ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
МІСЬКОГО ГОСПОДАРСТВА ІМЕНІ О. М. БЕКЕТОВА

Пояснювальна записка
до кваліфікаційної роботи бакалавра

на тему: Створення соціальної платформи для пошуку людей задля організації спільних активностей

Виконав: здобувач вищої освіти 4 курсу,
групи КН 2022-1
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

Андрій ДЕМЧУК
(прізвище та ініціали)

Керівник: Юрій ЛЕВІКОВ
(прізвище та ініціали)

Рецензент: Володимир БРЕДІХІН
(прізвище та ініціали)

м. Харків – 2026 рік

Харківський національний університет міського господарства імені О. М. Бекетова
(повне найменування закладу вищої освіти)
Навчально-науковий Інститут енергетичної, інформаційної та транспортної інфраструктури
Кафедра Комп'ютерних наук та інформаційних технологій
Рівень вищої освіти перший (бакалаврський)
Спеціальність 122 Комп'ютерні науки
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри КНтаІТ


Марина Новожилова

«_22_» червня__2026 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Демчук Андрій Юрійович
(прізвище, ім'я, по батькові)

1. Тема роботи Створення соціальної платформи для пошуку людей задля організації спільних активностей

керівник роботи Левіков Юрій Володимирович ст. викладач
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом закладу вищої освіти від «22» травня 2026 р. № 440-03

2. Термін подання роботи здобувачем вищої освіти

3. Вихідні дані до роботи Проєктування та реалізація соціальної платформи для пошуку людей задля організації спільних активностей

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Аналіз предметного середовища, дослідження сучасних підходів до побудови серверної та клієнтської частини соціальної платформи, аналіз існуючих аналогів, реалізація інтерфейсів користувача.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Презентація (18 слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ I	Юрій ЛЕВІКОВ, старший викладач кафедри КН та ІТ	11.05.2026	23.05.2026
Розділ II	Юрій ЛЕВІКОВ, старший викладач кафедри КН та ІТ	24.05.2026	02.06.2026
Розділ III	Юрій ЛЕВІКОВ, старший викладач кафедри КН та ІТ	03.06.2026	10.06.2026
Розділ IV	Вікторія МАЛИШЕВА, к. т., н., доцент кафедри ОП та БЖ	11.06.2026	14.06.2026

7. Дата видачі завдання 11.05.2026

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Вибір тема дипломної роботи	11.05.2026	Виконано
2	Затвердження тем, наукових керівників та календарного плану підготовки кваліфікованої роботи	15.05.2026	Виконано
3	Написання I розділу	23.05.2026	Виконано
4	Написання II розділу	02.06.2026	Виконано
5	Написання III розділу	10.06.2026	Виконано
6	Написання IV розділу	14.06.2026	Виконано
7	Подання дипломної роботи керівнику	15.06.2026	Виконано
8	Робота по усуненню зауважень керівника, уточнення і доповнення практичного матеріалу, оформлення додатків до роботи	16.06.2026	Виконано
9	Подання допрацьованого варіанту роботи керівнику	16.06.2026	Виконано
10	Захист матеріалів дипломної роботи на засіданні кафедри	18.06.2026	Виконано
11	Офіційний захист матеріалів дипломної роботи на засіданні екзаменаційної комісії	23.06.2026	Виконано

Здобувач вищої освіти _____ Андрій ДЕМЧУК
(підпис) (прізвище та ініціали)

Керівник роботи _____ Юрій ЛЕВІКОВ
(підпис) (прізвище та ініціали)"

АНОТАЦІЯ

Структура та обсяг роботи.

Пояснювальна записка кваліфікаційної роботи бакалавра студента групи КН 2022-1 спеціальності 122 Комп’ютерні науки Демчука Андрія Юрійовича темою «Створення соціальної платформи для пошуку людей задля організації спільних активностей» має 4 розділи, 69 сторінки, 19 рисунків, 14 таблиць, 16 джерел.

Метою дипломного проєкту є розробка веб-орієнтованої платформи для пошуку людей здаля організації спільних активностей.

Об’єктом досліджень є процес організації та координації спільних активностей за допомогою сучасних веб-технологій.

Предметом дослідження є архітектурні шаблони, моделі зберігання даних та алгоритми обробки запитів які використовуються для під час створення серверної та клієнтської частини соціальної платформи.

Основне завдання роботи спроєктувати та програмно реалізувати повнофункціональну веб-платформу, що забезпечує створення активностей за категоріями, управляння заявками учасників, обмін повідомленнями у реальному часі, систему сповіщень та інструментарій модерацій спільноти з механізмом банів.

Результатом роботи є готова веб-платформа яка може бути впроваджена як інструмент соціальної взаємодії для організації спільного дозвілля.

Ключові слова: ВЕБ-ПЛАТФОРМА, СЕРВІС, СОЦІАЛЬНА ПЛАТФОРМА, КОРИСТУВАЧІ, АКТИВНОСТІ, РЕАЛЬНИЙ ЧАС.

ANNOTATION

Structure and scope of work.

Explanatory note of the bachelor's qualification work of the student of group KN 2022-1, specialty 122 Computer Science, Demchuk Andriy Yuriyovych, on the topic "Creating a social platform for searching for people to organize joint activities" has 4 sections, 69 pages, 19 figures, 14 tables, 16 sources.

The goal of the thesis project is to develop a web-based platform for finding people remotely to organize joint activities.

The object of research is the process of organizing and coordinating joint activities using modern web technologies.

The subject of the study is architectural patterns, data storage models, and query processing algorithms used when creating the server and client parts of a social platform.

The main task of the work is to design and programmatically implement a fully functional web platform that provides the creation of activities by category, management of participant applications, real-time messaging, a notification system, and a community moderation toolkit with a ban mechanism.

The result of the work is a ready-made web platform that can be implemented as a social interaction tool for organizing joint leisure activities.

Keywords: WEB PLATFORM, SERVICE, SOCIAL PLATFORM, USERS, ACTIVITIES, REAL TIME.

ЗМІСТ

ВСТУП

Дослідження присвячено проєктуванню та програмній реалізації веб-платформи "Hangout", яка призначена для автоматизації процесів пошуку однодумців, спільної організації некомерційних локальних заходів, координації учасників у реальному часі та підтримки порядку у спільноті засобами модерації.

Актуальність теми. Процес глобальної цифровізації та перехід значної частини комунікацій у віртуальний простір призвели до загострення проблеми соціальної ізоляції молоді. Існуючі великі соціальні платформи орієнтовані на медіаспоживання, ведення особистих блогів або організацію масштабних комерційних подій. Натомість інструменти для швидкої та зручної координації спонтанного некомерційного дозвілля у малих групах відсутні або мають фрагментарний характер. Створення спеціалізованого веб-сервісу з гнучкою структурою даних, яка адаптується під конкретні категорії активностей, у поєднанні з вбудованим каналом комунікації між учасниками та системою адресних сповіщень дозволить оптимізувати процес пошуку партнерів для спільного проведення часу в онлайн та офлайн форматі та знизити організаційні бар'єри для користувачів.

Метою дослідження є проєктування та розробка високопродуктивної веб-платформи для пошуку людей та спільної організації некомерційних активностей, що охоплює розробку клієнтської частини та асинхронного сервера.

Для досягнення поставленої мети визначено такі завдання дослідження:

- виконати аналіз предметної області та існуючих рішень;
- спроєктувати гібридну архітектуру збереження даних
- розробити клієнтську частину а також асинхронну серверну частину веб-платформи;
- реалізувати підсистеми комунікації в реальному часі, фонових задач та модерації;

- провести автоматизоване тестування створених модулів.

Об'єктом дослідження є процес організації та координації спільних некомерційних активностей за допомогою сучасних інформаційних веб-технологій.

Предметом дослідження є архітектурні шаблони, технологічні стеки, моделі зберігання даних та алгоритми асинхронної обробки запитів, які використовуються під час створення серверної частини соціальної платформи для пошуку однодумців.

Практична цінність отриманих результатів полягає у створенні готової до розгортання веб-платформи, клієнтська частина якої реалізована як Single Page Application на базі Vue 3, TypeScript та Pinia, а серверна частина – на FastAPI із використанням реляційних та документних баз даних. Розроблена система може бути використана як уніфіковане ядро для створення різноманітних клієнтських додатків соціального та організаційного спрямування.

РОЗДІЛ 1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

Веб-платформа "Hangout" належить до сфери соціальних інтернет-сервісів, орієнтованих на організацію спільного дозвілля та локальних офлайн-заходів. Цей сегмент знаходиться на стику двох індустрій: соціальних мереж, які відповідають за побудову міжособистісних контактів на основі спільних інтересів, та автоматизованих систем управління подіями (event-management). Основна мета розробки полягає у створенні інструменту для некомерційного використання, що дозволяє користувачам знаходити компанію для спільного проведення часу у малих групах від 2 до 20 осіб та узгоджувати організаційні деталі заходу безпосередньо в межах платформи.

