ARRAY	список тегів
likes		integer	кількість вподобань
comments_count		integer	кількість коментарів
views		integer	кількість переглядів
is_published		boolean	статус публікації
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця news_comments забезпечує зберігання коментарів користувачів до новинних публікацій. Кожен коментар пов'язаний із конкретною статтею через зовнішній ключ article_id та з автором через user_id. Поле user_name дублює ім'я користувача на момент написання коментаря, що дозволяє коректно відображати авторство навіть у разі подальшої зміни нікнейму. Лічильник likes фіксує кількість вподобань, отриманих коментарем від інших учасників спільноти. Детальний опис полів наведено у таблиці 2.7.

Таблиця 2.7

Структура таблиці news_comments

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
article_id	Foreign Key	uuid	ідентифікатор статті
user_id	Foreign Key	uuid	ідентифікатор користувача
user_name		character varying	ім'я користувача
text		text	текст повідомлення або коментаря
likes		integer	кількість вподобань
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Для реєстрації реакцій користувачів на новинні матеріали використовуються таблиці news_article_likes та news_article_views, структури яких представлено у таблицях 2.8 та 2.9 відповідно. Таблиця news_article_likes фіксує факт вподобання конкретної статті конкретним користувачем через пару зовнішніх ключів, що разом утворюють унікальне обмеження, яке виключає дублювання реакцій. Таблиця news_article_views натомість використовує поле viewer_key замість прямого посилання на користувача, що дозволяє відстежувати унікальні перегляди також від неавторизованих відвідувачів платформи.

Таблиця 2.8

Структура таблиці news_article_likes

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
user_id	Foreign Key	uuid	ідентифікатор користувача
article_id	Foreign Key	uuid	ідентифікатор статті
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця 2.9

Структура таблиці news_article_views

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
article_id	Foreign Key	uuid	ідентифікатор статті
viewer_key		character varying	ідентифікатор переглядача
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Форум є одним із ключових інструментів соціальної взаємодії всередині платформи FanPulse. Його архітектура складається з восьми взаємопов'язаних таблиць, що забезпечують повноцінну ієрархічну структуру обговорень. Таблиця forum_categories описує верхньорівневі тематичні розділи форуму. Кожна категорія має унікальний символічний ідентифікатор slug для формування SEO-сумісних URL-адрес, поля кольорового оформлення та іконки для візуальної диференціації розділів у інтерфейсі, а також поле sort_order для керування порядком відображення. Категорії є базовим рівнем структурування форуму та слугують основою для організації тематичних обговорень. Структуру таблиці forum_categories наведено у таблиці 2.10.

Структура таблиці forum_categories

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
name		character varying	назва
slug		character varying	символьний ідентифікатор
description		text	короткий опис
color		character varying	колір оформлення
icon		character varying	іконка категорії
sort_order		integer	порядок сортування
is_active		boolean	статус активності
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця forum_topics зберігає окремі теми обговорень, кожна з яких обов'язково прив'язана до певної категорії через зовнішній ключ category_id. Окрім основного текстового контенту, запис теми містить поля статусу is_pinned та is_deleted, що реалізують механізм закріплення важливих обговорень та м'якого видалення без безповоротної втрати даних. Денормалізовані лічильники views_count, replies_count та likes_count дозволяють ефективно відображати статистику без виконання дорогих агрегуючих запитів. Структуру таблиці forum_topics наведено у таблиці 2.11.

Структура таблиці forum_topics

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
title		character varying	Заголовок
content		text	основний вміст

продовж. табл. 2.11

category_id	Foreign Key	uuid	ідентифікатор категорії
author_id	Foreign Key	uuid	ідентифікатор автора
status		character varying	поточний статус
is_pinned		boolean	статус закріплення
is_deleted		boolean	статус видалення
views_count		integer	кількість переглядів
replies_count		integer	кількість відповідей
likes_count		integer	кількість вподобань
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця forum_posts зберігає окремі повідомлення у межах тем. Ключовою особливістю цієї таблиці є поле parent_id, яке посиляється на саму себе через зовнішній ключ, формуючи рекурсивну деревоподібну структуру відповідей. Такий підхід дозволяє реалізувати вкладені коментарі довільної глибини без ускладнення схеми бази даних. Поля is_deleted та is_edited забезпечують коректне відстеження стану кожного повідомлення, а media_urls – зберігання посилань на прикріплені мультимедійні файли. Структуру таблиці forum_posts представлено у таблиці 2.12.

Таблиця 2.12

Структура таблиці forum_posts

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
topic_id	Foreign Key	uuid	ідентифікатор теми форуму
author_id	Foreign Key	uuid	ідентифікатор автора
parent_id	Foreign Key	uuid	ідентифікатор батьківського запису
content		text	основний вміст

продовж. табл. 2.12

is_deleted		boolean	статус видалення
is_edited		boolean	статус редагування
likes_count		integer	кількість вподобань
media_urls		text	посилання на медіафайли
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Для підтримки механізму вподобань публікацій форуму використовується таблиця forum_likes, структуру якої наведено у таблиці 2.13. Вона реєструє зв'язок між конкретним дописом та користувачем, що поставив відмітку «подобається». Пара зовнішніх ключів post_id та user_id спільно утворює унікальне обмеження, яке технічно унеможливорює повторне вподобання одного допису одним користувачем на рівні бази даних, що гарантує цілісність статистики.

Таблиця 2.13

Структура таблиці forum_likes

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
post_id	Foreign Key	uuid	ідентифікатор допису
user_id	Foreign Key	uuid	ідентифікатор користувача
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Система тегів форуму реалізована через пару пов'язаних таблиць: forum_tags та forum_topic_tags. Таблиця forum_tags є довідниковою та зберігає перелік доступних тегів з назвою, символьним ідентифікатором та кольоровим маркуванням для відображення в інтерфейсі. Таблиця forum_topic_tags реалізує зв'язок багато-до-багатьох між темами та тегами, дозволяючи призначати довільну кількість тегів кожній темі форуму. Структури цих таблиць наведено у таблицях 2.14 та 2.15 відповідно.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				49
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.14

Структура таблиці forum_tags

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
name		character varying	назва
slug		character varying	символьний ідентифікатор
color		character varying	колір оформлення
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця 2.15

Структура таблиці forum_topic_tags

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
topic_id	Foreign Key	uuid	ідентифікатор теми форуму
tag_id	Foreign Key	uuid	ідентифікатор тегу
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця forum_subscriptions реалізує механізм підписок користувачів на теми форуму, що дозволяє надсилати сповіщення про нові відповіді у відстежуваних обговореннях. Структуру цієї таблиці наведено у таблиці 2.16. Таблиця forum_reports, структуру якої відображено у таблиці 2.17, забезпечує функціонування системи скарг, у межах якої зареєстровані користувачі можуть подавати скарги на дописи із зазначенням причини. Поля resolved_by та admin_note використовуються для фіксації результатів розгляду скарги та службових коментарів адміністратора.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.16

Структура таблиці forum_subscriptions

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
topic_id	Foreign Key	uuid	ідентифікатор теми форуму
user_id	Foreign Key	uuid	ідентифікатор користувача
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця 2.17

Структура таблиці forum_reports

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
post_id	Foreign Key	uuid	ідентифікатор допису
reporter_id	Foreign Key	uuid	ідентифікатор заявника
reason		character varying	причина дії
status		character varying	поточний статус
resolved_by	Foreign Key	uuid	ідентифікатор адміністратора
admin_note		text	примітка адміністратора
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Система чатів забезпечує приватне та групове спілкування між учасниками платформи в режимі реального часу. Вона побудована на трьох взаємопов'язаних таблицях: chats, chat_members та messages. Таблиця chats зберігає метадані кожного чату: назву, опис, аватар, тип (особистий або груповий через поле type з перерахуванням USER-DEFINED) та ідентифікатор творця. Структуру таблиці chats наведено у таблиці 2.18.

Структура таблиці chats

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
name		character varying	назва
description		text	короткий опис
avatar_url		character varying	посилання на аватар
type		USER-DEFINED	тип об'єкта
created_by	Foreign Key	uuid	ідентифікатор творця
is_active		boolean	статус активності
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця chat_members реалізує зв'язок багато-до-багатьох між чатами та їхніми учасниками. Поле role визначає рівень привілеїв кожного учасника в межах конкретного чату, що дозволяє розрізняти адміністраторів групових бесід та звичайних членів. Структуру таблиці chat_members відображено у таблиці 2.19. Таблиця messages, деталізована у таблиці 2.20, зберігає хронологічну послідовність повідомлень, прив'язуючи кожне з них до чату та відправника через зовнішні ключі. Поле is_deleted підтримує м'яке видалення повідомлень без втрати загальної хронології листування.

Таблиця 2.19

Структура таблиці chat_members

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
chat_id	Foreign Key	uuid	ідентифікатор чату
user_id	Foreign Key	uuid	ідентифікатор користувача
role		USER-DEFINED	роль користувача

продовж. табл. 2.19

created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця 2.20

Структура таблиці messages

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
chat_id	Foreign Key	uuid	ідентифікатор чату
sender_id	Foreign Key	uuid	ідентифікатор відправника
content		text	основний вміст
is_deleted		boolean	статус видалення
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Модуль спільнот забезпечує функціонал тематичних ігрових груп, об'єднаних за інтересами учасників. Центральною таблицею є communities, яка зберігає загальну інформацію про кожну спільноту: назву, опис, посилання на аватар та банер, ідентифікатор власника та статус активності. Відмінністю цього модуля від інших є використання цілочисельного типу integer для первинних ключів замість uuid, що є цілком прийнятним рішенням для таблиць зі стабільним і відносно невеликим обсягом записів. Структуру таблиці communities наведено у таблиці 2.21.

Таблиця 2.21

Структура таблиці communities

Поле	Ключ	Тип	Призначення
id	Primary Key	integer	ідентифікатор запису
name		character varying	назва
description		text	короткий опис

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

продовж. табл. 2.21

avatar_url		character varying	посилання на аватар
banner_url		character varying	посилання на банер
owner_id	Foreign Key	uuid	ідентифікатор власника
is_active		boolean	статус активності
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця community_members управляє членством учасників у спільнотах. Кожен запис містить ідентифікатори спільноти та користувача, роль учасника (власник, адміністратор, модератор або звичайний член) та дату приєднання. Структуру цієї таблиці наведено у таблиці 2.22. Таблиця community_posts, описана у таблиці 2.23, забезпечує зберігання публікацій у стрічці кожної спільноти. Кожна публікація прив'язана до автора через зовнішній ключ author_id, що посиляється на таблицю users, та до спільноти через community_id.

Таблиця 2.22

Структура таблиці community_members

Поле	Ключ	Тип	Призначення
id	Primary Key	integer	ідентифікатор запису
community_id	Foreign Key	integer	ідентифікатор спільноти
user_id	Foreign Key	uuid	ідентифікатор користувача
role		text	роль користувача
joined_at		timestamp with time zone	дата приєднання

Таблиця 2.23

Структура таблиці community_posts

Поле	Ключ	Тип	Призначення
id	Primary Key	integer	ідентифікатор запису

продовж. табл. 2.23

title		character varying	Заголовок
content		text	основний вміст
author_id	Foreign Key	uuid	ідентифікатор автора
community_id	Foreign Key	integer	ідентифікатор спільноти
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця community_comments зберігає коментарі до публікацій спільнот. Аналогічно до таблиці forum_posts, вона підтримує вкладену структуру відповідей через поле parent_id із самопосиланням, що дозволяє формувати багаторівневі дискусії під кожним дописом. Структуру таблиці community_comments представлено у таблиці 2.24. Для реєстрації вподобань публікацій спільнот використовується таблиця community_post_likes зі структурою, наведеною у таблиці 2.25.

Таблиця 2.24

Структура таблиці community_comments

Поле	Ключ	Тип	Призначення
id	Primary Key	integer	ідентифікатор запису
content		text	основний вміст
post_id	Foreign Key	integer	ідентифікатор допису
author_id	Foreign Key	uuid	ідентифікатор автора
parent_id	Foreign Key	integer	ідентифікатор батьківського запису
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця 2.25

Структура таблиці community_post_likes

Поле	Ключ	Тип	Призначення
id	Primary Key	integer	ідентифікатор запису
post_id	Foreign Key	integer	ідентифікатор допису
user_id	Foreign Key	uuid	ідентифікатор користувача
created_at		timestamp with time zone	дата створення запису

Таблиця community_bans забезпечує роботу системи блокувань усередині окремих спільнот, що є відмінним від глобального бану на рівні платформи механізмом. Кожен запис блокування містить ідентифікатори спільноти та заблокованого користувача, посилання на адміністратора, що виконав дію, текстове обґрунтування рішення та часову мітку блокування. Детальний опис структури таблиці community_bans наведено у таблиці 2.26.

Таблиця 2.26

Структура таблиці community_bans

Поле	Ключ	Тип	Призначення
id	Primary Key	integer	ідентифікатор запису
community_id	Foreign Key	integer	ідентифікатор спільноти
user_id	Foreign Key	uuid	ідентифікатор користувача
banned_by_id	Foreign Key	uuid	ідентифікатор користувача, який виконав блокування
reason		text	причина дії
banned_at		timestamp with time zone	дата блокування

Календар релізів є унікальною функціональною особливістю платформи FanPulse, що фокусує увагу користувачів на актуальних подіях ігрової індустрії. Таблиця game_releases зберігає розширені метадані про заплановані та вже випущені ігрові проекти: назву, опис, обкладинку, дату виходу, поточний статус (очікується, вийшов, перенесено тощо через поле-перерахування status) та масиви

жанрів і підтримуваних ігрових платформ. Використання типу ARRAY для полів genres та platforms дозволяє ефективно зберігати множинні значення без введення додаткових довідникових таблиць. Структуру таблиці game_releases наведено у таблиці 2.27.

Таблиця 2.27

Структура таблиці game_releases

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
title		character varying	заголовок
description		text	короткий опис
cover_image_url		character varying	посилання на обкладинку
release_date		date	дата релізу
status		USER-DEFINED	поточний статус
genres		ARRAY	список жанрів
platforms		ARRAY	список платформ
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Таблиця release_reminders реалізує механізм особистих нагадувань, завдяки якому зареєстровані користувачі можуть підписатися на конкретний реліз гри та отримати своєчасне сповіщення про дату його виходу. Зв'язок здійснюється через два зовнішні ключі: user_id посилається на таблицю users, а release_id – на таблицю game_releases. Разом вони утворюють унікальну пару, що виключає дублювання підписок. Структуру таблиці release_reminders подано у таблиці 2.28.