Актуальність розробки програмного забезпечення такого типу підтверджують соціологічні дослідження останніх років. Згідно з аналітичною доповіддю [1], сучасне суспільство стикається з масштабною кризою дефіциту спілкування, яку визначають як епідемію самотності та ізоляції. Брак живих соціальних зв’язків негативно впливає на ментальний стан індивіда, підвищує рівень стресу та знижує загальне відчуття безпеки в громадах. Аналогічні висновки наведено у глобальному дослідженні [2], де фізичне та психологічне благополуччя особистості прямо пов’язують із рівнем її інтеграції у локальні соціальні групи. Наявність інструментів, що полегшують перехід від віртуального спілкування до реальної взаємодії через спільне дозвілля, є фізіологічною та соціальною потребою сучасної молоді.

Існуюче інформаційне середовище пропонує обмежену кількість рішень для організації спонтанних зустрічей невеликого масштабу. Більшість популярних платформ орієнтовані або на масові публічні комерційні заходи, або на ведення особистих блогів.

Проєкт "Hangout" фокусується на неформальних активностях молоді, які охоплюють сім основних категорій:

- відеоігри (Games) – із фіксацією назви гри, платформи проведення (PC, PlayStation, Xbox, Nintendo, Mobile, Cross-platform), жанру та інтеграцією із зовнішнім каталогом;
- настільні ігри (Board Games) – із зазначенням назви, кількості гравців та рівня складності правил;
- кіно (Movies) – із вказівкою назви фільму, жанру та місця проведення сеансу;
- аніме (Anime) – із фіксацією назви, діапазону серій для спільного перегляду та жанрової приналежності;
- спорт (Sport) – із зазначенням виду спорту, рівня підготовки учасників та вимог до спортивного інвентарю;
- музика (Music) – із визначенням жанру та ролі учасника, наприклад, слухач чи виконавець;
- їжа (Foods) – для організації спільних візитів до закладів харчування без додаткових умов.

Кожна створювана активність класифікується за двома ознаками. Перша ознака – це тип доступу, що може бути відкритим (open), коли приєднання відбувається автоматично, або закритим (closed), що передбачає підтвердження заявки організатором. Друга ознака – це формат проведення, який ділить заходи на онлайн-активності (online) та офлайн-зустрічі (offline) з обов’язковою географічною прив’язкою до місця проведення. Кожна подія проходить через визначений життєвий цикл станів – від активної фази збору учасників (active) через позначку про початок (started) та завершення (ended) до архівації (archived); окремо передбачені аварійні стани скасування організатором (canceled).

Автоматизація цієї сфери стикається з шістьма основними проблемами предметної області:

- високий рівень фрагментації пошуку партнерів для дозвілля, оскільки

оголошення про пошук людей розкидані по різних тематичних чатах у месенджерах та групах у соціальних мережах, де вони швидко втрачають актуальність і губляться в потоці інформації;

- відсутність уніфікованої структури даних для опису специфічних параметрів різних видів дозвілля, через що користувачам доводиться з'ясовувати технічні деталі (наприклад, сумісність ігрових платформ чи рівень підготовки у спорті) через довге особисте листування;
- відсутність автоматизованого контролю ліміту учасників та механізму обробки черги заявок у приватних групах, що призводить до організаційного хаосу під час координації зустрічей;
- необхідність переключатися між кількома застосунками для організації події та для подальшого спілкування з її учасниками, оскільки на існуючих платформах процес знайомства та оперативна комунікація щодо самого заходу штучно розірвані;
- брак адресних повідомлень про релевантні події в зоні інтересів користувача, через що користувачі змушені вручну переглядати велику кількість оголошень для пошуку придатних активностей;
- відсутність дієвих інструментів підтримки порядку в спільноті, що проявляється у поширенні токсичних повідомлень, спаму та фейкових оголошень за умови повної відсутності механізмів скарг чи модераторського втручання;
- потреба в ручному пошуку метаданих та зображень для візуального оформлення подій, що знижує швидкість створення публікацій.

Розв'язання цих проблем потребує розробки спеціалізованого веб-сервісу з гнучкою структурою зберігання даних, яка підлаштовується під конкретний тип активності, з вбудованим каналом синхронної комунікації між учасниками, з адресною системою сповіщень та з повноцінним модераторським інструментарієм, а також з асинхронною обробкою черг та інтеграцією із зовнішніми джерелами інформації.

.

1.1.1 Опис процесу діяльності

Процес діяльності, що автоматизується розроблюваним програмним продуктом, є повним життєвим циклом організації спільного заходу. Цей цикл складається з п’яти послідовних етапів.

Перший етап – реєстрація користувача в системі. Користувач заповнює форму реєстрації через клієнтський інтерфейс, вказуючи адресу електронної пошти, унікальне ім’я користувача та пароль. Система виконує валідацію введених даних: пароль має містити щонайменше 8 символів, серед яких обов’язково є одна велика літера та одна цифра; ім’я користувача обмежується довжиною від 3 символів і може містити лише латинські літери, цифри та символ підкреслення. Далі виконується перевірка на унікальність імені та пошти за реляційною базою даних. Якщо дані унікальні, пароль зберігається у захешованому вигляді за допомогою безпечного стійкого алгоритму, а запис фіксується в базі даних із присвоєнням стандартної ролі клієнта. Користувач отримує можливість налаштувати свій профіль, додати біографію, вибрати інтереси або завантажить аватар та фоновий банер.

Другий етап – автентифікація та авторизація. Для входу в систему користувач надсилає логін та пароль. Після успішної перевірки автентифікаційних даних система генерує пару токенів: короткоживучий токен доступу та довгоживучий токен оновлення. Довгоживучий токен записується в захищені HTTP-only cookie клієнта. Унікальний ідентифікатор сесії зберігається в оперативному сховищі для реалізації механізму ротації та контролю активних сесій. Поточна версія сесії фіксується в записі користувача в реляційній базі даних та використовується для централізованого скасування всіх активних токенів у разі зміни пароля або блокування.

Третій етап – створення активності. Авторизований користувач ініціює створення події. Він заповнює обов'язкові загальні поля: заголовок, опис, категорія, формат, тип доступу, ліміт учасників та дата проведення. Дата проведення події обов'язково має бути запланована щонайменше за дві години від поточного часу сервера. Крім загальних полів, користувач заповнює

специфічний блок даних, структура якого залежить від вибраної категорії. Система виконує автоматичну валідацію цієї структури на відповідність типу категорії за допомогою системи валідації схем. Якщо формат проведення визначено як офлайн, обов'язково вказується географічна локація. Після валідації запис додається до документної бази даних. Теги події автоматично обробляються в реляційній базі даних за допомогою операції злиття з підрахунком частоти використання кожного тегу. Якщо створюється подія категорії "відеоігри", система надсилає повідомлення в асинхронну чергу задач для фонового завантаження обкладинки гри з зовнішнього каталогу через фоновий обробник. Перед збереженням нового запису система перевіряє відсутність часових конфліктів із іншими активностями організатора. Якщо дата завершення не вказана, система автоматично розраховує її як дата початку плюс дві години. Паралельно ініціюється задача розсилання адресних сповіщень користувачам, які додали відповідні теги до своїх улюблених.

Четвертий етап – пошук подій та перегляд стрічки. Користувач здійснює запит до системи для отримання списку доступних активностей. Система виконує фільтрацію документів у документній базі даних на основі активного статусу події та часу її початку з підтримкою додаткових фільтрів за категорією, форматом, набором тегів та діапазоном дат. Видача результатів реалізована за допомогою курсорної пагінації. Для зменшення навантаження на систему під час відображення стрічки, інформація про авторів подій завантажується з реляційної бази даних одним пакетним запитом.

П'ятий етап – подача заявки та участь у події. Користувач вибирає активність і надсилає запит на участь. Якщо подія має відкритий тип доступу, користувач автоматично додається до списку учасників, а лічильник поточної кількості членів збільшується на одиницю за допомогою асинхронної операції. Якщо подія є закритою, створюється запит зі статусом очікування (pending). Організатор події отримує адресне сповіщення і може схвалити або відхилити запит. Перед схваленням заявки на подію система додатково перевіряє відсутність часових перетинів з іншими активностями, у яких користувач вже

бере участь, що унеможливлює одночасну присутність у двох місцях. Система контролює, щоб кількість підтверджених учасників не перевищувала встановлений ліміт події.

Шостий етап – синхронна комунікація учасників події. Після підтвердження участі користувач отримує доступ до окремого чату активності, який функціонує на основі протоколу WebSocket. Сервер підтримує постійне з'єднання та передає повідомлення усім підключеним учасникам через менеджер WebSocket-каналів. Повідомлення проходять автоматичну нормалізацію тексту для забезпечення коректного відображення. Окрім текстових повідомлень, у потік чату транслюються системні події життєвого циклу. Дані про авторів зберігаються разом із повідомленнями для збереження цілісності історії чату навіть після видалення облікового запису автора. Окрема система обмежень частоти повідомлень із накопиченням порушень в оперативній пам'яті процесу захищає чат від спаму та примусово розриває з'єднання користувача при перевищенні допустимого ліміту.