Таблиця 2.28

Структура таблиці release_reminders

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису

продовж. табл. 2.28

user_id	Foreign Key	uuid	ідентифікатор користувача
release_id	Foreign Key	uuid	ідентифікатор релізу
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

Модуль модерації відповідає за підтримання порядку та чистоти публічного контенту всередині геймерської спільноти. Таблиця banned_comment_words містить динамічний перелік заборонених слів, ненормативної лексики та виразів, що є неприйнятними у межах платформи. Механізм автоматичної перевірки тексту перед публікацією реалізується на рівні бізнес-логіки бекенду шляхом порівняння введеного тексту з вмістом цієї таблиці. Такий підхід дозволяє адміністраторам динамічно оновлювати список заборонених виразів без зміни програмного коду. Структуру таблиці banned_comment_words наведено у таблиці 2.29.

Таблиця 2.29

Структура таблиці banned_comment_words

Поле	Ключ	Тип	Призначення
id	Primary Key	uuid	ідентифікатор запису
word		character varying	заборонене слово
created_at		timestamp with time zone	дата створення запису
updated_at		timestamp with time zone	дата оновлення запису

У процесі проектування фізичної схеми бази даних активно використовувалися обмеження первинних та зовнішніх ключів для суворого забезпечення посилальної цілісності та підтримки коректних зв'язків між сутностями. Це дозволяє повністю уникати аномалій дублювання даних та гарантує узгодженість інформації при виконанні складних аналітичних запитів з операцією JOIN. Для оптимізації роботи бази даних на найбільш використовуваних полях, таких як дати релізів, ідентифікатори користувачів та логіни, було створено B-tree індекси. Це

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

дозволяє прискорити виконання операцій пошуку, фільтрації та сортування інформації. Усі операції модифікації даних виконуються в межах транзакцій, що забезпечує цілісність і узгодженість даних при конкурентному доступі. Це дозволяє уникати часткових оновлень та неузгодженого стану бази даних у разі помилок або одночасного виконання запитів великою кількістю користувачів. Механізми керування транзакціями PostgreSQL додатково забезпечують обробку конфліктів доступу до ресурсів.

Архітектура розробленої бази даних FanPulse заздалегідь передбачає можливість як вертикального масштабування за рахунок оптимізації конфігурації PostgreSQL, так і горизонтального масштабування шляхом реплікації та шардингу даних. Подальше розширення функціоналу платформи, наприклад інтеграція системи ігрових досягнень або підключення зовнішніх API-сервісів, може здійснюватися шляхом додавання нових таблиць без радикальної перебудови існуючої структури. Загалом спроектована база даних FanPulse повністю забезпечує реалізацію всіх основних функціональних вимог соціальної мережі для фанатів комп'ютерних ігор та створює надійну технічну основу для її подальшого розвитку й модернізації.

2.4 Організація взаємодії користувача із системою та безпекою

Проектування програмного інтерфейсу та комплексної системи безпеки вебплатформи FanPulse становить один із найбільш критичних та відповідальних етапів життєвого циклу розробки, оскільки саме на цьому рівні теоретичні концептуальні моделі й технічні вимоги трансформуються у практичні програмні механізми безпосередньої взаємодії користувача з інформаційним середовищем спільноти. Архітектура користувацького інтерфейсу розроблялася із чітким дотриманням сучасних інженерних принципів модульності, горизонтальної масштабованості, високої продуктивності та ергономічності. Основний вектор проектування був спрямований на створення інтуїтивно зрозумілого, інтерактивного середовища, яке забезпечує мінімальний час навігаційного переходу користувача між ключовими функціональними модулями платформи, такими як новинна стрічка, деревоподібні форумні гілки обговорень, інтерактивний календар

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

релізів відеоігор та персональні чати. Побудова такої архітектури базується на класичному життєвому циклі розробки програмного забезпечення, що охоплює сім послідовних стадій, кожна з яких визначає параметри надійності, безпеки та інтерфейсної взаємодії системи (рис. 2.2).



Рис. 2.2. Сім етапів розробки програмного забезпечення

Важливим інженерним завданням під час проєктування клієнт-серверної архітектури стало забезпечення високого рівня узгодженості між фронтенд- та бекенд-частинами, де для мінімізації затримок було впроваджено парадигму асинхронної обробки кожної дії користувача.

На початкових етапах життєвого циклу, що включають генерацію ідей та детальний аналіз системних вимог, було закладено фундамент для майбутньої структури інтерфейсу [17] та механізмів інформаційного захисту. Аналітичне моделювання дозволило чітко розмежувати права доступу та визначити дворівневу модель взаємодії із системою, яка поділяється на гостьовий режим та режим авторизованого користувача. На етапі аналізу вимог було встановлено, що інтерфейс для неавторизованих гостей повинен забезпечувати повний доступ до читання публічного контенту (новин, календаря релізів, форумів), але повністю блокувати будь-які транзакційні запити на модифікацію бази даних, такі як створення постів чи надсилання повідомлень. Своєчасно сформовані технічні специфікації дозволили спроектувати інтерфейс адміністратора як центральний вузол керування платформою, оснащений інтерактивним аналітичним дашбордом для моніторингу активності спільноти та інструментами миттєвого застосування дисциплінарних заходів, що є первинним контуром забезпечення безпеки всередині соціальної мережі.

Перехід до безпосереднього проєктування продукту та його подальшої розробки дозволив визначити оптимальний технологічний стек для реалізації запланованих інтерфейсних та безпекових рішень. З інженерно-технічної точки зору, серверна частина програмного комплексу побудована на базі високопродуктивного фреймворку FastAPI, який функціонує на основі асинхронного інтерфейсу серверів вебшлюзу, забезпечуючи паралельну обробку HTTP-запитів в умовах великої кількості одночасних підключень. Взаємодія з реляційною базою даних PostgreSQL реалізована за допомогою ORM SQLAlchemy, що забезпечує зручний рівень абстракції для роботи з даними. Система міграцій Alembic використовується для керування версіями схеми бази даних та дозволяє безпечно вносити зміни до структури таблиць без порушення роботи застосунку. Це забезпечує контрольоване оновлення структури даних на різних етапах розробки [30].

Клієнтська частина реалізована на базі фреймворку Next.js, що гарантує високу швидкість завантаження сторінок за рахунок механізмів серверного рендерингу, покращує користувацький досвід та забезпечує ефективну пошукову оптимізацію. На етапі розробки підтримувалася інженерна культура коду, яка базувалася на ізоляції залежностей через віртуальні середовища `venv`, використанні системи Git [9] для контролю версій та автоматичній перевірці якості коду за допомогою лінтингу у середовищі Visual Studio Code. Це дозволяло зменшити кількість помилок у коді та підвищити його читабельність і підтримуваність.

Для забезпечення постійного зберігання мультимедійних файлів у системі використовується хмарний сервіс Cloudinary, який відповідає за завантаження, зберігання та доставку зображень. Такий підхід усуває залежність від локального файлового сховища сервера та гарантує збереження користувацьких файлів після повторного розгортання застосунку.

Важливим кроком у межах життєвого циклу розробки став етап інтеграції та комплексного тестування, під час якого особлива увага приділялася верифікації розроблених інтерфейсів API та підсистем безпеки [5]. Інженерні тести були спрямовані на перевірку захищеності системи від поширених мережових вразливостей та здійснення суворої валідації всіх вхідних даних на рівні Pydantic-

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				61
Змн.	Арк.	№ докум.	Підпис	Дата		

моделей фреймворку FastAPI, що запобігає спробам SQL-ін'єкцій чи міжсайтового скриптингу [19]. У межах тестування безпекових механізмів було підтверджено працездатність алгоритмів обов'язкового криптографічного хешування паролів користувачів перед їхнім збереженням у базу даних, що унеможливило викрадення автентифікаційних даних у відкритому вигляді. Крім того, на цьому етапі було всебічно протестовано підсистему автоматичної цензури, яка в асинхронному режимі сканує текстовий вміст повідомлень і коментарів на наявність забороненої лексики ще до моменту виконання операції запису у сховище даних, забезпечуючи підтримку безпечної та екологічної атмосфери в геймерській спільноті.

Заключні етапи розгортання та подальшого технічного обслуговування системи дозволили перенести розроблену інфраструктуру у хмарне середовище з дотриманням принципу розподілу функціональних обов'язків між окремими сервісами. Клієнтська частина вебплатформи, реалізована засобами Next.js, розгортається у хмарному середовищі, що забезпечує швидке відображення користувацького інтерфейсу та підтримку сучасних механізмів оптимізації вебзастосунків. Серверна частина, створена на основі FastAPI, функціонує разом із базою даних PostgreSQL у хмарній інфраструктурі Render, що дозволяє автоматизувати процеси розгортання та оновлення програмного забезпечення.

Для зберігання мультимедійного контенту в FanPulse інтегровано хмарний сервіс Cloudinary. Cloudinary дає змогу зберігати зображення незалежно від локального файлового сховища сервера, автоматично оптимізувати їх розмір та формат, а також забезпечувати швидку доставку контенту користувачам через глобальну мережу CDN. Такий підхід підвищує масштабованість системи, зменшує навантаження на серверну частину та усуває ризик втрати файлів під час оновлення або повторного розгортання застосунку.

Процес розгортання підтримує механізми безперервної інтеграції та доставки (CI/CD), завдяки яким кожна зміна програмного коду автоматично проходить етапи збірки, перевірки та публікації нової версії системи. На стадії супроводу здійснюється постійний моніторинг працездатності вебплатформи, аналіз

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				62
Змн.	Арк.	№ докум.	Підпис	Дата		

журналів подій та оцінка користувацької активності, що дозволяє оперативно виявляти потенційні помилки, підвищувати рівень безпеки та вдосконалювати функціональні можливості вебплатформи FanPulse.

Висновки до другого розділу

У другому розділі було визначено вимоги до соціальної мережі FanPulse та спроектовано основні компоненти системи. На основі поставлених завдань сформовано функціонал для гостей, зареєстрованих користувачів і адміністраторів, а також описано основні сценарії роботи із застосунком. Це дозволило чітко визначити можливості кожної категорії користувачів та межі їхнього доступу до функцій платформи.

У процесі проєктування розроблено структуру системи, яка включає модулі новин, календаря релізів, форумів, спільнот, чатів та адміністративної панелі. Для кращого розуміння взаємодії користувачів із платформою побудовано діаграму варіантів використання та наведено опис основних прецедентів. Отримані результати дозволяють оцінити логіку роботи системи ще до етапу програмної реалізації.

Також було спроектовано базу даних, яка забезпечує зберігання інформації про користувачів, публікації, коментарі, повідомлення, спільноти та інші об'єкти системи. Визначено основні зв'язки між сутностями, що дозволяє підтримувати цілісність даних і забезпечує коректну роботу всіх модулів платформи. Крім цього, структура бази даних передбачає можливість подальшого розширення функціоналу без суттєвих змін у її архітектурі.

Під час проєктування враховано питання безпеки та розмежування доступу між різними категоріями користувачів: передбачено механізми авторизації, автентифікації, модерації контенту та адміністрування системи, а також захист даних користувачів і контроль дій у межах платформи; додатково визначено вимоги до користувацького інтерфейсу, що забезпечують зручну навігацію, швидкий доступ до основних функцій системи та зрозумілу взаємодію з сервісом.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				63
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. РЕАЛІЗАЦІЯ СОЦІАЛЬНОЇ МЕРЕЖІ ДЛЯ ФАНАТІВ КОМП'ЮТЕРНИХ ІГОР

3.1 Реалізація функціоналу для неавторизованих користувачів

Неавторизований користувач, або гість, це будь-який відвідувач сайту FanPulse, який ще не пройшов реєстрацію або не увійшов у свій акаунт. Для такого користувача доступна головна сторінка платформи, розділ новин, форум, календар релізів ігор та сторінка реєстрації і входу. Всі ці сторінки реалізовані так, щоб гість міг ознайомитися з контентом платформи, але не мав можливості залишати коментарі, ставити лайки, писати на форумі чи у чатах.

Головна сторінка сайту (рис. 3.1) реалізована у файлі `page.tsx` в кореневій директорії `app` фронтенду. Для FanPulse головна сторінка одночасно є розділом новин, тобто вона повністю реалізована через компонент `NewsPage`. Це означає, що першим, що бачить гість при відкритті сайту, є стрічка новинних статей про ігрову індустрію. Такий підхід обраний свідомо, оскільки новини найчастіше оновлюються і є основним контентом, що приваблює нових відвідувачів.

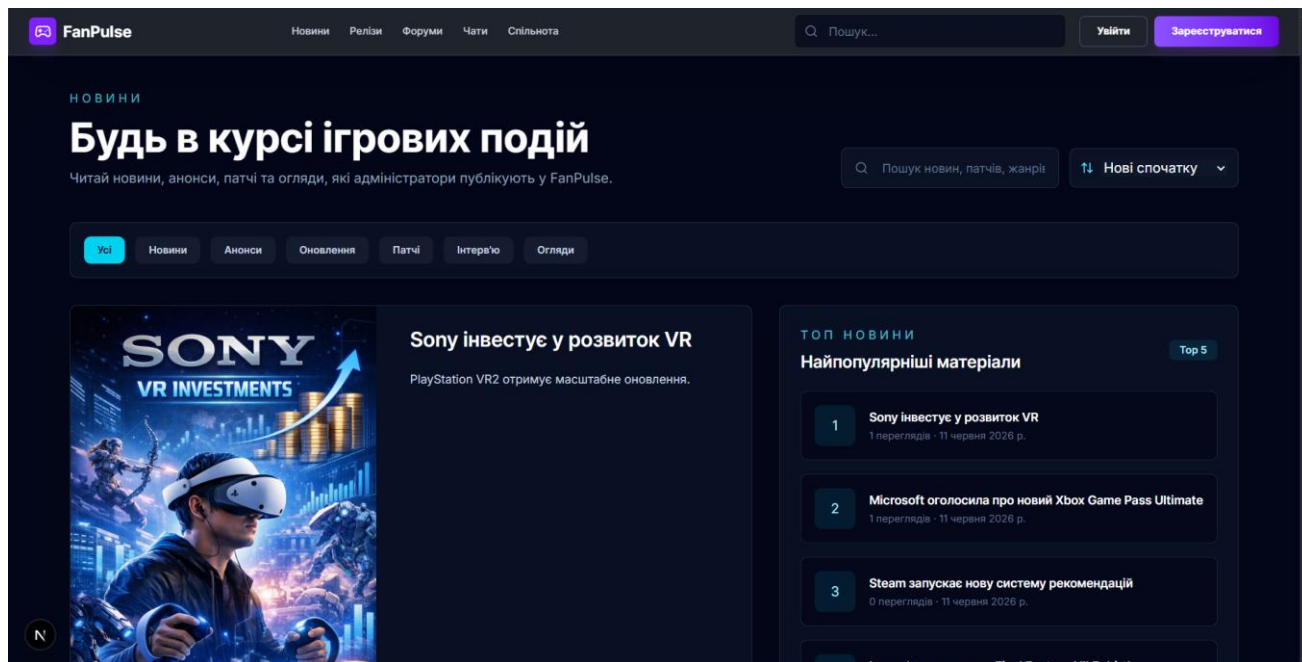


Рис. 3.1. Головна сторінка сайту зі стрічкою новин

На головній сторінці новини виводяться у вигляді карток. Кожна картка містить зображення обкладинки, заголовок, короткий опис, категорію, теги та лічильники переглядів і вподобань. Над списком карток розташована панель з

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				64
Змн.	Арк.	№ докум.	Підпис	Дата		

пошуком та фільтрами. Пошук дозволяє знайти статтю за словом у заголовку, описі або тегах. Поруч знаходиться список категорій, наприклад Новини, Анонси, Оновлення, Патчі, Інтерв'ю та Огляди, при виборі яких список карток автоматично звужується. Також доступне сортування за датою публікації, популярністю, кількістю переглядів або кількістю коментарів.

Дані для головної сторінки отримуються з бекенду через запит до ендпоінту новин, реалізованого у роутері news.py, за допомогою сервісу newsService. Гість може натиснути на будь-яку картку і перейти на детальну сторінку статті (рис. 3.2) за адресою /news/[id], де відображається повний текст новини, зображення та список коментарів інших користувачів. Сам гість не може написати новий коментар чи поставити лайк, але бачить уже опубліковані реакції інших учасників спільноти.

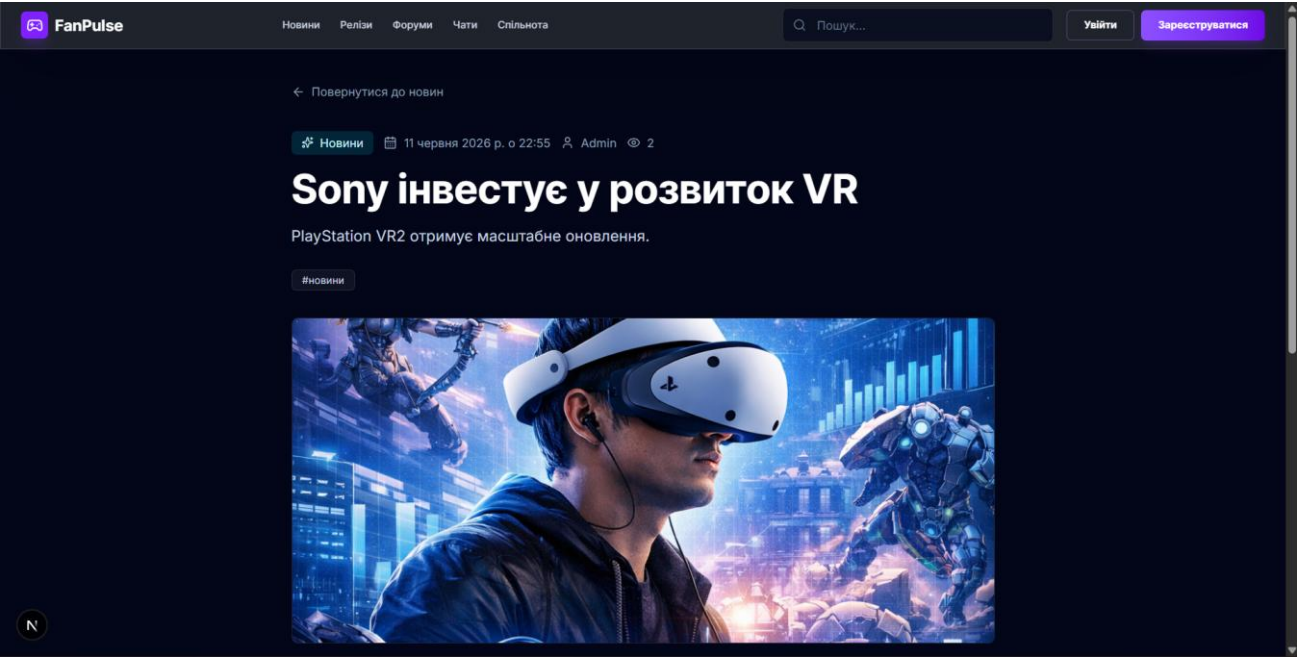


Рис. 3.2. Детальна сторінка новинної статті

При відкритті конкретної статті система автоматично фіксує перегляд через таблицю news_article_views. Якщо користувач не авторизований, його перегляд записується через спеціальний ідентифікатор viewer_key, який генерується на основі браузера відвідувача. Це дозволяє рахувати унікальні перегляди навіть для гостей, не вимагаючи від них реєстрації.

Форум (рис. 3.3) знаходиться за адресою /forum і реалізований через компоненти ForumCategories та ForumTopics. На головній сторінці форуму гість бачить список категорій обговорень, наприклад загальні теми, новини ігрової індустрії або технічна підтримка. Натиснувши на категорію, користувач переходить на список тем усередині цієї категорії. Кожна тема показує заголовок, автора, кількість відповідей, кількість переглядів і дату останньої активності. Відкривши конкретну тему за адресою /forum/topics/[id], гість може прочитати всі повідомлення у цій темі, включно з вкладеними відповідями, але кнопки додавання нового повідомлення для нього недоступні або при спробі написати з'являється повідомлення про потребу авторизації.

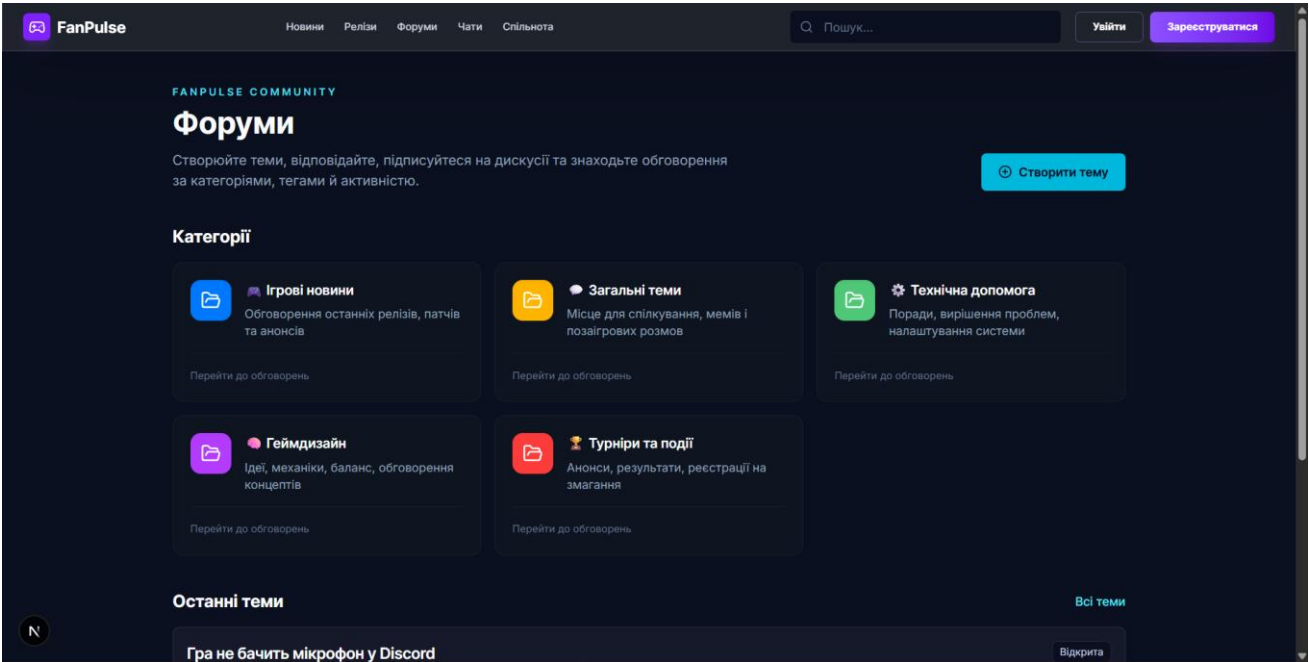


Рис. 3.3. Сторінка форуму зі списком категорій та тем

Календар релізів (рис. 3.4) доступний за адресою /releases. Ця сторінка показує список ігор з датами виходу, платформами та короткими описами. Реалізовано фільтрацію за статусом релізу: майбутні релізи, ігри що вже вийшли та скасовані. Окремо є фільтр за платформою, наприклад PC, PlayStation 5, Xbox або Nintendo Switch. Список платформ для фільтра формується динамічно на основі даних з бази, тобто якщо в базі з'явиться нова платформа, вона автоматично з'явиться у фільтрі без зміни коду. Гість може переглядати всі ці дані, але кнопка

з дзвіночком для підписки на нагадування про реліз доступна тільки для авторизованих користувачів.

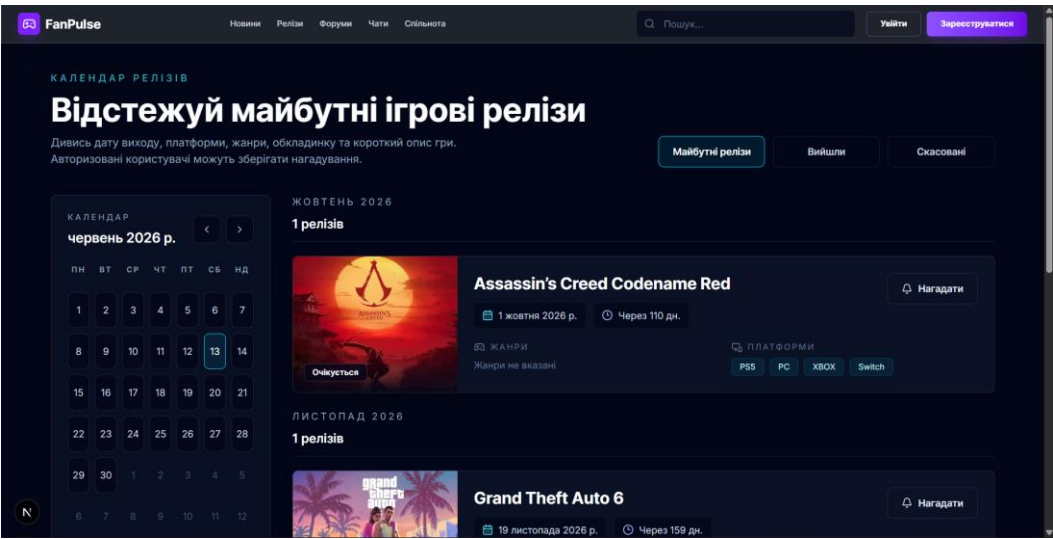


Рис. 3.4. Сторінка календаря релізів з фільтрами

Сторінки реєстрації (рис. 3.5) та входу знаходяться у групі маршрутів (auth) і реалізовані компонентами RegisterForm та LoginForm. Форма реєстрації містить поля для імені користувача, електронної пошти та пароля. Перед відправкою даних на сервер форма перевіряє, чи заповнені всі поля та чи відповідає пароль вимогам по довжині. Після натискання кнопки реєстрації запит надсилається на бекенд через сервіс authService, який звертається до ендпоінту /auth/register.

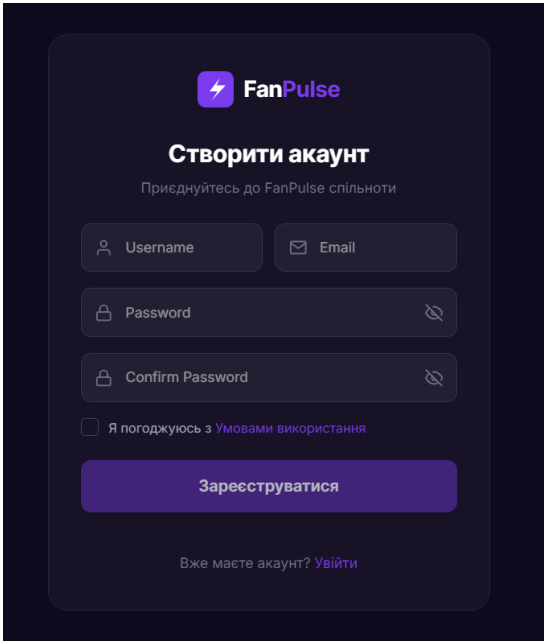


Рис. 3.5. Форма реєстрації нового користувача

Форма входу (рис. 3.6) працює аналогічно, але приймає тільки email та пароль і звертається до ендпоінту `/auth/login`. Якщо дані введені некоректно, наприклад email не знайдений у базі або пароль не співпадає з хешем, сервер повертає помилку, і на формі з'являється відповідне повідомлення червоного кольору під полями вводу. Після успішного входу користувач автоматично перенаправляється на головну сторінку, де замість гостьового дашборду тепер відображається персональна стрічка контенту.

Рис. 3.6. Форма входу в систему

Важливою частиною роботи з гостями є обмеження доступу на рівні бекенду, а не тільки на рівні інтерфейсу. Навіть якщо гість якимось чином спробує надіслати запит на створення повідомлення безпосередньо, обходячи інтерфейс, сервер все одно відмовить у виконанні цієї дії. Це досягається через механізм залежностей FastAPI, який перевіряє наявність та валідність токена авторизації перед тим, як виконати захищену операцію. Якщо токен відсутній або недійсний, ендпоінт повертає помилку зі статусом 401, що означає відсутність авторизації. Таким чином, реалізація для гостей побудована за принципом, коли інтерфейс

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

підказує користувачу що йому недоступно, а бекенд гарантує, що ці обмеження неможливо обійти.

Підсумовуючи, функціонал для неавторизованих користувачів реалізований так, щоб максимально показати можливості платформи і заохотити людину зареєструватися. Гість може читати новини, переглядати форум, дивитися календар релізів ігор та ознайомитися з загальним виглядом спільнот, але будь-яка дія, що змінює дані в базі, наприклад лайк, коментар чи створення теми, вимагає від нього спочатку пройти реєстрацію або вхід у систему.

Після успішного входу в систему користувач отримує доступ до повного функціоналу платформи FanPulse. Авторизований стан зберігається через два токени: access token, який живе недовго і використовується для кожного запиту, та refresh token, який зберігається довше і дозволяє автоматично оновити access token без повторного введення пароля. Ці токени зберігаються у глобальному сховищі стану auth.store, реалізованому через бібліотеку Zustand, і автоматично додаються до заголовків кожного запиту у файлі axios.ts.

Перша можливість, яку отримує користувач, це керування власним профілем. Сторінка профілю (рис. 3.7) знаходиться у захищеній групі маршрутів (protected)/profile і доступна тільки авторизованим користувачам. Користувач може натиснути кнопку редагування і змінити своє ім'я або завантажити нове зображення профілю. Завантаження аватара реалізоване через ендпоінт /user/me/avatar, на бекенді файл зберігається у папці uploads/avatars.