Сьомий етап – автоматичне керування життєвим циклом активності. Спеціалізовані фонові задачі автоматично змінюють статус події за переходами active → started → ended → archived. За дві години до запланованого початку всім підтвердженим учасникам надсилається сповіщення-нагадування. У момент настання дати початку статус події переходить у started, після завершення – в ended з одночасним збільшенням лічильника відвіданих заходів у профілях усіх присутніх учасників. Через визначений проміжок часу запис активності переходить у термінальний стан archived. Окрема задача автоматично знімає тимчасові бани користувачів після завершення їх строку дії.

Восьмий етап – модерація спільноти та контроль порядку. Будь-який зареєстрований користувач має право подати скаргу на інший обліковий запис, конкретну активність або повідомлення в чаті із зазначенням обґрунтування довжиною від 10 до 1000 символів. Скарга створюється зі статусом open та потрапляє у чергу модерації. Модератор бере скаргу в роботу, переглядає

історію раніше поданих скарг на користувача та ухвалює одне з рішень: задовольнити скаргу з накладенням бану або відхилити її як необґрунтовану. Накладення бану миттєво анулює всі активні сесії порушника, що гарантує негайне відключення від платформи.

Процес також передбачає альтернативні та аварійні сценарії:

- ротація токенів: при закінченні терміну дії токена доступу клієнт надсилає запит на оновлення; система вилучає старий ідентифікатор з оперативного сховища даних та генерує нову пару токенів; якщо токен намагаються використати повторно, система анулює всі сесії користувача;
- обробка завантаження медіафайлів: при завантаженні зображення профілю система спочатку зберігає новий шлях до файлу в реляційній базі даних, а потім завантажує файл у об'єктне сховище; у разі помилки завантаження виконується компенсуюча транзакція, яка повертає попереднє значення шляху до файлу в базі даних;
- асинхронне отримання обкладинок: фоновий обробник робить запит до зовнішнього каталогу, обробляє зображення, завантажує у об'єктне сховище та оновлює статус обкладинки в документній базі даних; у разі помилки мережі задача автоматично повторюється з експоненціальною затримкою;
- скасування події організатором: подія переходить у статус canceled, усім учасникам розсилається сповіщення, а активні WebSocket з'єднання чату примусово закриваються;
- дедуплікація адресних сповіщень: система запобігає дублюванню сповіщень навіть у разі повторної обробки задач;
- періодична перевірка цілісності гібридного сховища: окрема задача регулярно зіставляє ідентифікатори зовнішніх ключів між базами даних та виявляє посилання на видалені записи користувачів.

Схематичне представлення повного життєвого циклу процесу діяльності у вигляді діаграми діяльності UML наведено на рисунку 1.1.

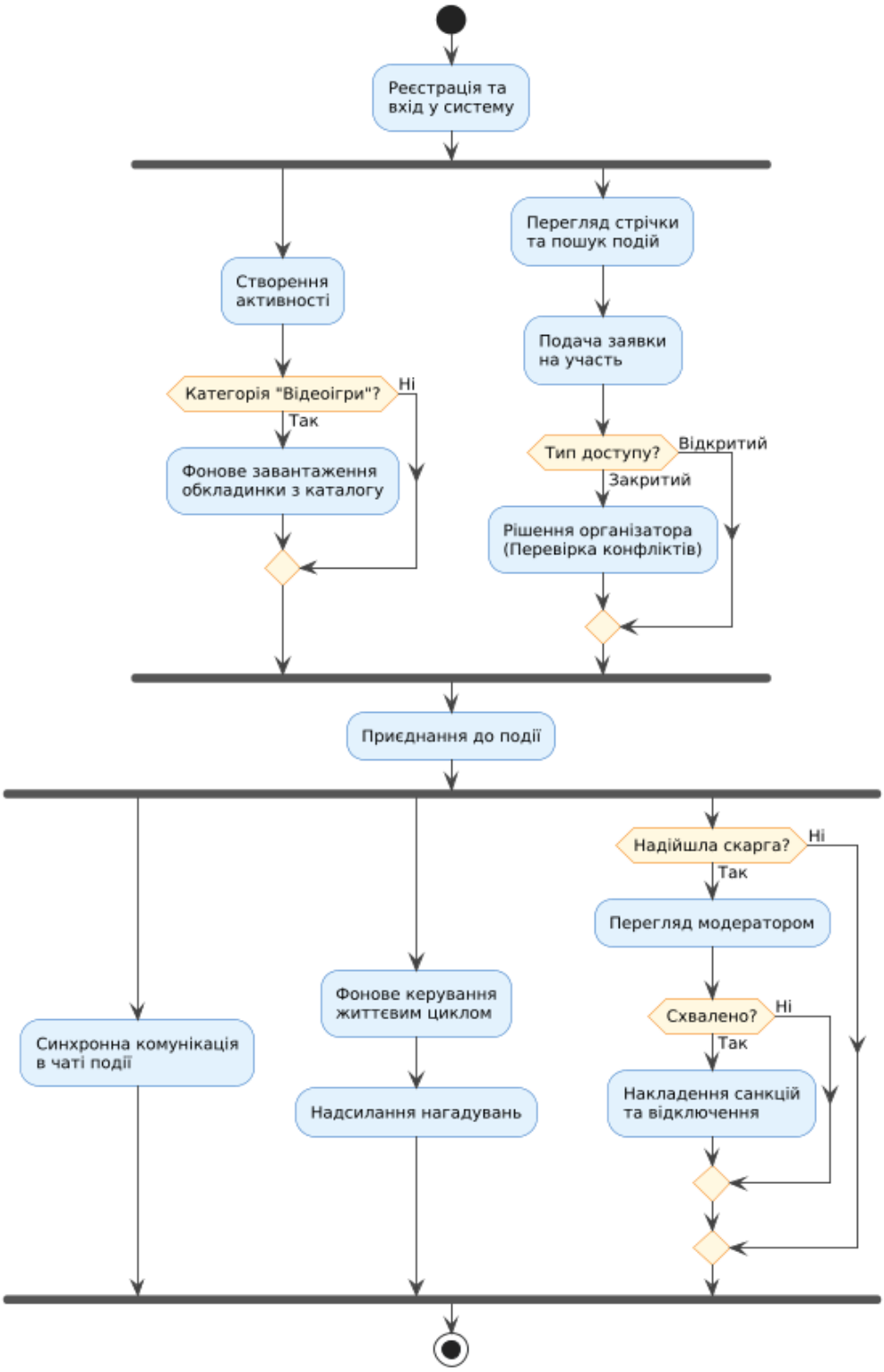

Рисунок 1.1 – Діаграма діяльності процесу організації спільного заходу

1.1.2 Опис функціональної моделі

Функціональну структуру системи визначають взаємодії між п'ятьма суб'єктами, кожен з яких має чітко окреслене коло прав та обов'язків.

До анонімного користувача належать неавторизовані відвідувачі платформи. Цей суб'єкт має доступ лише до ознайомчого функціоналу. Він може пройти процедуру реєстрації в системі або виконати вхід за наявними обліковими даними. Йому дозволено переглядати загальну стрічку активностей без можливості бачити детальну інформацію про учасників події чи приєднуватися до них. Також анонімний користувач може виконувати тестові пошукові запити до ігрової бази даних для перевірки наявності ігор.

Зареєстрований користувач володіє повним набором інструментів для взаємодії всередині платформи. Він може керувати власним профілем: змінювати ім'я, пошту, текстовий опис про себе, завантажувати нові зображення аватара чи банера та керувати списком улюблених інтересів. Цей суб'єкт має право створювати власні події будь-якої категорії, редагувати та видаляти їх. У створених ним подіях він виконує функції організатора: схвалює чи відхиляє запити на участь, вилучає учасників, доступно йому також керування контентом подій з урахуванням обмежень за поточним статусом життєвого циклу. Зареєстрований користувач має право подавати заявки на участь у подіях інших користувачів, залишати такі події за власним бажанням та користуватися пошуком ігор із використанням кешованих запитів. Йому доступний синхронний обмін повідомленнями у чатах тих подій, у яких він є підтвердженим учасником, а також персональний реєстр адресних сповіщень з можливістю фільтрації за статусом прочитання та типом події. Користувач має право подавати скарги на інших учасників спільноти, конкретні події або повідомлення в чатах, керувати тонкими налаштуваннями отримання сповіщень за визначеними категоріями та користуватися секційними переглядами власних активностей та чатів.