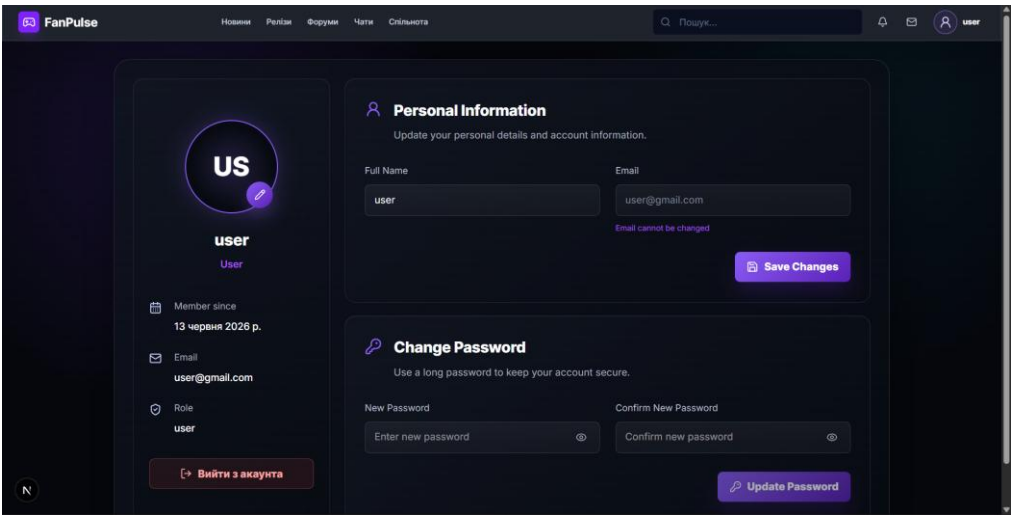


Рис. 3.7. Сторінка профілю авторизованого користувача

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

Окрім самого профілю, авторизованому користувачу доступний особистий дашборд рис. 3.8, на якому зведено інформацію про активність користувача: кількість створених тем форуму, дописів у спільнотах та підписок на нагадування про релізи ігор.

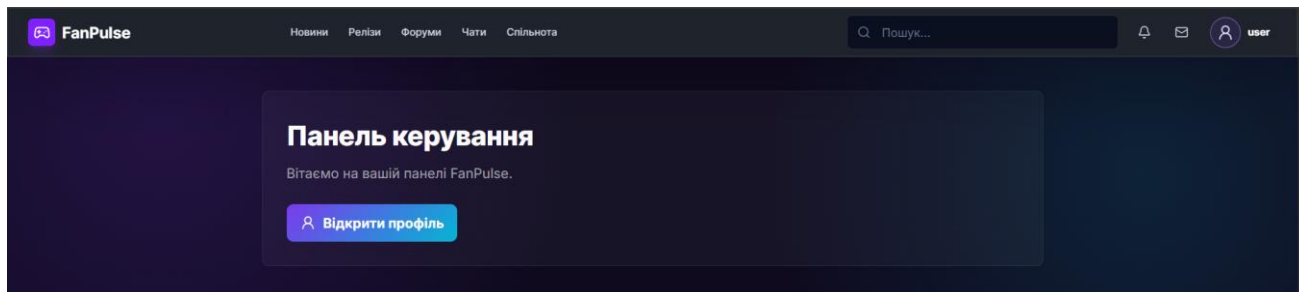


Рис. 3.8. Особистий дашборд авторизованого користувача

Окремий блок на сторінці профілю дозволяє змінити пароль. Для цього потрібно ввести старий пароль та новий пароль двічі для підтвердження. Запит надсилається на бекенд, де перевіряється правильність старого пароля через порівняння хешів за допомогою бібліотеки bcrypt, і тільки після успішної перевірки новий пароль хешується і зберігається в базі даних. Якщо старий пароль введено неправильно, форма показує відповідну помилку без зміни даних у базі.

Авторизований користувач потрапляє на ту саму головну сторінку зі стрічкою новин, але отримує повний доступ до взаємодії з контентом. На сторінці конкретної статті за адресою /news/[id] з'являється форма для додавання коментаря (рис. 3.9), а під самою статтею і під кожним коментарем доступні кнопки лайка. Натискання на кнопку лайка надсилає запит до бекенду, який перевіряє чи користувач вже ставив лайк цій статті раніше. Якщо ні, створюється новий запис у таблиці news_article_likes і лічильник likes збільшується на одиницю. Якщо лайк вже стояв, запис видаляється і лічильник зменшується. Це реалізовано так, щоб один користувач не міг проголосувати за одну статтю кілька разів.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

Valve оголосила про запуск нової системи рекомендацій у Steam, яка використовує машинне навчання для аналізу поведінки гравців. Алгоритм враховує не лише історію покупок, але й час, проведений у грі, відгуки друзів та популярність серед спільноти. Це дозволить користувачам швидше знаходити ігри, які відповідають їхнім інтересам. Valve підкреслює, що нова система також допоможе незалежним розробникам отримати більше уваги до своїх проєктів.

♡ 0 💬 0 коментарів

Коментарі (0)





Надіслати

Коментарів ще немає. Будьте першим у цій розмові.

Рис. 3.9. Форма додавання коментаря до новинної статті

Форум для авторизованих користувачів стає повністю інтерактивним. З'являється кнопка створення нової теми (рис. 3.10), яка веде на сторінку /forum/create. Форма створення теми реалізована через компонент ForumTopicForm і дозволяє вказати заголовок, текст, категорію та теги обговорення. Перед збереженням теми бекенд перевіряє текст на наявність слів зі словника заборонених виразів у таблиці banned_comment_words. Якщо хоча б одне заборонене слово знайдено, сервер повертає помилку і тема не зберігається в базі.

FanPulse Новини Релізи Форуми Чати Спільнота 🔍 Пошук... 🔔 ✉ 👤 user

< Назад до форумів

Категорія

- 🎮 Ігрові новини
- 🗨️ **Загальні теми**
- ⚙️ Технічна допомога
- 🎨 Геймдизайн
- 🏆 Турніри та події

Створити нову тему

Категорія
🗨️ Загальні теми

Заголовок
Які ігри ви зараз проходите?

Опис
«Привіт, друзі! 🎮
Ця тема створена для вільного спілкування про те, у що ви граєте прямо зараз. Поділіться своїми враженнями, улюбленими моментами та порадьте щось новеньке іншим учасникам.
Можна писати як про великі релізи, так і про інді-ігри чи класику. Головне — живе обговорення та дружня атмосфера!»

Теги
🔖 Ідея 🔖 **Новинка** 🔖 Гайд 🔖 Баг 🔖 Мод

Створити тему

Рис. 3.10. Форма створення нової теми на форумі

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

Усередині теми форуму авторизований користувач може публікувати повідомлення через компонент ForumPostForm. Кожне повідомлення може бути як окремим коментарем до теми, так і відповіддю на повідомлення іншого користувача. У разі створення відповіді в базі даних формується запис із полем `parent_id`, яке містить посилання на батьківське повідомлення. Такий підхід дозволяє відображати вкладені відповіді з відповідними відступами та формувати деревоподібну структуру обговорення.

Користувачі також мають можливість взаємодіяти з контентом форуму, ставлячи вподобання до повідомлень інших учасників та підписуючись на цікаві теми для подальшого відстеження активності (рис. 3.11). Це сприяє підвищенню залученості користувачів та створенню активної спільноти навколо обговорення ігрової тематики.

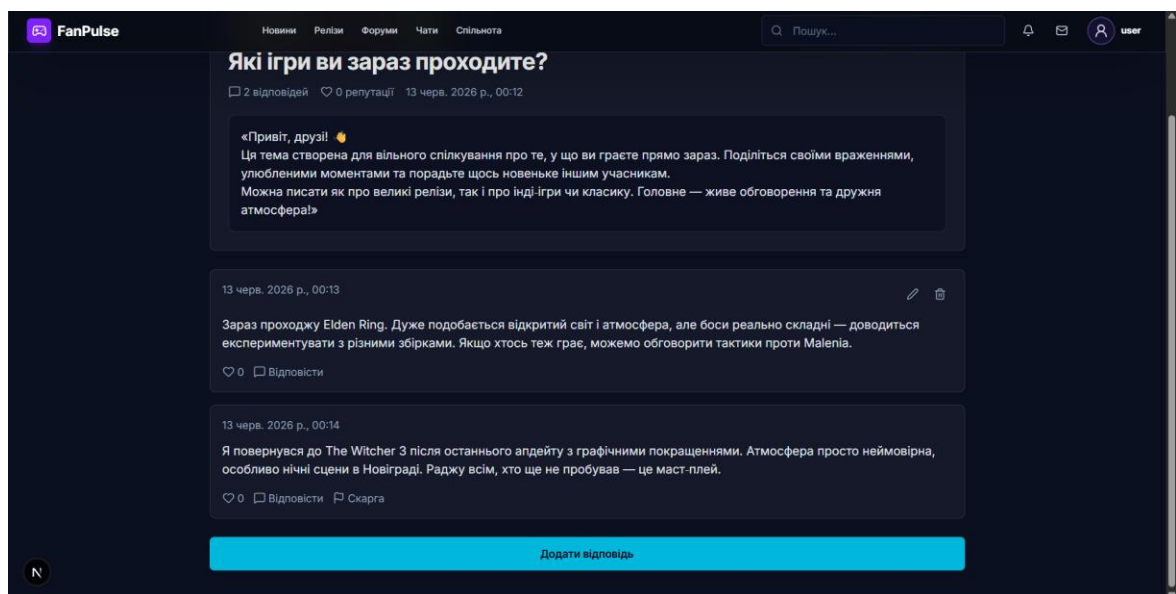


Рис. 3.11. Тема форуму з вкладеними відповідями користувачів

Для підтримки комфортної атмосфери спілкування на платформі реалізовано автоматичну систему модерації текстового контенту. Перед збереженням повідомлення виконується перевірка його вмісту на наявність забороненої лексики. У разі виявлення небажаних слів система блокує публікацію повідомлення та інформує користувача про необхідність змінити його зміст. Такий механізм дозволяє зменшити кількість неприйняттого контенту та підтримувати дотримання правил спільноти.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				72
Змн.	Арк.	№ докум.	Підпис	Дата		

Модуль спільнот (рис. 3.12) дозволяє авторизованому користувачу створити власну групу за інтересами через сторінку /communities/new. Користувач вказує назву, опис, а також може завантажити аватар та банер спільноти. Після створення користувач автоматично стає власником спільноти з найвищим рівнем прав. На сторінці спільноти /communities/[id] відображається стрічка постів, список учасників та кнопка приєднання для тих, хто ще не є членом спільноти.

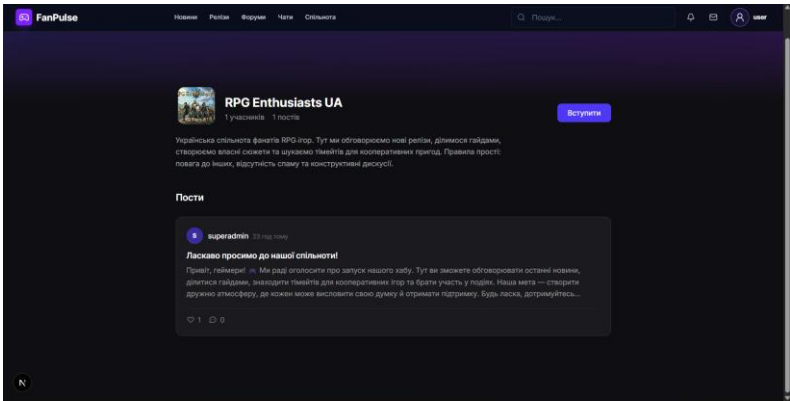


Рис. 3.12. Сторінка спільноти зі стрічкою постів

Створення нової спільноти доступне авторизованому користувачу за адресою /communities/new (рис. 3.13). Форма дозволяє вказати назву, опис та завантажити аватар спільноти через ендпоінт /communities/upload-avatar. Після успішного створення поточний користувач автоматично призначається власником спільноти у таблиці community_members.

Рис. 3.13. Форма створення нової спільноти

При натисканні на пост у стрічці спільноти користувач переходить на сторінку /posts/[id] (рис. 3.14), де відображається повний текст поста, лічильник лайків та список коментарів. Авторизований користувач може залишити власний коментар або відповісти на коментар іншого учасника, що створює вкладену структуру через поле parent_id у таблиці community_comments.

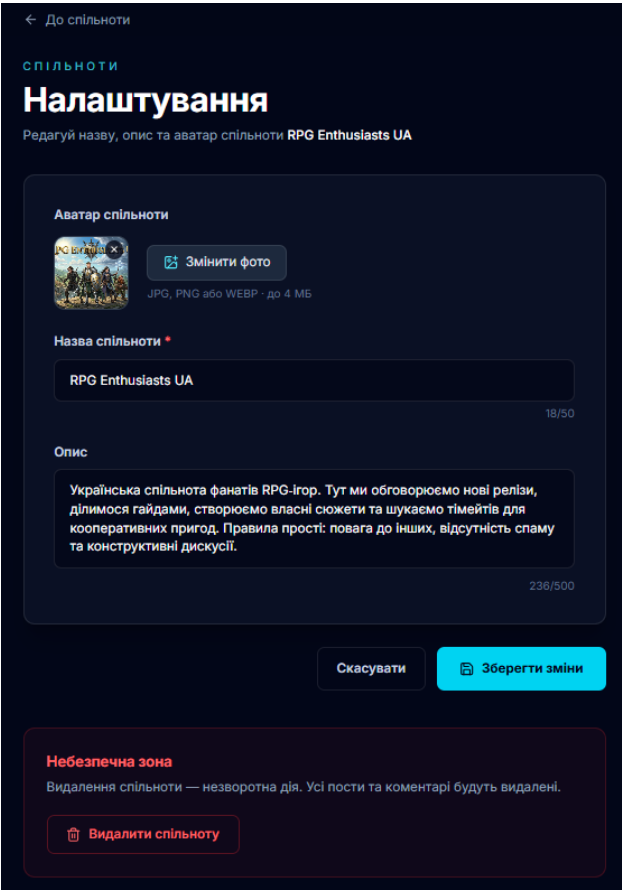


Рис. 3.14. Сторінка поста спільноти з коментарями

Якщо користувач є власником або адміністратором спільноти, для нього на сторінці спільноти з'являється посилання на сторінку налаштувань /communities/[id]/settings. Там можна змінити назву, опис, зображення спільноти, а також керувати учасниками: підвищувати їх до модераторів, призначати банк або видаляти учасників зі спільноти. Кожна дія перевіряється на бекенді через сервіс community_service, який спочатку дивиться роль поточного користувача в таблиці community_members і тільки потім дозволяє виконати дію.

Система чатів (рис. 3.15) реалізована на сторінці /chats у захищеній групі маршрутів. Користувач бачить список своїх чатів зліва, а праворуч відкривається вікно з історією повідомлень обраного чату. Для надсилання повідомлень

використовується форма з валідацією через бібліотеку react-hook-form та схему zod, яка не дозволяє надіслати порожнє повідомлення або повідомлення довжиною більше чотирьох тисяч символів. Повідомлення зберігаються в таблиці messages і прив'язані до конкретного чату через chat_id та до автора через sender_id.