Модератор є привілейованим користувачем системи. Крім усіх можливостей звичайного зареєстрованого користувача, цей суб'єкт має повноваження для контролю порядку на платформі. До його основних обов'язків належать перегляд черги поданих скарг із фільтрацією за статусами обробки, взяття скарги в роботу, винесення рішень про задоволення скарги з накладенням санкцій або її відхилення як необґрунтованої. Модератор має право самостійно ініціювати накладення безстрокового або тимчасового бану на будь-якого користувача за межами процедури обробки скарги, а також знімати раніше накладені бани. Йому доступний розширений профіль будь-якого користувача з повною історією дій.

Зовнішній сервіс RAWG API є інформаційним донором. Він приймає пошукові запити від системи та повертає структуровані дані про відеоігри, включаючи офіційні назви, жанри та посилання на першоджерела зображень.

Фоновий обробник задач Celery виконує технічні операції в асинхронному режимі. Він не взаємодіє з користувачем напряму, а обробляє чергу інструкцій від сервера: завантажує зображення з мережі, виконує їх стиснення та масштабування, взаємодіє з об'єктним сховищем, оновлює відповідні статуси у базі даних, формує та надсилає адресні сповіщення користувачам, керує переходом подій між станами життєвого циклу, виконує періодичні нагадування про події, що скоро розпочнуться, автоматично знімає тимчасові бани після закінчення їх строку та контролює цілісність гібридного сховища через зіставлення посилань між реляційною та документною базами даних. Розподіл прав доступу до ресурсів системи зафіксовано в таблиці 1.1.

Таблиця 1.1 – Матриця прав доступу до функціоналу системи

Ресурс системи	Операція	Анонімний користувач	Зареєстрований користувач	Модератор
Власний профіль	Читання та редагування	Ні	Так	Так
Профілі інших користувачів	Перегляд публічних полів	Так	Так	Так
Загальна стрічка подій	Перегляд та фільтрація	Так	Так	Так

Продовження таб. 1.1

Створення нової події	Створення запису в БД	Ні	Так	Так
Власна подія	Редагування та видалення	Ні	Так	Так
Подія іншого автора	Редагування та видалення	Ні	Ні	Так
Заявки на участь у події	Подача та скасування	Ні	Так	Так
Заявки у власній події	Схвалення чи відхилення	Ні	Так	Так
Чат активності	Синхронна комунікація	Ні	Так(як учасник)	Так
Web-сповіщення	Перегляд, прочитання, видалення	Ні	Так	Так
Скарга на користувача	подію або повідомлення	Ні	Так	Так
Обробка скарг (take/resolve/dismiss)	Винесення модераторського рішення	Ні	Ні	Так
Бан та зняття бану користувача	Накладення санкцій	Ні	Ні	Так
Розширений профіль користувача	Перегляд історії санкцій та скарг	Ні	Ні	Так
Інтеграція з RAWG API	Пошук та кешування ігор	Так	Так	Так

Усі функції системи, що реалізують бізнес-логіку, розділені на окремі прецеденти використання. Для анонімного користувача визначено прецеденти реєстрації, автентифікації, перевірки працездатності серверів за допомогою запитів до службових ендпоінтів та базового перегляду стрічки. Для зареєстрованого користувача додаються прецеденти оновлення профілю, керування безпекою облікового запису, завантаження мультимедіа, створення та адміністрування активностей, взаємодії з системою заявок, синхронної комунікації в чаті активності, керування власними адресними сповіщеннями з налаштуванням преференцій, перегляду секційних дашбордів власних активностей та чатів, а також подачі скарг на порушення правил спільноти. Робота модератора виділена в окремі прецеденти адміністративного

втручання: перегляд черги скарг із фільтрацією за статусами, обробка скарги через дії take, resolve та dismiss, накладення тимчасових та безстрокових банів за межами процедури обробки скарг, зняття раніше накладених санкцій та перегляд розширеного профілю користувача з повною історією дій. Функціонування фонових обробників описується прецедентами асинхронної обробки медіа, керування життєвим циклом активностей, формування та надсилання адресних сповіщень, автоматичного зняття прострочених тимчасових банів, періодичної перевірки цілісності гібридного сховища, а також компенсуючих операцій з видалення осиротілих об'єктів у сховищі.

Діаграму варіантів використання проєктованої системи, яка відображає взаємодію основних суб’єктів із функціональними прецедентами веб-платформи, представлено на рисунку 1.2.

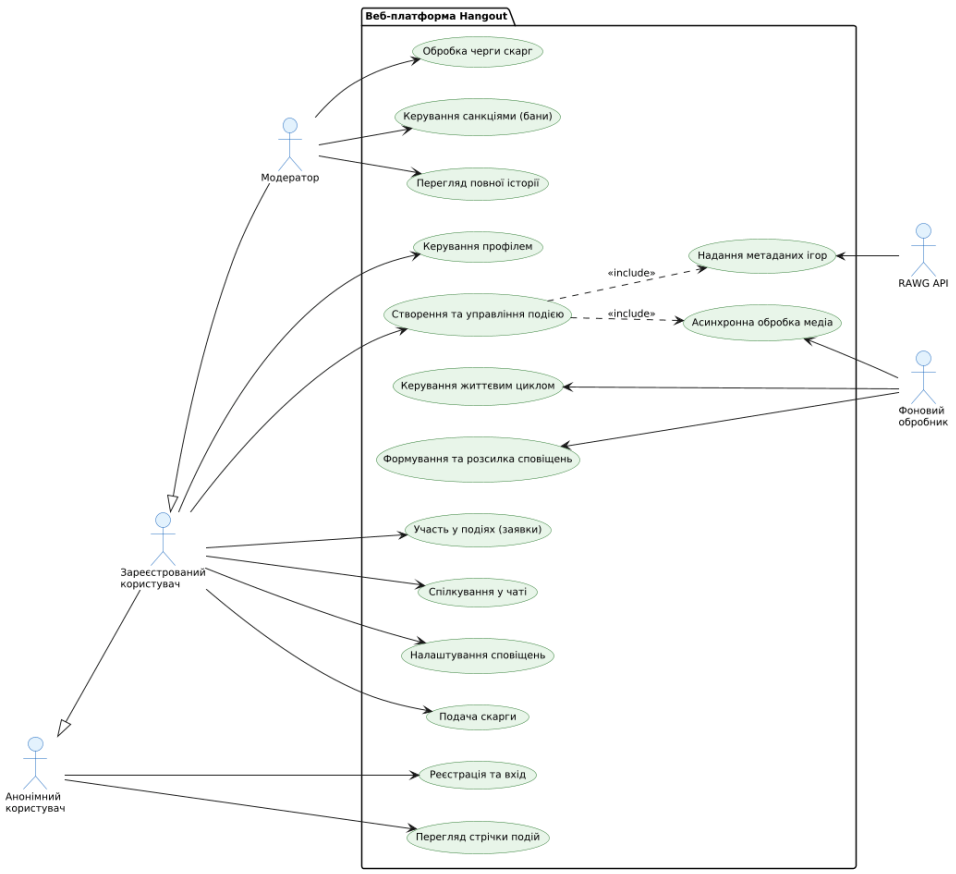

Рисунок 1.2 – Діаграма варіантів використання проєктованої системи

1.2 Огляд наявних аналогів

Для оцінки конкурентного середовища та визначення позиціонування платформи "Hangout" проаналізовано шість існуючих програмних продуктів, які виконують схожі функції на ринку організації спільних заходів та соціальної взаємодії.

Перший аналог – платформа Meetup. Це глобальний сервіс, створений для об’єднання людей у локальні спільноти за інтересами. Згідно зі статистичними даними [3], попит на довготривалі групові об’єднання залишається стабільним. Водночас сервіс більше орієнтований на організацію тематичних спільнот і регулярних зустрічей, ніж на швидкий пошук для разової події.

Другий аналог – сервіс Eventbrite. Це професійна платформа для створення, публікації та адміністрування подій різного масштабу. Згідно з аналітичним звітом [4], на ринку спостерігається зростання унікальних подій. Водночас для некомерційного сценарію використання Eventbrite може бути менш зручним через орієнтацію на інструменти продажу квитків, маркетингу та організації подій.

Третій аналог – інструменти організації подій у соціальній мережі Facebook (Facebook Groups та Facebook Events). Вони мають найбільше охоплення аудиторії та є безкоштовними. Проте алгоритмічна стрічка новин Facebook часто приховує публікації про локальні події від користувачів, які не виявляють високої активності у конкретній групі. Створені події не мають структурованих полів для валідації специфічних даних, а оголошення швидко опускаються в стрічці під впливом нових постів. Також відсутній інструмент автоматичного контролю заповненості групи.

Четвертий аналог – комунікаційна платформа Discord. Вона є популярною серед геймерів та має розвинені інструменти для текстового та голосового спілкування. Проте Discord побудований як месенджер, а не як

координатор подій. У ньому немає єдиної глобальної стрічки, де користувач міг би знайти компанію поза межами серверів, на які він уже підписаний. Поняття події в Discord обмежене простим плануванням часу без інтеграції з географічними картами чи зовнішніми каталогами ігор.

П’ятий аналог – спеціалізований сервіс OpenSports. Він розроблений виключно для організації спортивних ігор, збору команд та оренди майданчиків. Сервіс має якісні фільтри за рівнями підготовки та видами спорту, але його вузька спеціалізація не дозволяє використовувати його для інших категорій дозвілля, таких як настільні ігри, кіно чи спільний перегляд аніме.

Шостий аналог – мобільний додаток Heylo. Він створений для управління регулярними клубами (наприклад, біговими чи книжковими). Додаток поєднує чати, календар подій та збір членських внесків. Його недоліком є орієнтація на вже існуючі, сталі компланіси. Heylo не допомагає знайти нову компанію для разової випадкової активності, оскільки всі події всередині додатку закриті межами конкретного клубу.

З технічної точки зору проєкт "Hangout" будується на основі шести архітектурних переваг:

- концепція комбінованого збереження даних Polyglot Persistence, яка передбачає поєднання реляційної бази даних для збереження транзакційних даних користувачів та документо-орієнтованої бази даних для гнучких документів активностей із різним набором полів;
- використання асинхронних черг задач для взаємодії із зовнішніми каталогами та обробки медіафайлів, що усуває затримки при обробці запитів користувачів;
- організація синхронної комунікації в реальному часі на основі протоколу WebSocket з інтегрованими механізмами контролю спаму та збереження цілісності історії;
- пряме завантаження файлів до об'єктного сховища за допомогою

технології тимчасово підписаних посилань, що знижує мережеве навантаження на систему.

1.3 Постановка задачі

Головною метою роботи є проєктування та розробка серверної частини асинхронного веб-сервісу для соціальної платформи "Hangout", яка автоматизує процеси створення, пошуку та координації спільних некомерційних активностей у малих групах, забезпечує синхронну комунікацію учасників та підтримує порядок у спільноті засобами модерації.

Для досягнення поставленої мети необхідно вирішити такі завдання:

- спроєктувати комбіновану архітектуру збереження даних на базі реляційної та документо-орієнтованої СУБД;
- розробити клієнтську частину та серверну частину;
- розробити систему реєстрації, автентифікації та контролю сесій користувачів з високим рівнем захисту від несанкціонованого доступу;
- реалізувати адаптивний механізм публікації та валідації подій із гнучкою структурою даних для різних категорій дозвілля;
- розробити асинхронний модуль інтеграції з ігровим каталогом та об’єктним сховищем;
- реалізувати канали синхронної комунікації та адресних сповіщень;
- спроєктувати інструментарій модерації спільноти та контролю порядку;
- забезпечити високу продуктивність системи через оптимізацію запитів.

Функціональні вимоги до розроблюваного серверного програмного забезпечення розділені на шість груп.

FR-1: Керування профілями користувачів – визначає правила створення та контролю профілів. Система повинна забезпечувати реєстрацію користувачів з унікальними іменами та поштою, валідацію паролів, безпечний вхід із використанням механізму токенів та їх ротацію. При зміні

пароля система має автоматично анулювати всі сесії користувача. Користувач повинен мати можливість редагувати особисту інформацію, вибирати інтереси та завантажувати медіафайли профілю обмеженого розміру, контроль цілісності яких покладається на об'єктне сховище.

FR-2: Організація активностей та пошук – містить правила публікації подій за сімома категоріями. Обов’язковою є перевірка часу початку події та окремо перевіряється наявність координат для офлайн-формату. Якщо дата завершення не вказана, система розраховує її автоматично. Стрічка подій повинна віддаватися частинами з одночасним завантаженням інформації про авторів подій.

FR-3: Управління складом учасників – описує процес координації членів події. Система повинна обробляти запити на приєднання, підтримувати статус очікування для закритих подій, надавати автору можливість схвалення або відхилення кандидатів та контролювати ліміт вільного місця в групі. Також повинна проводитися перевірка часових конфліктів у розкладі при створенні подій та додаванні учасників.

FR-4: Комунікація та інтерактивні функції – описує чат активності на основі протоколу WebSocket. Сервер повинен підтримувати постійні з'єднання учасників, доставляти текстові повідомлення та системні події життєвого циклу в реальному часі, надавати можливість отримання історії повідомлень, виконувати нормалізацію вхідного тексту та контролювати частоту надсилання повідомлень із примусовим розривом з'єднання порушників.

FR-5: Підсистема сповіщень та преференцій – описує доставку нотифікацій. Сервер повинен формувати адресні повідомлення про події системи, надавати користувачу реєстр сповіщень з можливістю фільтрації за статусом прочитання, підтримувати дії масового позначення прочитаним, soft-delete-видалення, автоматично контролювати термін зберігання та виконувати дедуплікацію повторних надсилань. Користувач повинен мати можливість самостійно налаштовувати отримання необов'язкових категорій

повідомлень.

FR-6: Модерація спільноти та автоматичне керування – описує інструментарій підтримки порядку. Система повинна приймати скарги з обґрунтуванням, надавати модератору чергу скарг з фільтрацією, реалізовувати дії взяття в роботу, задоволення та відхилення скарги з накладенням санкцій, а також самостійного блокування користувачів з негайним припиненням їх сесій. Також підсистема повинна в автоматичному режимі переводити події між статусами життєвого циклу, надсилати нагадування перед початком заходу, знімати прострочені санкції та обробляти сценарії скасування подій.

Нефункціональні вимоги до розроблюваної системи визначають якісні характеристики веб-платформи та обмеження процесу її створення. Платформа будується за клієнт-серверною архітектурною моделлю, де фронтенд реалізується як Single Page Application із реактивним управлінням станом на базі сучасного компонентного підходу, а бекенд має суворий розподіл на сім ізольованих шарів (маршрутизатори, сервіси, схеми валідації, моделі даних, конфігурація, фонові задачі та утиліти) для забезпечення масштабованості кодової бази та запобігання прямим запитам до баз даних із рівня маршрутизаторів. Продуктивність та масштабованість досягаються за рахунок повного використання асинхронного виконання операцій введення-виведення на всіх рівнях взаємодії, оптимізації запитів до баз даних без використання ресурсномістких операцій зміщення, кешування результатів зовнішніх запитів, горизонтального масштабування фонових обробників задач та прямого завантаження медіафайлів без проксування через веб-сервер.

Висновки до розділу

У першому розділі виконано комплексний аналіз предметної області створення веб-платформи для пошуку однодумців та спільної організації

некомерційних заходів. На основі аналізу соціологічних досліджень обґрунтовано актуальність розробки, спрямованого на подолання соціальної ізоляції молоді шляхом залучення до спільного локального дозвілля.

Розглянуто життєвий цикл організації подій та визначено основні етапи функціонування системи. Описано процес реєстрації та автентифікації користувачів, створення подій, керування учасниками, вбудовану синхронну комунікацію, систему сповіщень, а також механізм модерації та обробки скарг. Спроєктовано функціональну модель системи, визначено ролі основних учасників взаємодії та сформовано матрицю розподілу прав доступу до ресурсів веб-сервісу.

Проведено порівняльний аналіз існуючих програмних аналогів у сфері планування заходів і соціальних комунікацій. Виявлено їхні основні недоліки для некомерційного молодіжного використання, зокрема комерційну спрямованість, орієнтацію на масові події, відсутність єдиного середовища для організації та спілкування, а також недостатню гнучкість у структуруванні подій.

За результатами аналізу сформульовано функціональні та нефункціональні вимоги до системи, обґрунтовано вибір технологічного стеку для клієнтської частини на базі Vue 3, TypeScript і Pinia, а також серверної частини на FastAPI. Описано основні інфраструктурні рішення та визначено архітектурні підходи, що забезпечують продуктивність, безпеку та надійність програмного продукту.

РОЗДІЛ 2 ІНФОРМАЦІЙНЕ ТА МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Аналіз предметної області

Розробка веб-платформи потребує чіткого визначення об'єктів дослідження та побудови інформаційної моделі. Система Hangout має п'ять основних інформаційних об'єктів:

а) користувач – містить автентифікаційні дані, налаштування профілю, історію участі у заходах та індивідуальні преференції щодо сповіщень

б) активність – центральний об’єкт системи, що об’єднує користувачів; описується набором специфічні параметрів залежно від категорії, має географічні координати для офлайн-формату та статус життєвого циклу;

в) заявка на участь – встановлює зв’язок між користувачами та активністю, фіксує момент подачі запиту та рішення організатора;

г) повідомлення чату – забезпечує синхронну комунікацію муж учасниками конкретної події, містить текстові дані або системні сповіщення;

д) скарга – фіксує факт порушення правил спільноти та ініціює модераторський процес.