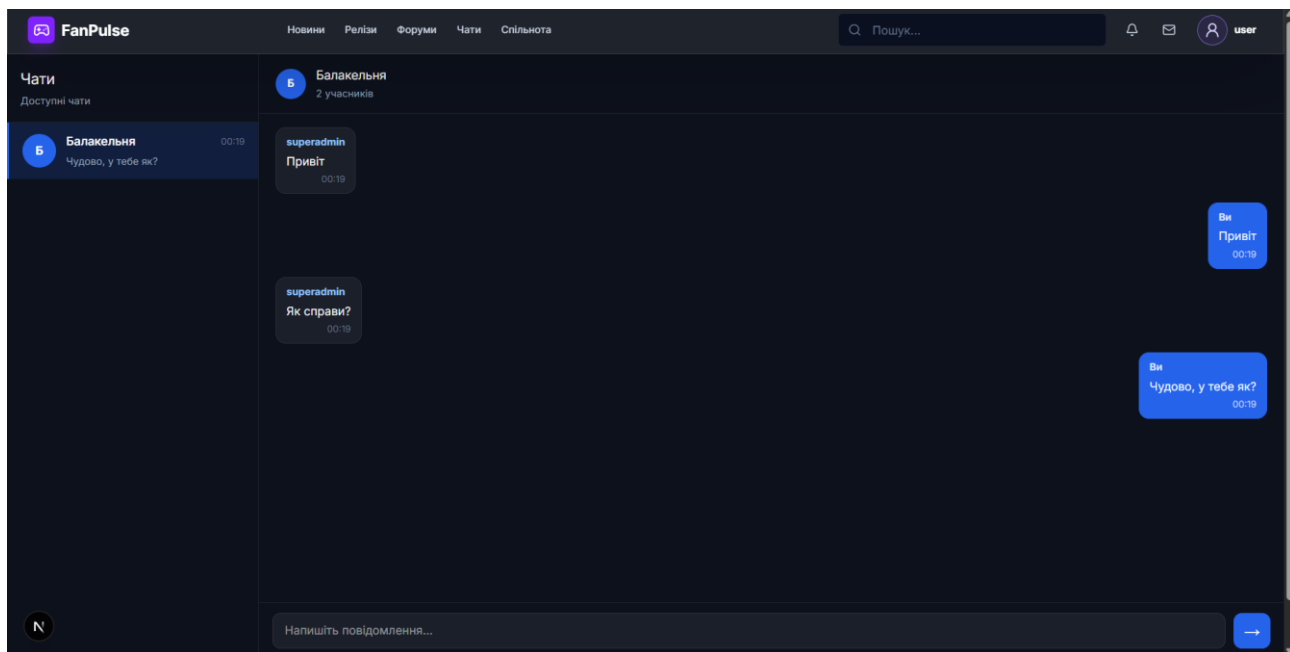


Рис. 3.15. Сторінка чатів з історією повідомлень

В календарі релізів авторизований користувач отримує додаткову можливість підписатися на нагадування про конкретну гру через кнопку з іконкою дзвіночка. При натисканні створюється запис у таблиці release_reminders, що пов'язує ідентифікатор користувача з ідентифікатором релізу. Якщо нагадування вже встановлено, повторне натискання видаляє запис, тобто кнопка працює як перемикач.

Усі описані можливості доступні користувачу з базовою роллю «user». Якщо адміністратор присвоїв йому роль модератора певної спільноти або форуму, додатково з'являються кнопки керування контентом у відповідних межах, наприклад видалення чужих повідомлень у своїй спільноті. Загалом реалізація для авторизованих користувачів покриває всі основні соціальні функції платформи: спілкування на форумі, участь у спільнотах, обмін повідомленнями, реакції на контент та персоналізацію власного профілю.

3.2 Реалізація адміністративної частини системи

Адміністративна панель (рис. 3.16) FanPulse реалізована як окремий розділ /admin і доступна тільки користувачам з роллю admin або superadmin. Доступ перевіряється як на фронтенді, так і на бекенді. На фронтенді при спробі відкрити будь-яку сторінку з префіксом /admin система перевіряє роль поточного користувача зі стану auth.store, і якщо роль не відповідає, користувача перенаправляють на головну сторінку. На бекенді кожен ендпоінт адмінпанелі захищений залежністю, яка додатково перевіряє роль із токена, тому навіть пряме звернення до API без потрібної ролі поверне помилку доступу.

Загальна структура адмінпанелі складається з бокового меню та основної робочої області. Бокове меню реалізоване в компоненті Sidebar.tsx і містить пункти: Дашборд, Новини, Календар релізів, Користувачі, Чати, Коментарі, Пости форуму та Скарги. Кожен пункт меню веде на свою окрему сторінку зі списком записів та можливістю редагування.

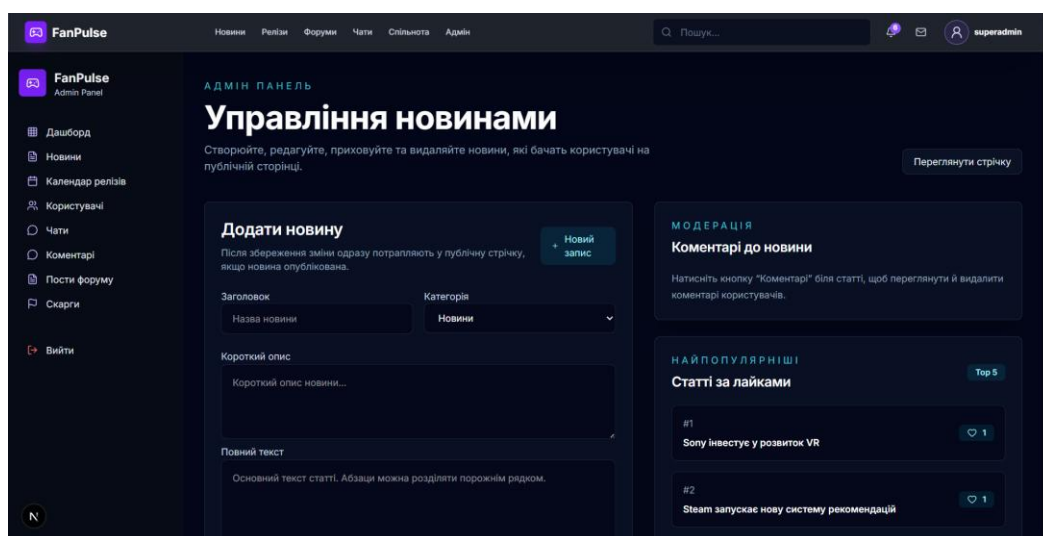


Рис. 3.16. Загальний вигляд адміністративної панелі з бічним меню

Головна сторінка адмінпанелі /admin реалізована через компонент Dashboard і показує загальну статистику (рис. 3.17) платформи у вигляді карток StatCard. Картки відображають кількість зареєстрованих користувачів, кількість опублікованих новин, активних тем форуму та інші ключові показники. Дані для дашборду отримуються одним запитом до бекенду, який рахує кількість записів у відповідних таблицях бази даних на момент завантаження сторінки.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				76
Змн.	Арк.	№ докум.	Підпис	Дата		

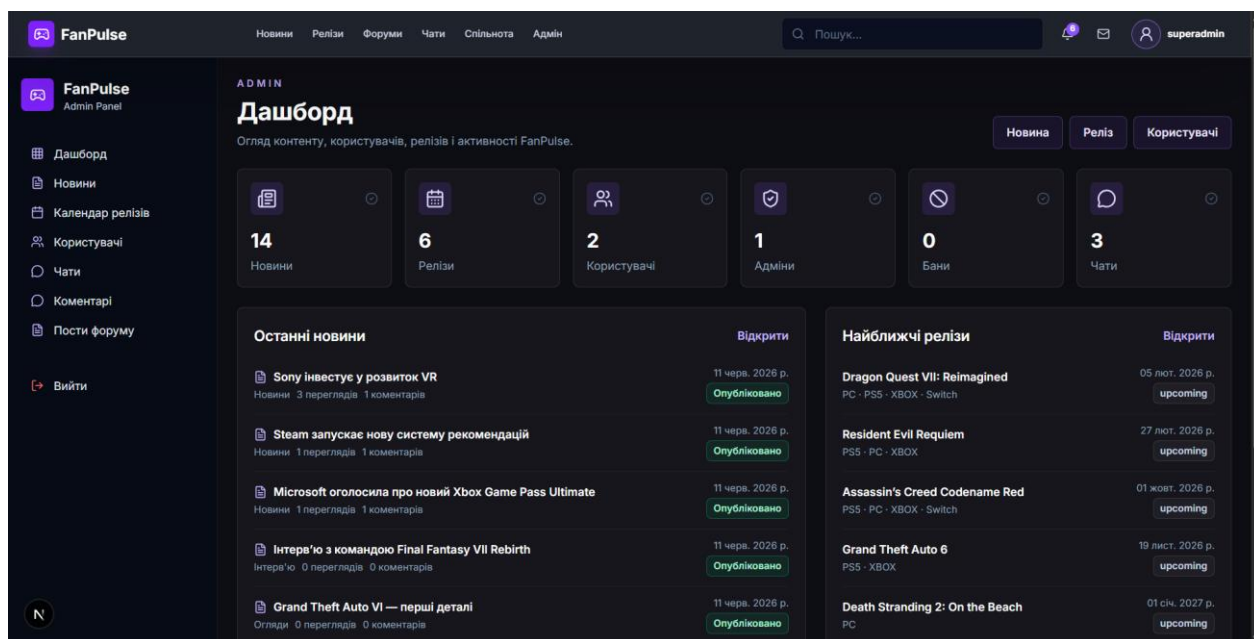


Рис. 3.17. Аналітичний дашборд адміністратора

Розділ управління новинами (рис. 3.18) знаходиться за адресою /admin/news і реалізований через компонент AdminNewsPage. Адміністратор бачить таблицю статей із функціями редагування, видалення та додавання нових матеріалів через спеціальну форму. Перемикач is_published дозволяє зберігати статтю як чернетку або миттєво публікувати її без повторного заповнення полів.

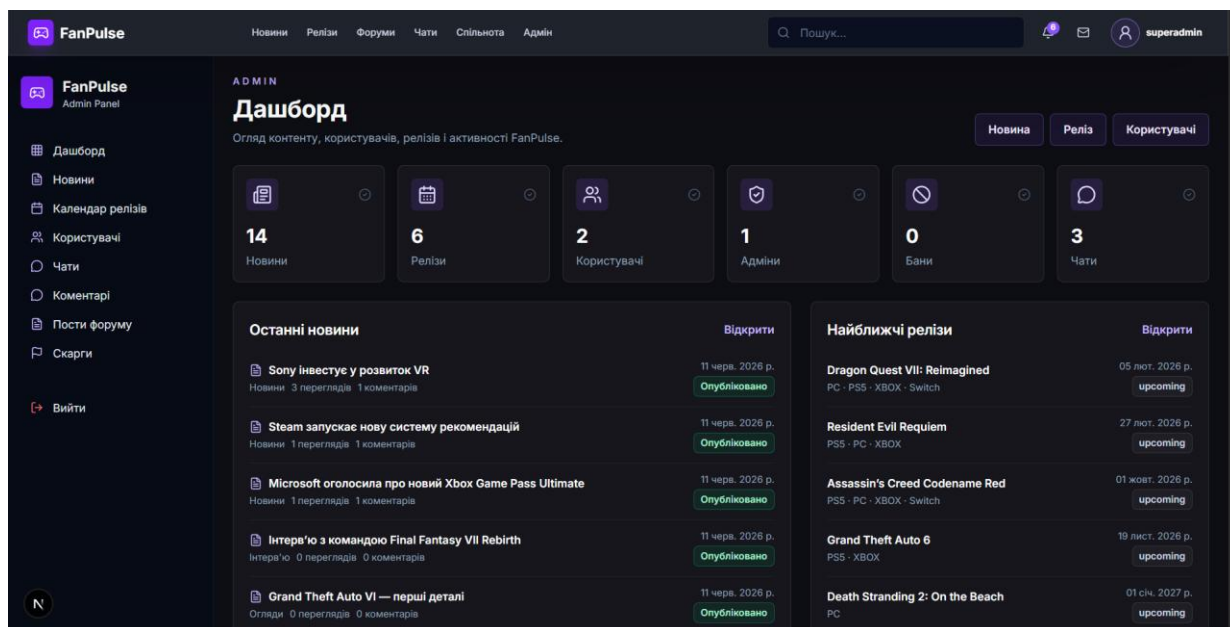


Рис. 3.18. Сторінка управління новинами в адмінпанелі

Розділ календаря релізів (рис. 3.19) знаходиться за адресою /admin/releases-calendar і реалізований через компоненти ReleasesCalendar та ReleaseForm. Адміністратор може додавати нові ігри або редагувати наявні, заповнюючи дані про

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				
Змн.	Арк.	№ докум.	Підпис	Дата		77

назву, дату, обкладинку, платформи та опис. Зміна статусу релізу в адмінпанелі автоматично оновлює його відображення у відповідних вкладках фільтра для користувачів.

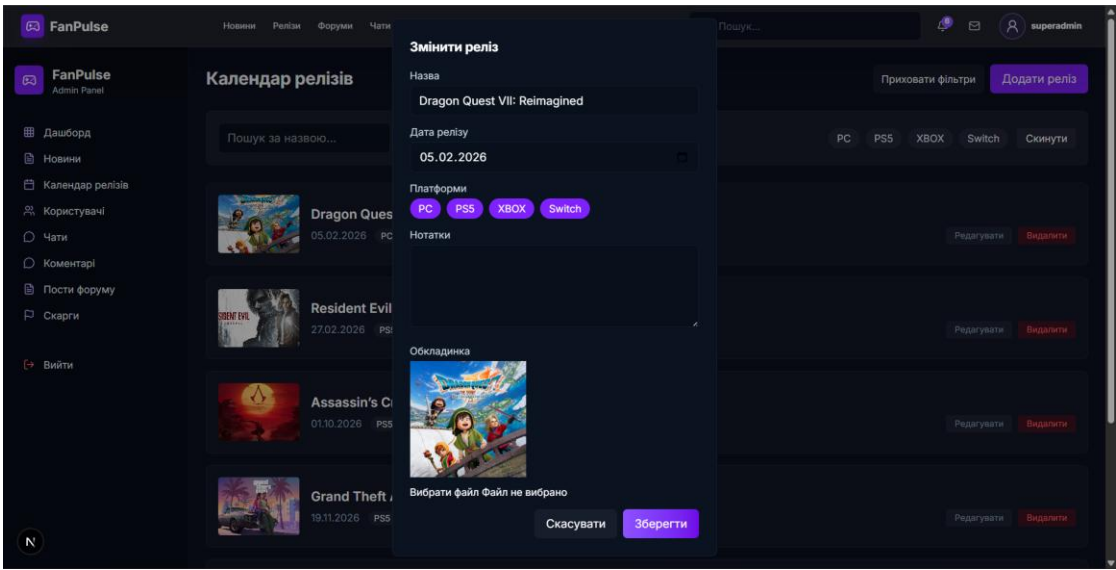


Рис. 3.19. Форма додавання та редагування релізу гри

Розділ управління користувачами (рис. 3.20) реалізований на сторінці /admin/users із можливістю пошуку акаунтів за іменем або email. Адміністратор може переглядати поточні ролі, статуси активності й блокувань, а також змінювати їх для кожного профілю. Оновлені дані (роль, перемикачі is_banned та is_muted) миттєво застосовуються через PATCH-запит до ендпоінту /user/{user_id}.