Інформаційна модель платформи базується на обміні даними між клієнтським застосунком та сервером. Клієнт формує запити на створення або зміну стану об'єктів, а сервер виконує валідацію, обробляє бізнес-логіку та зберігає результати у базах даних. Особливістю моделі є наявність асинхронних фонових процесів, які самостійно генерують нові дані без прямого запиту від користувача. До таких процесів належать автоматична зміна статусів активностей за розкладом та завантаження обкладинок ігор із зовнішнього каталогу.

2.1.1 Вхідні дані

Вхідні дані системи поділяються на п'ять груп. До першої групи належать дані реєстрації та автентифікації. Користувач передає електронну пошту, унікальне ім'я та пароль. Система обмежує довжину пароля мінімум 8 символами з обов'язковою наявністю цифри та великої літери. Для підтримання сесії використовуються короткоживучі токени доступу (JWT) та довгоживучі токени оновлення.

Другу групу формують параметри профілю та мультимедійні файли. Текстова інформація профілю (біографія, інтереси) проходить перевірку на довжину та наявність заборонених символів. Файли аватарів та фонових зображень завантажуються безпосередньо від клієнта. Розмір аватара обмежується 5 мегабайтами, банера – 10 мегабайтами. Сервер приймає лише формати JPEG, PNG, GIF та WEBP. Для захисту від атак типу decompression bomb сервер застосовує обмеження максимальної кількості пікселів у зображенні на рівні 25 мільйонів.

Третя група – це дані для створення та редагування активностей. Сервер отримує структуровані JSON-об'єкти. Обов'язковими параметрами є заголовок, опис, тип доступу, формат проведення та ліміт учасників. Залежно від вибраної категорії система вимагає додаткові поля, наприклад, назву консолі для відеоігор або рівень складності для настільних ігор. Валідація цих даних відбувається за допомогою бібліотеки Pydantic. Система відхиляє запити, якщо запланований час початку події менший ніж дві години від поточного часу. Для офлайн-подій обов'язковою є наявність географічних координат.

Четвертою групою вхідних даних є повідомлення у чатах. Вони передаються через протокол WebSocket. Текстові повідомлення обмежуються довжиною до 2000 символів. Перед збереженням сервер виконує нормалізацію тексту за стандартом NFC та видаляє символи нульової ширини.

П'ята група включає дані для модерації. Користувачі передають скарги на інших учасників, події або повідомлення. Скарга обов'язково має містити обґрунтування довжиною від 10 до 1000 символів.

Існують жорсткі інфраструктурні обмеження на частоту надсилання вхідних даних (rate limiting). Механізм ковзного вікна в оперативній пам'яті обмежує спроби реєстрації до 3 запитів на хвилину з однієї IP-адреси. Запити на зміну мультимедійних файлів обмежуються 10 запитами на 5 хвилин. Для WebSocket-з'єднань діє окремий лічильник: у разі перевищення ліміту повідомлень сервер примусово розриває з'єднання з кодом помилки.

2.1.2 Вихідні дані

Вихідні дані системи надаються користувачам у форматах REST-відповідей, WebSocket-повідомлень та медіа-посилань.

Першим типом вихідних даних є результати запитів до REST API. Сервер повертає JSON-об'єкти з інформацією профілів, деталями активностей та статусами скарг. Для оптимізації передачі великих обсягів даних використовується курсорна пагінація. Під час запиту загальної стрічки активностей клієнт отримує обмежену кількість записів та курсор для завантаження наступної порції. Це дозволяє уникнути перевантаження мережі та знизити використання ресурсів бази даних на операціях зміщення.

Другим типом є тимчасово підписані посилання на медіафайли. Замість прямої передачі зображень через оперативну пам'ять веб-сервера, система віддає клієнту згенероване URL-посилання для доступу до об'єктного сховища MinIO. Це посилання має обмежений час дії, зазвичай одну годину. Такий підхід забезпечує безпеку файлів та знімає навантаження на основний сервер.

Третій тип вихідних даних – це потік інформації у синхронних чатах. Сервер використовує механізм публікації-підписки (Pub/Sub) для трансляції повідомлень усім активним учасникам події. Окрім текстових повідомлень, система автоматично генерує та надсилає системні сповіщення про зміну

складу учасників (наприклад, додавання нового користувача або його вилучення).

Четвертим типом є веб-сповіщення (notifications). Вони інформують користувачів про асинхронні події, що відбулися на платформі. Система генерує 10 різних видів сповіщень. До них належать сповіщення про схвалення чи відхилення заявки, нагадування про швидкий початок події, скасування заходу організатором та результати розгляду скарг. Окремим підвидом є сповіщення про збіг тегів, які розсилаються користувачам, коли на платформі з'являється нова подія з їхніми улюбленими інтересами. Сервер виконує дедуплікацію цих повідомлень, щоб користувач не отримав кілька однакових сповіщень. Записи про сповіщення зберігаються у базі даних і автоматично видаляються після закінчення їхнього терміну дії.

2.2 Проєктування системи

Система Hangout складається з трьох логічних компонентів: клієнтського застосунку, серверного програмного інтерфейсу (API) та підсистеми фонових обчислень. Клієнтська частина реалізована як односторінковий застосунок (SPA), що виконується у браузері користувача. Вона взаємодіє із сервером через HTTP-запити та постійні WebSocket-з'єднання. Серверна частина відповідає за автентифікацію, бізнес-логіку та управління даними. Вона має архітектуру, орієнтовану на сервіси, де кожен модуль відповідає за конкретний домен платформи. Підсистема фонових обчислень складається з черги повідомлень та пулу воркерів. Вона виконує ресурсомісткі задачі поза основним циклом обробки запитів, що забезпечує швидку відповідь сервера користувачу.

2.2.1 Проєктування бази даних

Проєктування бази даних платформи використовує підхід багатомовної

персистентності (Polyglot Persistence). Замість зберігання всіх даних в одній системі, архітектура розподіляє дані між чотирма різними типами сховищ відповідно до їхніх характеристик.

Першим сховищем є реляційна база даних PostgreSQL. Вона зберігає структуровані дані користувачів, списки тегів та налаштування профілів. Ці дані мають суворі зв'язки та вимагають повної відповідності принципам ACID (атомарність, узгодженість, ізольованість, довговічність). Використання реляційної моделі гарантує унікальність електронних адрес та цілісність посилань між таблицями.

Таблиця 2.1 – Структура таблиць реляційної бази даних PostgreSQL

Таблиця	Поле	Тип	Обмеження
users	id	UUID	PK
	username	VARCHAR	UNIQUE, NOT NULL
	email	VARCHAR	UNIQUE, NOT NULL
	hashed_password	VARCHAR	NOT NULL
	session_version	INT	NOT NULL, DEFAULT 0
	avatar_key	VARCHAR	–
	banner_key	VARCHAR	–
	bio	TEXT	–
	created_activities_count	INT	DEFAULT 0
	visited_activites_count	INT	DEFAULT 0
	Is banned	BOOLEAN	DEFAULT FALSE
tags	id	UUID	PK
	name	VARCHAR	NOT NULL
	slug	VARCHAR	UNIQUE, NOT NULL
	usage_count	INT	DEFAULT 0
user_tags	user_id	UUID	PK, FK → users
	tag_id	UUID	PK, FK → tags
notification_preferences	user_id	UUID	PK, FK → users
	membership_updates	BOOLEAN	DEFAULT TRUE
	activity_reminders	BOOLEAN	DEFAULT TRUE
	tag_subscriptions	BOOLEAN	DEFAULT TRUE
	report_updates	BOOLEAN	DEFAULT TRUE

Таблиця users зберігає облікові записи. Поле session_version інкрементується при зміні пароля або бані, що інвалідовує всі видані раніше токени. Таблиця tags містить нормалізовані мітки інтересів, а зв'язувальна таблиця user_tags реалізує відношення «багато до багатьох» між користувачами та тегами. Таблиця notification_preferences встановлює відношення «один до одного» з users та дозволяє кожному користувачу індивідуально вмикати або вимикати категорії сповіщень.

Другим сховищем є документо-орієнтована база даних MongoDB. Вона зберігає активності, заявки на участь, повідомлення чатів та скарги. Різні категорії активностей мають різні набори атрибутів (наприклад, платформа для відеоігор відрізняється від рівня складності настільних ігор). Документна модель дозволяє ефективно зберігати такі динамічні структури без необхідності створювати розріджені таблиці чи складні нормалізовані схеми. Окрім того, MongoDB забезпечує високу швидкість запису, що є критичним для збереження великої кількості повідомлень у чатах. Структуру колекцій MongoDB наведено у табл. 2.2.