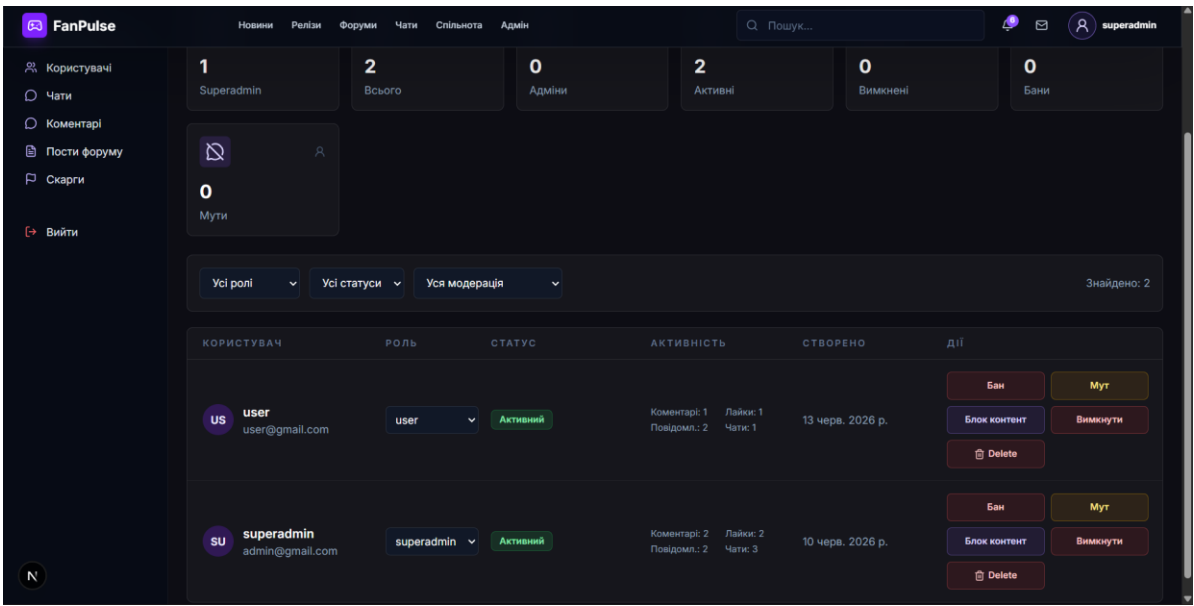


Рис. 3.20. Список користувачів з можливістю зміни ролей та статусів

Розділ Чати в адмінпанелі дозволяє адміністратору переглядати список усіх чатів на платформі та за потреби створювати нові групові чати, додаючи до них учасників вручну. Це може використовуватися, наприклад, для створення офіційних оголошень або тематичних кланових чатів від імені платформи.

Розділ Коментарі (рис. 3.21) об'єднує модерацію коментарів з різних частин сайту: новин, форуму та спільнот. Адміністратор бачить останні додані коментарі у вигляді списку з текстом, автором та посиланням на місце публікації. Якщо коментар порушує правила, його можна видалити одним натисканням кнопки. Видалення виконується через м'яке видалення, тобто запис у базі позначається полем `is_deleted` замість фізичного видалення рядка, що дозволяє за потреби відновити дані.

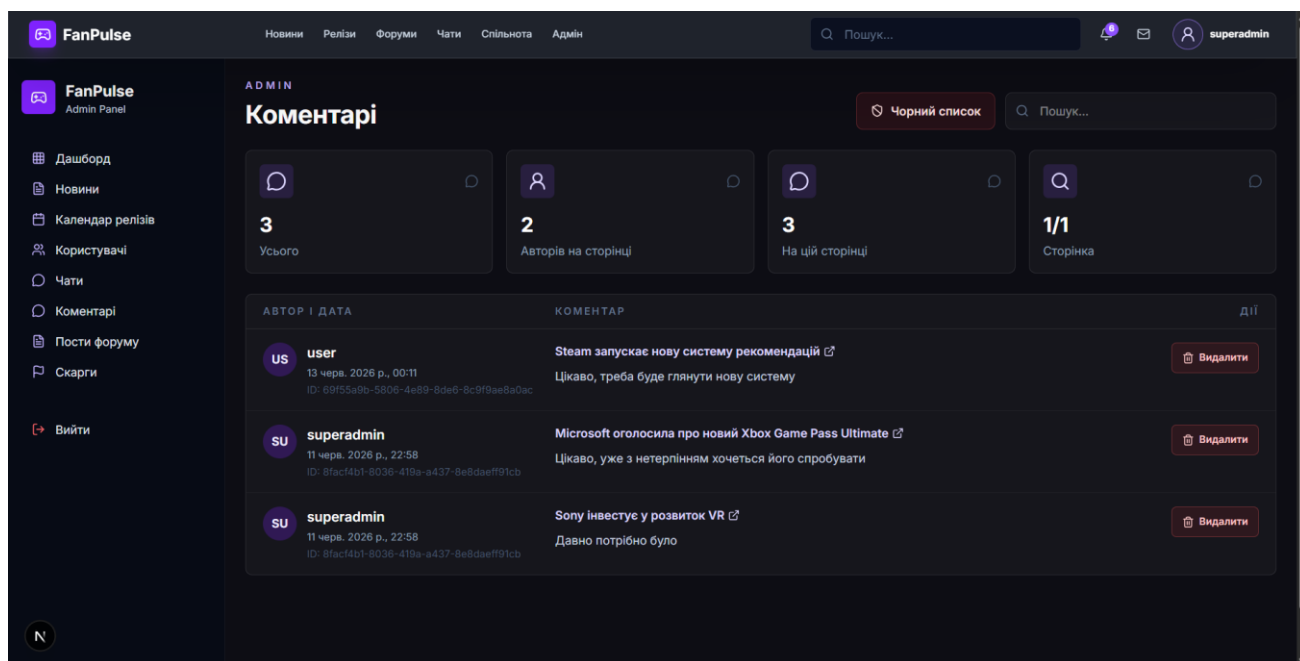


Рис. 3.21. Сторінка модерації коментарів

Розділ Скарги /admin/reports реалізований для розгляду скарг, які користувачі залишають на пости форуму через вбудовану форму повідомлення. Система скарг побудована на основі таблиці `forum_reports`, кожен запис якої містить ідентифікатор поста, ідентифікатор автора скарги, текст причини та поточний статус: `pending` (очікує), `resolved` (вирішено) або `rejected` (відхилено). Додатково передбачено поле `admin_note` для внутрішньої примітки адміністратора.

Інтерфейс сторінки складається з трьох статистичних карток, що відображають кількість скарг, які очікують розгляду, загальну кількість за обраним фільтром і кількість записів на поточній сторінці (рис. 3.22). Під картками розташовані кнопки-фільтри, що дозволяють переключатися між усіма скаргами або конкретним статусом. Список скарг показує для кожного запису: поточний статус з кольоровою міткою, дату надходження, причину скарги, ідентифікатор поста та скорочений ідентифікатор автора скарги. Якщо адміністратор раніше залишив нотатку, вона також відображається у відповідному рядку.

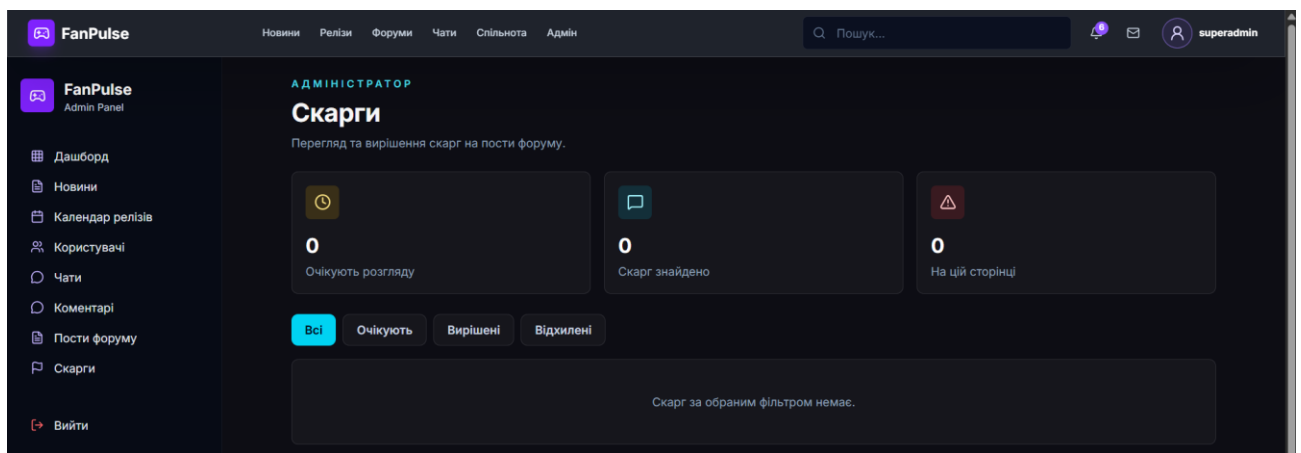


Рис. 3.22. Адміністративна сторінка скарг

Для скарг зі статусом `pending` доступна кнопка «Розглянути». При її натисканні під записом розгортається панель із полем введення текстової примітки та двома кнопками дії. Кнопка «Вирішити» змінює статус скарги на `resolved`, що означає підтвердження порушення та вжиття відповідних заходів. Кнопка «Відхилити» встановлює статус `rejected`, якщо скарга визнана необґрунтованою. Обидва запити надсилаються через хук `useResolveReport` до ендпоінту `PATCH /forum/admin/reports/{report_id}` на бекенді, після чого кеш запитів автоматично інвалідується і список оновлюється без перезавантаження сторінки. Пагінація реалізована з кроком по п'ятнадцять записів на сторінку.

Розділ Форум в адміністративній панелі призначений для централізованого керування структурою та вмістом форумної частини платформи. Адміністратор отримує доступ до статистичних показників, що відображають загальну кількість створених тем, відповідей користувачів та нових публікацій за поточний

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				80
Змн.	Арк.	№ докум.	Підпис	Дата		

день. Це дозволяє оперативно оцінювати активність спільноти та контролювати її розвиток.

Основною функцією модуля є модерація форумних тем (рис. 3.23). Адміністратор може здійснювати пошук тем за назвою, фільтрувати їх за поточним статусом та переходити до перегляду конкретної дискусії. Для кожної теми доступний набір інструментів керування: закріплення у верхній частині списку, зміна статусу на «закрита» або «архівна», а також повне видалення у разі потреби.

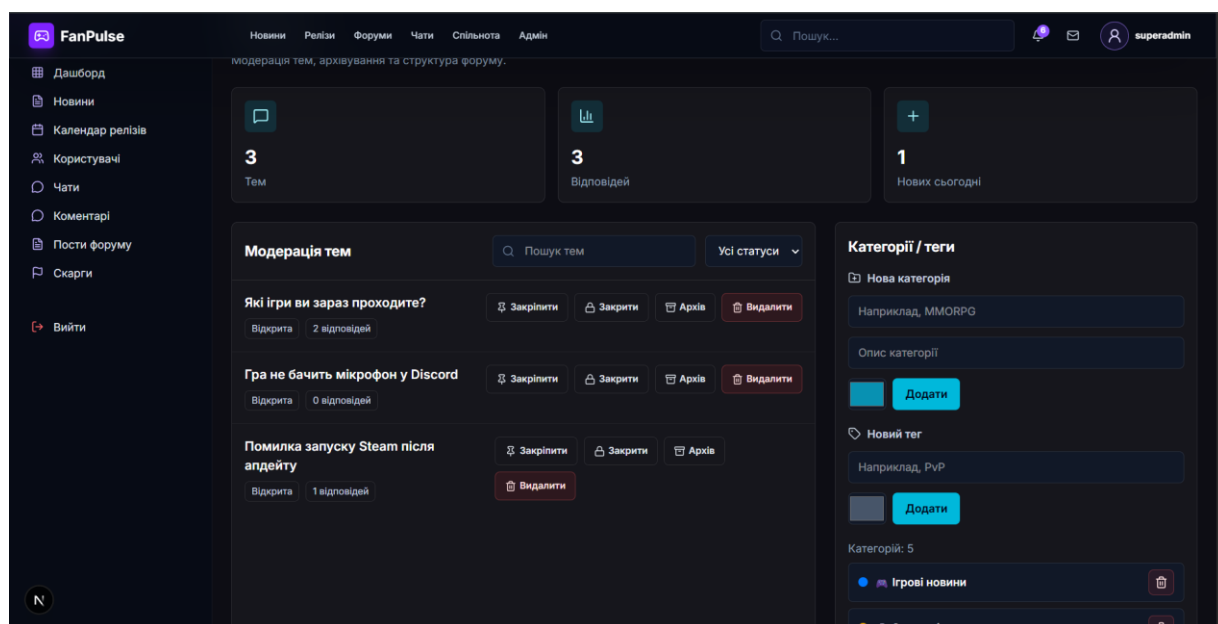


Рис. 3.23. Адміністративна панель керування форумом

Окремим елементом адміністративної панелі є підсистема керування категоріями та тегами форуму. Адміністратор може створювати нові категорії, задаючи їх назву, опис та колірне оформлення, а також видаляти непотрібні категорії. Аналогічно реалізовано роботу з тегами, які використовуються для додаткового тематичного групування обговорень. Для кожного тегу можна задати назву та колір, після чого він стає доступним для використання у форумних темах.

Усі адміністративні операції виконуються відповідно до моделі розмежування прав доступу. Доступ до інструментів модерації, створення та видалення категорій і тегів мають лише користувачі з адміністративними привілеями, тоді як звичайні учасники спільноти можуть лише створювати та переглядати власний контент у межах доступного функціоналу.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				81
Змн.	Арк.	№ докум.	Підпис	Дата		

Підсумовуючи, адміністративний модуль форуму FanPulse забезпечує ефективне керування структурою та контентом форумної підсистеми. Поєднання статистичних інструментів, механізмів модерації тем та засобів керування категоріями і тегами дозволяє підтримувати впорядковану структуру форуму та забезпечувати комфортну взаємодію користувачів із платформою.

Висновки до третього розділу

У третьому розділі розглянуто реалізацію вебплатформи FanPulse та її основних функціональних модулів. Описано процес розробки клієнтської та серверної частин застосунку, а також взаємодію між ними для забезпечення коректної роботи системи.

Під час розробки було створено зручний користувацький інтерфейс для різних категорій користувачів. Неавторизовані відвідувачі можуть переглядати новини, форум та календар ігрових релізів. Після авторизації користувачі отримують доступ до додаткових можливостей, зокрема участі в обговореннях на форумі, спілкування в чатах, коментування та оцінювання публікацій, підписки на нагадування про майбутні релізи, а також створення та керування власними спільнотами.

Для адміністрування платформи реалізовано окрему адміністративну панель. Вона дозволяє керувати новинами, календарем релізів, користувачами, форумом, чатами та коментарями. Також реалізовано систему скарг, за допомогою якої адміністратори можуть переглядати повідомлення про порушення, додавати службові примітки та приймати рішення щодо подальшої обробки скарги.

У процесі розробки враховано питання безпеки та контролю доступу. Для автентифікації та авторизації використано JWT-токени. Також реалізовано розмежування прав доступу між різними категоріями користувачів, перевірку доступу до адміністративних функцій та автоматичну модерацію текстового контенту за допомогою словника заборонених слів.

Після завершення розробки було виконано інтеграцію клієнтської та серверної частин системи, а також проведено тестування основних функціональних

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				82
Змн.	Арк.	№ докум.	Підпис	Дата		

можливостей. Результати тестування підтвердили коректну роботу реалізованих модулів та відповідність системи поставленим вимогам.

Таким чином, у третьому розділі реалізовано основні функції вебплатформи FanPulse та продемонстровано результати їх практичного впровадження. Розроблена система забезпечує зручну взаємодію користувачів, підтримує роботу ігрових спільнот і може бути розширена новими можливостями в подальшому.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				83
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

У ході виконання кваліфікаційної роботи було розроблено соціальну мережу FanPulse для фанатів комп'ютерних ігор. Платформа надає можливість переглядати ігрові новини, брати участь в обговореннях на форумі, спілкуватися в особистих і групових чатах, створювати тематичні спільноти, публікувати власні матеріали та відстежувати дати виходу нових ігор. Основною метою проєкту було створення єдиного середовища для спілкування та взаємодії користувачів, яке поєднує функції кількох популярних сервісів.

У першому розділі досліджено специфіку предметної області та проаналізовано існуючі платформи для ігрових спільнот на прикладі Discord і Twitch. Аналіз показав, що Discord орієнтований на спілкування, а Twitch – на трансляцію контенту. При цьому жоден сервіс не поєднує в одному середовищі чати, новини та форуми. На основі виявлених недоліків сформульовано вимоги до системи FanPulse та визначено її основні функції.

Також у розділі обґрунтовано вибір технологій. Для серверної частини обрано Python і фреймворк FastAPI, а для зберігання даних – PostgreSQL та SQLAlchemy. Обмін повідомленнями в чатах реалізовано через REST API з періодичним оновленням історії. Клієнтську частину побудовано на React та Next.js, що забезпечує швидку роботу інтерфейсу. Для взаємодії з сервером використано axios, а для керування станом застосунку – Zustand.

У другому розділі описано проєктування системи. Визначено функціональні та нефункціональні вимоги, розподілено ролі користувачів та розроблено діаграму варіантів використання з описом основних сценаріїв. Було спроектовано структуру бази даних. Створено набір таблиць для зберігання інформації про користувачів, новини, спільноти, форуми, чати, коментарі та ігрові релізи. Для забезпечення цілісності даних між таблицями налаштовано відповідні зв'язки.

Також у другому розділі розглянуто організацію взаємодії між клієнтською та серверною частинами застосунку. Для обміну даними використовується REST API та формат JSON. Для захисту облікових записів реалізовано систему

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				84
Змн.	Арк.	№ докум.	Підпис	Дата		

авторизації на основі JWT-токенів і розмежування доступу відповідно до ролі користувача.

У третьому розділі описано реалізацію основних модулів платформи. Неавторизовані користувачі можуть переглядати новини, форум і календар релізів, а також проходити реєстрацію та авторизацію. Для авторизованих користувачів доступні додаткові можливості, серед яких коментування, оцінювання публікацій, створення спільнот, участь у чатах та підписка на нагадування про майбутні релізи.

Також реалізовано форум із підтримкою вкладених відповідей, систему спільнот із розподілом ролей між учасниками та чат для обміну повідомленнями в режимі реального часу. Для підтримання порядку на платформі передбачено перевірку текстового контенту за словником заборонених слів та систему скарг на порушення.

Для адміністрування роботи платформи створено окрему адміністративну панель. Вона дозволяє керувати новинами, релізами, користувачами, форумом, коментарями та скаргами. Адміністратор може змінювати ролі користувачів, блокувати облікові записи, модерувати контент і контролювати роботу основних розділів системи.

У результаті виконання роботи було реалізовано всі поставлені завдання. Проведено аналіз існуючих рішень, спроектовано архітектуру системи та базу даних, реалізовано клієнтську і серверну частини застосунку, а також забезпечено механізми безпеки та розмежування доступу. Розроблена платформа FanPulse є працездатним програмним продуктом і може використовуватися як основа для подальшого розвитку та розширення функціональних можливостей системи.

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				85
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Целуєв В.О., Савіцький Р.С.. Методи захисту персональних даних користувачів у геймерських соціальних мережах. Тези XVI Міжнародної науково-технічної конференції "Інформаційно-комп'ютерні технології", 02-03 квітня 2026 року. Житомир : «Житомирська політехніка», 2026. С.492.
2. Целуєв В.О., Савіцький Р.С.. Real-time комунікація у ігровій платформі засобами websocket. Тези XVI Міжнародної науково-технічної конференції "Інформаційно-комп'ютерні технології", 02-03 квітня 2026 року. Житомир : «Житомирська політехніка», 2026. С.494.
3. Axios : Promise based HTTP client [Електронний ресурс]. – URL: <https://axios-http.com/> (дата звернення: 10.04.2026).
4. Chapter 11. indexes. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/indexes.html> (дата звернення: 25.05.2026).
5. Cross-Origin Resource Sharing (CORS) - HTTP | MDN. *MDN Web Docs*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS> (дата звернення: 25.04.2026).
6. Discord - Group Chat That's All Fun & Games [Електронний ресурс]. – URL: <https://discord.com> (дата звернення: 10.06.2026).В
7. FastAPI [Електронний ресурс]. – URL: <https://fastapi.tiangolo.com/> (дата звернення: 20.04.2026).
8. Vercel. Getting started: layouts and pages | next.js. *Next.js by Vercel - The React Framework*. URL: <https://nextjs.org/docs/app/getting-started/layouts-and-pages> (дата звернення: 05.06.2026).
9. Git - Reference [Електронний ресурс]. – URL: <https://git-scm.com/docs> (дата звернення: 15.04.2026).
10. HTTP: Hypertext Transfer Protocol | MDN. *MDN Web Docs*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP> (дата звернення: 20.04.2026).

11. Install tailwind CSS with next.js. *Tailwind CSS - Rapidly build modern websites without ever leaving your HTML*. URL: <https://tailwindcss.com/docs/installation/framework-guides/nextjs> (дата звернення: 20.04.2026).
12. Introduction - Zustand [Електронний ресурс]. – URL: <https://zustand.docs.pmnd.rs/learn/getting-started/introduction> (дата звернення: 21.04.2026).
13. Json â JSON encoder and decoder. *Python documentation*. URL: <https://docs.python.org/3/library/json.html> (дата звернення: 26.04.2026).
14. JSON web token introduction - jwt.io. *JSON Web Tokens - jwt.io*. URL: <https://www.jwt.io/introduction#what-is-json-web-token> (дата звернення: 23.04.2026).
15. Next.js Docs [Електронний ресурс]. – URL: <https://nextjs.org/docs> (дата звернення: 05.06.2026).
16. Vercel. Next.js docs: app router | next.js. *Next.js by Vercel - The React Framework*. URL: <https://nextjs.org/docs/app> (дата звернення: 05.06.2026).
17. Nielsen J. 10 Usability Heuristics for User Interface Design [Електронний ресурс]. – URL: <https://www.nngroup.com/articles/ten-usability-heuristics/> (дата звернення: 10.06.2026).
18. OpenAPI Specification v3.2.0 [Електронний ресурс]. – URL: <https://spec.openapis.org/oas/latest.html> (дата звернення: 23.05.2026).
19. OWASP top ten web application security risks | OWASP foundation. *OWASP Foundation, the Open Source Foundation for Application Security | OWASP Foundation*. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 08.06.2026).
20. Part I. tutorial. *PostgreSQL Documentation*. URL: <https://www.postgresql.org/docs/current/tutorial.html> (дата звернення: 26.04.2026).
21. PostgreSQL: Documentation [Електронний ресурс]. – URL: <https://www.postgresql.org/docs/> (дата звернення: 20.04.2026).

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савицький Р.С.				87
Змн.	Арк.	№ докум.	Підпис	Дата		

22. Pydantic Docs [Електронний ресурс]. – URL: <https://pydantic.dev/docs/> (дата звернення: 27.04.2026).
23. React [Електронний ресурс]. – URL: <https://react.dev/> (дата звернення: 28.04.2026).
24. REST API documentation tool | swagger UI. *API Documentation & Design Tools for Teams | Swagger*. URL: <https://swagger.io/tools/swagger-ui/> (дата звернення: 18.05.2026).
25. Security - FastAPI. *FastAPI* - *FastAPI*. URL: <https://fastapi.tiangolo.com/tutorial/security/> (дата звернення: 19.05.2026).
26. SQL (relational) databases - fastapi. *FastAPI* - *FastAPI*. URL: <https://fastapi.tiangolo.com/tutorial/sql-databases/> (дата звернення: 23.04.2026).
27. SQLAlchemy Documentation [Електронний ресурс]. – URL: <https://docs.sqlalchemy.org/en/20/> (дата звернення: 22.05.2026).
28. Twitch [Електронний ресурс]. – URL: <https://www.twitch.tv/> (дата звернення: 10.06.2026).
29. TypeScript Documentation [Електронний ресурс]. – URL: <https://www.typescriptlang.org/docs/> (дата звернення: 11.05.2026).
30. Welcome to Alembic's Documentation! [Електронний ресурс]. – URL: <https://alembic.sqlalchemy.org/en/latest/> (дата звернення: 29.04.2026).
31. What is REST?: REST API Tutorial [Електронний ресурс]. – URL: <https://restfulapi.net/> (дата звернення: 19.04.2026).

ДОДАТКИ

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				89
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК А

Рис. А.1. ER-діаграма бази даних

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савицький Р.С.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг Б.1 Refresh-токен ротація з виявленням крадіжки (replay detection)

```

async def refresh(self, refresh_token: str) -> TokenResponse:
    """Оновлення access-токена за допомогою валідного refresh-токена."""
    try:
        payload = decode_refresh_token(refresh_token)
    except Exception:
        raise InvalidTokenError("Refresh token недійсний або прострочений")

    user_id_str: str | None = payload.get("sub")
    if not user_id_str:
        raise InvalidTokenError("Пошкоджений токен: відсутній суб'єкт")

    stored = await self.repo.get_refresh_token(refresh_token)

    if stored is None:
        raise TokenNotFoundError()

    # Виявлення повторного використання токена (рев'юкерування сесії)
    if stored.is_revoked:
        await self.repo.revoke_all_user_tokens(uuid.UUID(user_id_str))
        raise InvalidTokenError("Виявлено спробу повторного використання токена. Будь ла-
ска, увійдіть знову.")

    if stored.expires_at < datetime.now(UTC):
        raise InvalidTokenError("Refresh token прострочений")

    user = await self.repo.get_by_id(uuid.UUID(user_id_str))
    if user is None or not user.is_active:
        raise InvalidTokenError("Користувач не знайдений або деактивований")

    # Ротація токенів: відкликаємо старий і видаємо новий
    await self.repo.revoke_refresh_token(refresh_token)
    return await self._issue_tokens(user)

```

Лістинг Б.2 Перевірка типу токена при декодуванні JWT

```

def _create_token(
    subject: str,
    token_type: TokenType,
    expires_delta: timedelta,
    extra_claims: dict | None = None,
) -> str:
    now = datetime.now(UTC)

```

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

payload: dict = {
    "sub": subject,
    "type": token_type,
    "iat": now,
    "exp": now + expires_delta,
}
if extra_claims:
    payload.update(extra_claims)

return jwt.encode(payload, settings.SECRET_KEY, algorithm=settings.JWT_ALGORITHM)

def decode_access_token(token: str) -> dict:
    try:
        payload = jwt.decode(
            token,
            settings.SECRET_KEY,
            algorithms=[settings.JWT_ALGORITHM],
        )
    except JWTError:
        raise

    if payload.get("type") != TokenType.ACCESS:
        raise JWTError("Невірний тип токена: очікувався токен доступу (access token)")

    return payload

```

Лістинг Б.3 Фільтрація заборонених слів у коментарях

```

async def create_comment(
    self, article_id: uuid.UUID, data: NewsCommentCreate, user: User
) -> NewsCommentRead:
    if user.is_muted or user.content_blocked:
        raise HTTPException(
            status_code=status.HTTP_403_FORBIDDEN,
            detail="Ви не можете створювати контент зараз",
        )

    article = await self.repo.get_by_id(article_id)
    if article is None:
        raise HTTPException(status_code=status.HTTP_404_NOT_FOUND, detail="Новину не знайдено")

    text = data.text.strip()
    normalized_text = text.lower()
    banned_words = await self.repo.get_banned_words()

```

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

blocked_word = next(
    (word.word for word in banned_words if word.word and word.word in normalized_text),
    None,
)
if blocked_word:
    raise HTTPException(
        status_code=status.HTTP_422_UNPROCESSABLE_ENTITY,
        detail="Коментар містить заборонені слова",
    )

comment = await self.repo.create_comment(
    article, user_id=user.id,
    user_name=user.username or user.email.split("@")[0],
    text=text,
)
return NewsCommentRead.model_validate(comment)

```

Лістинг Б.4 Каскад залежностей авторизації (RBAC через FastAPI Depends)

```

async def get_current_user(
    credentials: HTTPAuthorizationCredentials | None = Depends(_bearer_scheme),
    db: AsyncSession = Depends(get_db),
) -> User:
    credentials_exception = HTTPException(
        status_code=status.HTTP_401_UNAUTHORIZED,
        detail="Не вдалося перевірити облікові дані",
        headers={"WWW-Authenticate": "Bearer"},
    )

    async def get_current_admin(
        current_user: User = Depends(get_current_user),
    ) -> User:
        if not current_user.role.is_admin_panel_role:
            raise HTTPException(
                status_code=status.HTTP_403_FORBIDDEN,
                detail="Потрібні права адміністратора",
            )
        return current_user

```

Лістинг Б.5 Атомарне оновлення лічильника лайків (race condition protection)

```

sync def create_like(self, post_id: uuid.UUID, user_id: uuid.UUID) -> None:
    """Поставити вподобайку та збільшити лічильник у дописі."""
    self.session.add(ForumLike(post_id=post_id, user_id=user_id))
    await self.session.execute(
        update(ForumPost)