Таблиця 2.2 – Структура колекцій документо-орієнтованої бази даних MongoDB

Таблиця	Поле	Тип	Призначення
activities	_id	ObjectId	Ідентифікатор документа
	title	String	Назва активності
	type	String	open / closed
	format	String	online / offline
	category	String	Одна з 7 категорій
	extra_data	Object	Дискримінований набір атрибутів категорії
	date, end_date	DateTime	Часові межі події
	max_members	Int	Ліміт учасників (2–20)
	current_members	Int	Поточна кількість (атомарний $inc)
	status	INT	active / started / ended / archived / canceled
	creator_id	UUID	Soft FK → users (PostgreSQL)
membership	_id	ObjectId	Ідентифікатор документа
	activity_id	ObjectId	Soft FK → activities

Продовження таб. 2.2

	user_id	UUID	Soft FK → users (PostgreSQL)
	status	String	pending / approved / rejected / left / kicked
chat_messages	_id	ObjectId	Ідентифікатор документа
	activity_id	ObjectId	Soft FK → activities
	user_id	UUID	Soft FK → users (PostgreSQL)
	content	String	Текст повідомлення (NFC, до 2000 символів)
	message_type	String	text / system
reports	_id	ObjectId	Ідентифікатор документа
	reporter_id	UUID	Soft FK → users (автор скарги)
	reported_user_id	UUID	Soft FK → users (порушник)
	reason	String	Обґрунтування (10–1000 символів)
	status	String	open / in_review / resolved / dismissed

Колекція activities використовує поле extra_data як дискримінований об'єкт. Залежно від значення category, це поле містить різний набір атрибутів. Колекція membership має частковий унікальний індекс за парою (activity_id, user_id) для активних статусів, що запобігає подвійному приєднанню.

Третім сховищем є система кешування в оперативній пам'яті Redis. Вона виконує роль ефемерного сховища. Redis керує сесіями автентифікації, зберігаючи ідентифікатори дійсних токенів оновлення. Також ця система зберігає лічильники для механізмів обмеження частоти запитів та виконує дедуплікацію веб-сповіщень. Оскільки Redis працює виключно в оперативній пам'яті, він гарантує мінімальні затримки при читанні та записі.

Четвертим сховищем є об'єктне сховище MinIO. Воно відповідає за збереження мультимедійних файлів (аватарів, фонових зображень профілів та обкладинок ігор). Файли зберігаються у вигляді бінарних об'єктів (BLOB), доступ до яких надається через тимчасові підписані посилання. Цей підхід звільняє основні бази даних від навантаження великими масивами неструктурованих даних.

Для підтримки цілісності даних між реляційною та документною базами

використовується механізм м'яких зовнішніх ключів (soft foreign keys). Документи в MongoDB зберігають ідентифікатори користувачів з PostgreSQL. Забезпечення узгодженості цих посилань контролюється на рівні бізнес-логіки серверного застосунку та спеціальними періодичними задачами перевірки. Загальну ER-діаграму, що демонструє зв'язки між сутностями обох баз даних, наведено на рис. 2.1

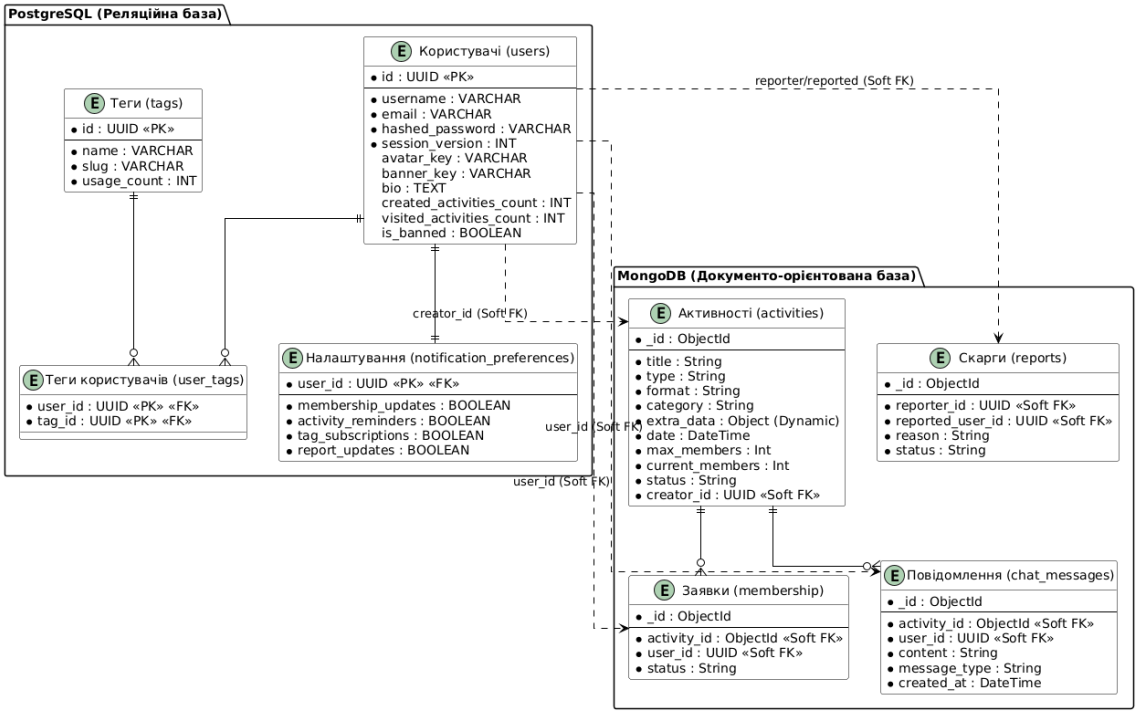

Рисунок 2.1 – ER-діаграма бази даних платформи Hangout із зв'язками між реляційним (PostgreSQL) та документним (MongoDB) сховищами

2.2.2 Побудова об’єктно-орієнтованої моделі

Серверна частина платформи побудована за тришаровою архітектурою. Кожен шар має чітку зону відповідальності та взаємодіє лише з сусіднім шаром. Такий поділ спрощує тестування, оскільки кожен модуль можна перевіряти ізольовано, та полегшує подальше розширення функціоналу без зміни суміжних компонентів [5].

Першим шаром є шар маршрутизації (Routers). Він приймає HTTP-запити та WebSocket-з'єднання від клієнта, виконує попередню перевірку прав

доступу через механізм впровадження залежностей (Dependency Injection) та передає дані у шар бізнес-логіки. Серверний застосунок має 11 модулів маршрутизації, кожен з яких обслуговує окремий домен платформи: автентифікацію, профілі, активності, членство, чат, ігри, сповіщення, скарги, користувачів, модерацію та перевірку стану сервера.

Другим шаром є шар сервісів (Services). Він містить всю бізнес-логіку платформи. Кожен сервіс інкапсулює правила конкретного домену та оперує об'єктами, описаними у підрозділі 2.1. Сервіси не залежать від протоколу передачі даних (HTTP чи WebSocket) і можуть викликатися як з маршрутизаторів, так і з фонових задач. Перелік основних сервісів та їхні функції наведено у табл. 2.3.

Таблиця 2.3 – Перелік сервісів серверного застосунку

Сервіс	Функціональне призначення
AuthService	Реєстрація, автентифікація, ротація токенів оновлення, інвалідація сесій
ProfileService	Редагування профілю, завантаження медіа файлів у об'єктне сховище з компенсуючими операціями
ActivityService	Створення та редагування активностей, курсорна пагінація стрічки, валідація за категоріями
ActivityConflictService	Перевірка перетину розкладів подій при приєднанні або схваленні заявки
MembershipService	Подача заявок, схвалення, відхилення, вилучення учасників з атомарною зміною лічильника
ChatService	Збереження та видача історії повідомлень, нормалізація тексту, валідація довжини
GameSearchService	Проксі-запити до зовнішнього каталогу ігор RAWG з кешуванням результатів
WebNotificationService	Генерація, читання та автоматичне видалення веб-сповіщень з дедуплікацією
ReportService	Створення скарг, модераторських дії (взяти, вирішити, відхилити)
BanService	Тимчасові та постійні бани з інвалідацією сесій, ведення історії
TagService	Нормалізація тегів, підрахунок використання, перемикання обраних тегів

Третім шаром є шар інфраструктури (Core). Він надає інтерфейси до зовнішніх систем: баз даних, кешу, об'єктного сховища та брокера

повідомлень. Цей шар містить модулі конфігурації (читання змінних середовища), з'єднання з базами даних, криптографічних операцій (підпис та верифікація JWT, хешування паролів), обмеження частоти запитів та управління WebSocket-з'єднаннями. Окремим компонентом інфраструктурного шару є менеджер WebSocket-з'єднань. Він реалізує патерн "публікація-підписка": кожна активність має власний канал, і всі підключені учасники автоматично отримують нові повідомлення та системні події.

Паралельно з основним серверним процесом працює підсистема фонових задач. Вона складається з двох компонентів: планувальника (Beat), який запускає періодичні завдання за розкладом, та пулу виконавців (Workers), які обробляють як періодичні, так і одноразові задачі. Фонові задачі використовують ті самі сервіси та інфраструктурні модулі, що й основний сервер, але працюють у окремих процесах. Структуру взаємодії шарів серверної частини наведено на рис. 2.2.