```

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        .where(ForumPost.id == post_id)
        .values(likes_count=ForumPost.likes_count + 1)
    )
    await self.session.flush()

```

Лістинг Б.6 Динамічна побудова запиту: пошук, фільтри, пагінація

```

async def get_topics(
    self,
    *,
    category_id: uuid.UUID | None = None,
    tag_id:    uuid.UUID | None = None,
    status:    str | None      = None,
    search:    str | None      = None,
    sort:      str             = "newest",
    offset:    int             = 0,
    limit:     int             = 20,
) -> tuple[list[ForumTopic], int]:
    """Отримати список тем з фільтрами, пошуком та сортуванням."""
    q = (
        select(ForumTopic)
        .options(
            selectinload(ForumTopic.topic_tags).selectinload(ForumTopicTag.tag),
            selectinload(ForumTopic.category),
        )
        .where(ForumTopic.is_deleted == False)
    )
    if category_id:
        q = q.where(ForumTopic.category_id == category_id)
    if status:
        q = q.where(ForumTopic.status == status)
    if search:
        q = q.where(
            ForumTopic.title.ilike(f"%{search}%") |
            ForumTopic.content.ilike(f"%{search}%")
        )
    if tag_id:
        subq = select(ForumTopicTag.topic_id).where(ForumTopicTag.tag_id == tag_id)
        q = q.where(ForumTopic.id.in_(subq))

    # Логіка сортування
    if sort == "popular":
        q = q.order_by(ForumTopic.is_pinned.desc(), ForumTopic.likes_count.desc())
    elif sort == "views":
        q = q.order_by(ForumTopic.is_pinned.desc(), ForumTopic.views_count.desc())

```

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

else:
    q = q.order_by(ForumTopic.is_pinned.desc(), ForumTopic.created_at.desc())

total = await self.session.scalar(
    select(func.count()).select_from(q.subquery())
) or 0
r = await self.session.execute(q.offset(offset).limit(limit))
return list(r.scalars().all()), total

```

Лістинг Б.7 Зміна PostgreSQL ENUM на проді без втрати даних

def downgrade() -> None:

```

"""Скасування міграції: повернення структури типу userrole до початкового стану."""
# Перепризначаємо користувачів з роллю superadmin на admin перед зміною типу
op.execute("UPDATE users SET role = 'admin' WHERE role = 'superadmin'")

# Видаляємо дефолтне значення, щоб змінити тип колонки
op.execute("ALTER TABLE users ALTER COLUMN role DROP DEFAULT")

# Перейменовуємо старий ENUM-тип
op.execute("ALTER TYPE userrole RENAME TO userrole_old")

# Створюємо оновлений ENUM-тип без superadmin
op.execute("CREATE TYPE userrole AS ENUM ('admin', 'user')")

# Змінюємо тип колонки в таблиці користувачів
op.execute(
    "ALTER TABLE users ALTER COLUMN role TYPE userrole "
    "USING role::text::userrole"
)

# Повертаємо дефолтне значення
op.execute("ALTER TABLE users ALTER COLUMN role SET DEFAULT 'user'")

# Видаляємо старий тип
op.execute("DROP TYPE userrole_old")

```

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТОК В

Таблиця В.1

Перелік endpoint'ів модуля «Автентифікація»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	POST	/auth/register	Реєстрація нового користувача	Публічний
2	POST	/auth/login	Вхід за email та паролем	Публічний
3	POST	/auth/refresh	Оновлення access-токена за refresh-токеном	Публічний
4	POST	/auth/logout	Вихід з поточного пристрою (анулювання refresh-токена)	Публічний
5	GET	/auth/me	Отримати дані поточного користувача	Користувач
6	DELETE	/auth/me	Видалити власний обліковий запис	Користувач

Таблиця В.2

Перелік endpoint'ів модуля «Новини»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/news	Список новин (пошук, фільтр за категорією, пагінація)	Публічний
2	GET	/news/{article_id}	Отримати статтю за ідентифікатором	Публічний
3	GET	/news/{article_id}/comments	Список коментарів до статті	Публічний
4	POST	/news/{article_id}/comments	Додати коментар до статті	Користувач
5	DELETE	/news/{article_id}/comments/{comment_id}	Видалити власний коментар	Користувач
6	POST	/news/{article_id}/like	Поставити/прибрати лайк статті	Користувач
7	POST	/news	Створити статтю	Адмін
8	PATCH	/news/{article_id}	Оновити статтю	Адмін
9	DELETE	/news/{article_id}	Видалити статтю	Адмін
10	POST	/news/upload-image	Завантажити зображення для статті	Адмін

продовж. табл В.2

11	GET	/news/admin/comments	Список усіх коментарів до новин	Адмін
12	DELETE	/news/admin/comments/{article_id}/{comment_id}	Видалити будь-який коментар	Адмін
13	GET	/news/admin/banned-comment-words	Список заборонених слів у коментарях	Адмін
14	POST	/news/admin/banned-comment-words	Додати заборонене слово	Адмін

Таблиця В.3

Перелік endpoint'ів модуля «Форум»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/forum/categories	Список категорій форуму	Публічний
2	GET	/forum/tags	Список тегів форуму	Публічний
3	GET	/forum/topics	Список тем (фільтри, пошук, сортування, пагінація)	Публічний
4	GET	/forum/topics/{topic_id}	Отримати тему за ідентифікатором	Публічний
5	GET	/forum/topics/{topic_id}/posts	Список постів (відповідей) у темі	Публічний
6	POST	/forum/topics	Створити нову тему	Користувач
7	PATCH	/forum/topics/{topic_id}	Редагувати тему	Автор/Адмін
8	DELETE	/forum/topics/{topic_id}	Видалити тему	Автор/Адмін
9	POST	/forum/topics/{topic_id}/subscribe	Підписатися/відписатися від теми	Користувач
10	POST	/forum/topics/{topic_id}/posts	Створити пост (відповідь) у темі	Користувач
11	PATCH	/forum/posts/{post_id}	Редагувати пост	Автор/Адмін
12	DELETE	/forum/posts/{post_id}	Видалити пост	Автор/Адмін
13	POST	/forum/posts/{post_id}/like	Лайк/дизлайк поста	Користувач
14	POST	/forum/posts/{post_id}/report	Поскаржитися на пост	Користувач

продовж. табл В.3

15	POST	/forum/categories	Створити категорію	Адмін
16	PATCH	/forum/categories/{cat_id}	Оновити категорію	Адмін
17	DELETE	/forum/categories/{cat_id}	Видалити категорію	Адмін
18	POST	/forum/tags	Створити тег	Адмін
19	DELETE	/forum/tags/{tag_id}	Видалити тег	Адмін
20	GET	/forum/admin/stats	Статистика форуму	Адмін
21	GET	/forum/admin/reports	Список скарг	Адмін
22	PATCH	/forum/admin/reports/{report_id}	Розглянути скаргу	Адмін
23	PATCH	/forum/admin/topics/{topic_id}/status	Змінити статус теми (open/closed/archived)	Адмін
24	PATCH	/forum/admin/topics/{topic_id}/pin	Закріпити/відкріпити тему	Адмін

Таблиця В.4

Перелік endpoint'ів модуля «Користувач»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/user/me	Отримати власний профіль	Користувач
2	PATCH	/user/me	Оновити власний профіль	Користувач
3	POST	/user/me/avatar	Завантажити аватар	Користувач
4	GET	/user	Список усіх користувачів	Адмін
5	PATCH	/user/{user_id}	Оновити користувача (роль, статуси)	Адмін
6	DELETE	/user/{user_id}	Видалити користувача	Адмін

Таблиця В.5

Перелік endpoint'ів модуля «Релізи»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/releases	Список релізів ігор (фільтри: статус, жанр, платформа)	Публічний
2	GET	/releases/{release_id}	Отримати реліз за ідентифікатором	Публічний
3	POST	/releases/{release_id}/remind	Перемкнути нагадування про реліз	Користувач

продовж. табл В.5

4	POST	/releases	Створити реліз	Адмін
5	PATCH	/releases/{release_id}	Оновити реліз	Адмін
6	DELETE	/releases/{release_id}	Видалити реліз	Адмін
7	POST	/releases/upload-cover	Завантажити обкладинку релізу	Адмін

Таблиця В.6

Перелік endpoint'ів модуля «Адміністрування чатів»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/admin/chats	Список усіх чатів (з фільтрами)	Адмін
2	POST	/admin/chats	Створити груповий чат	Адмін
3	GET	/admin/chats/{chat_id}	Отримати чат за ідентифікатором	Адмін
4	PATCH	/admin/chats/{chat_id}	Оновити чат (назву, опис, статус)	Адмін
5	DELETE	/admin/chats/{chat_id}	Видалити чат і всі його повідомлення	Адмін
6	GET	/admin/chats/{chat_id}/members	Отримати список учасників чату	Адмін
7	POST	/admin/chats/{chat_id}/members	Додати учасників до чату	Адмін
8	PATCH	/admin/chats/{chat_id}/members/{user_id}	Оновити роль учасника чату	Адмін
9	DELETE	/admin/chats/{chat_id}/members/{user_id}	Видалити учасника з чату	Адмін
10	GET	/admin/chats/{chat_id}/messages	Отримати повідомлення чату (включно з видаленими)	Адмін
11	POST	/admin/chats/{chat_id}/messages	Надіслати повідомлення у чат	Адмін
12	DELETE	/admin/chats/{chat_id}/messages/{message_id}	Видалити повідомлення (soft delete)	Адмін

Таблиця В.7

Перелік endpoint'ів модуля «Чати»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/chats	Список моїх чатів	Користувач

		Целуєв В.О.			ІПЗ.КР.Б-121-26-ПЗ	Арк.
		Савіцький Р.С.				99
Змн.	Арк.	№ докум.	Підпис	Дата		

продовж. табл В.7

2	GET	/chats/{chat_id}/messages	Отримати повідомлення з мого чату	Користувач
3	POST	/chats/{chat_id}/messages	Надіслати повідомлення у мій чат	Користувач

Таблиця В.8

Перелік endpoint'ів модуля «Спільноти»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/communities	Список спільнот (пошук, пагінація)	Публічний
2	GET	/communities/{community_id}	Отримати деталі спільноти	Публічний
3	GET	/communities/{community_id}/members	Список учасників спільноти	Публічний
4	POST	/communities	Створити нову спільноту	Користувач
5	POST	/communities/upload-avatar	Завантажити аватар спільноти	Користувач
6	POST	/communities/{community_id}/join	Вступити до спільноти	Користувач
7	DELETE	/communities/{community_id}/leave	Покинути спільноту	Користувач
8	PATCH	/communities/{community_id}	Оновити спільноту	Власник
9	DELETE	/communities/{community_id}	Видалити спільноту	Власник
10	PATCH	/communities/{community_id}/members/{user_id}/role	Змінити роль учасника	Власник
11	POST	/communities/{community_id}/bans	Заблокувати користувача у спільноті	Власник/Модератор
12	GET	/communities/{community_id}/bans	Список заблокованих користувачів	Власник/Модератор
13	DELETE	/communities/{community_id}/bans/{user_id}	Розблокувати користувача	Власник/Модератор

Таблиця В.9

Перелік endpoint'ів модуля «Пости спільнот»

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/communities/{community_id}/posts	Список постів спільноти (пагінація)	Публічний
2	GET	/posts/{post_id}	Отримати деталі поста	Публічний
3	GET	/posts/{post_id}/comments	Отримати коментарі до поста	Публічний
4	POST	/communities/{community_id}/posts	Створити пост у спільноті	Користувач
5	PATCH	/posts/{post_id}	Редагувати пост	Автор
6	DELETE	/posts/{post_id}	Видалити пост	Автор/Модератор
7	POST	/posts/{post_id}/like	Лайк/дизлайк поста	Користувач
8	POST	/posts/{post_id}/comments	Додати коментар до поста	Користувач
9	DELETE	/comments/{comment_id}	Видалити коментар	Автор/Модератор

Таблиця В.10

Зведена структура API за модулями

Кількість endpoint'ів	Функціональний модуль
6	Модуль «Автентифікація» (/auth)
15	Модуль «Новини» (/news)
24	Модуль «Форум» (/forum)
6	Модуль «Користувач» (/user)
7	Модуль «Релізи» (/releases)
12	Модуль «Адмін – чати» (/admin/chats)
3	Модуль «Чати» (/chats)
13	Модуль «Спільноти» (/communities)
9	Модуль «Пости спільнот» (/posts, /comments)
1	Перевірка працездатності сервісу (/health)
95	Усього

Перелік endpoint'ів перевірки працездатності

№	Метод	Шлях (відносно /api/v1)	Опис	Доступ
1	GET	/health	Перевірка працездатності сервісу (статус, версія API)	Публічний

Опис прецеденту «Листування в чаті»

Назва варіанту використання	Листування в чаті
Основні актори	Зареєстрований користувач, Адміністратор
Короткий опис	Авторизований користувач або адміністратор має право надсилати та отримувати повідомлення в режимі реального часу в межах існуючих чатів для ведення комунікації.

Таблиця Г.2

Опис прецеденту «Управління чатами»

Назва варіанту використання	Управління чатами
Основні актори	Адміністратор
Короткий опис	Адміністратор має право створювати нові чати та додавати до них учасників, формуючи середовище для подальшої комунікації.

Таблиця Г.3

Опис прецеденту «Вступ до спільноти»

Назва варіанту використання	Вступ до спільноти
Основні актори	Зареєстрований користувач, Адміністратор
Короткий опис	Зареєстрований користувач або адміністратор має право приєднуватися до обраних кланів або спільнот для участі у груповій діяльності платформи.

Таблиця Г.4

Опис прецеденту «Ручна модерація контенту»

Назва варіанту використання	Ручна модерація контенту
Основні актори	Адміністратор
Короткий опис	Адміністратор має право контролювати потік публікацій, переглядати контент та видаляти чи редагувати повідомлення, які не були виявлені автоматичною фільтрацією, але порушують правила платформи.

Опис прецеденту «Керування правилами модерації»

Назва варіанту використання	Керування правилами модерації
Основні актори	Адміністратор, Система
Короткий опис	Адміністратор формує список заборонених слів, на основі якого система автоматично здійснює перевірку публікацій користувачів та блокує контент, що порушує встановлені правила.

Опис прецеденту «Перегляд аналітики»

Назва варіанту використання	Керування правилами модерації
Основні актори	Адміністратор
Короткий опис	Адміністратор має право переглядати статистичні дані про активність користувачів та стан платформи через адмін-дашборд.

Опис прецеденту «Реєстрація та вхід»

Назва варіанту використання	Реєстрація та вхід
Основні актори	Гість
Короткий опис	Неzareєстрований користувач має право створити обліковий запис або увійти в систему для отримання доступу до персоналізованого функціоналу.