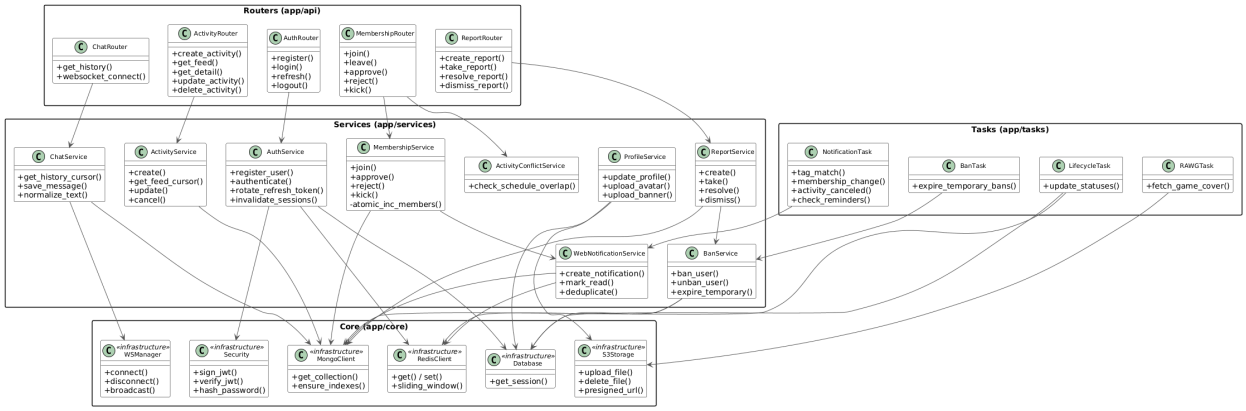

Рисунок 2.2 – UML-діаграма класів серверної частини платформи Hangout

Клієнтська частина побудована як односторінковий застосунок на основі фреймворку Vue 3. Архітектура клієнта також має шарову організацію, хоча вона відрізняється від серверної.

Шар стану (Stores) відповідає за централізоване управління даними на стороні клієнта. Кожен стор зберігає фрагмент глобального стану застосунку, надає методи для його зміни та виконує запити до серверного API. Платформа

має 8 сторів. Перелік сторів та їхні функції наведено у табл. 2.4.

Таблиця 2.4 – Перелік сторів клієнтського застосунку

Стор	Функціональне призначення
auth	Токен доступу, поточний користувач, ініціалізація сесії, керування модалками автентифікації
activity	Стрічка активностей з курсорною пагінацією, фільтри, деталь окремої активності
chat	Активний чат, історія повідомлень, стан WebSocket-з'єднання, маркери непрочитаних
profile	Дані власного профілю, згенеровані посилання на аватар та банер
notifications	Список сповіщень, лічильник непрочитаних, періодичне опитування сервера
moderationReports	Список скарг для модератора, фільтрація за статусом
moderationUsers	Розширений профіль користувача для модератора, історія банів
toasts	Черга тимчасових повідомлень для відображення у інтерфейсі

Шар композитних функцій (Composables) містить багаторазову логіку, яка не прив'язана до конкретного компонента. Найвагомішою з них є функція управління WebSocket-з'єднанням чату. Вона інкапсулює логіку підключення, автоматичного перепідключення з експоненційною затримкою (до 5 спроб) та обробки специфічних кодів розриву з'єднання. Інші композитні функції забезпечують клієнтську перевірку перетину розкладу ще до відправлення запиту на сервер та реалізують доступні модалки підтвердження дій.

Шар API-клієнта виконує роль тонкої обгортки над HTTP-запитами. Він містить 11 модулів, кожен з яких відповідає одному домену серверного API. Центральним елементом є перехоплювач (interceptor), який автоматично оновлює протермінований токен доступу. При отриманні відповіді з кодом 401 перехоплювач виконує один запит на оновлення токена, а всі інші запити, що чекають у цей момент, ставляться у чергу та виконуються після успішного оновлення.

Шар візуальних компонентів (Components) містить окремі елементи інтерфейсу. Компоненти організовані за доменами: активності, чат, модерація, сповіщення та загальні елементи (аватар, кнопки, індикатори). Кожен компонент отримує дані через передані параметри (props) та генерує події

(emits) для зворотного зв'язку з батьківськими елементами. Загальну архітектуру клієнтської частини наведено на рис. 2.3.

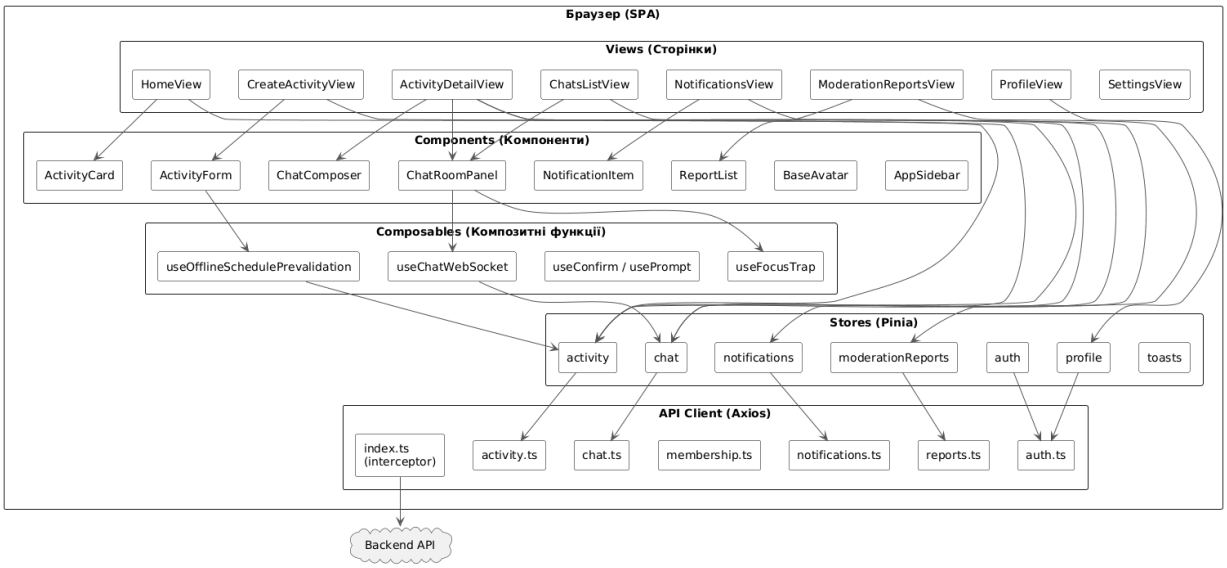

Рисунок 2.3 – Діаграма компонентів клієнтської частини платформи Hangout

2.3 Математичне та алгоритмічне забезпечення

Платформа Hangout використовує декілька алгоритмів для забезпечення коректної роботи бізнес-логіки. Цей підрозділ описує чотири основні алгоритми: управління життєвим циклом активностей, перевірку перетину розкладів, обмеження частоти запитів та компенсуючі транзакції при завантаженні файлів.

2.3.1 Алгоритм управління життєвим циклом активності.

Кожна активність у системі проходить послідовність станів, яка моделюється скінченним автоматом (Finite State Machine). Автомат має п’ять станів: active, started, ended, archived, canceled. Переходи між станами відбуваються за двома механізмами: явним (за дією користувача) та автоматичним (за розкладом фонової задачі).

Початковим станом кожної новоствореної активності є active. Це означає, що подія відкрита для приєднання. Коли поточний час досягає

запланованої дати початку, фонова задача переводить активність у стан started. Після закінчення запланованого часу (поле end_date) стан змінюється на ended. Через 24 години після завершення активність автоматично переходить у стан archived. На цьому етапі система також інкрементує лічильник відвіданих активностей для кожного учасника зі статусом approved.

Паралельно існує явний перехід: організатор може скасувати активність у будь-який момент до її завершення. У цьому випадку стан змінюється на canceled, а система надсилає обов'язкове сповіщення всім учасникам.

Фонова задача виконується кожні 5 хвилин. Алгоритм її роботи складається з трьох послідовних кроків. На першому кроці задача вибирає з бази даних усі документи зі статусом active, у яких поле date менше або дорівнює поточному часу, та оновлює їхній статус на started. На другому кроці задача знаходить усі документи зі статусом started, у яких поле end_date менше або дорівнює поточному часу, та оновлює їхній статус на ended. На третьому кроці задача знаходить усі документи зі статусом ended, у яких різниця між поточним часом та end_date перевищує 24 години, оновлює їхній статус на archived та інкрементує лічильник visited_activities_count для відповідних користувачів у реляційній базі. Діаграму станів наведено на рис. 2.4.

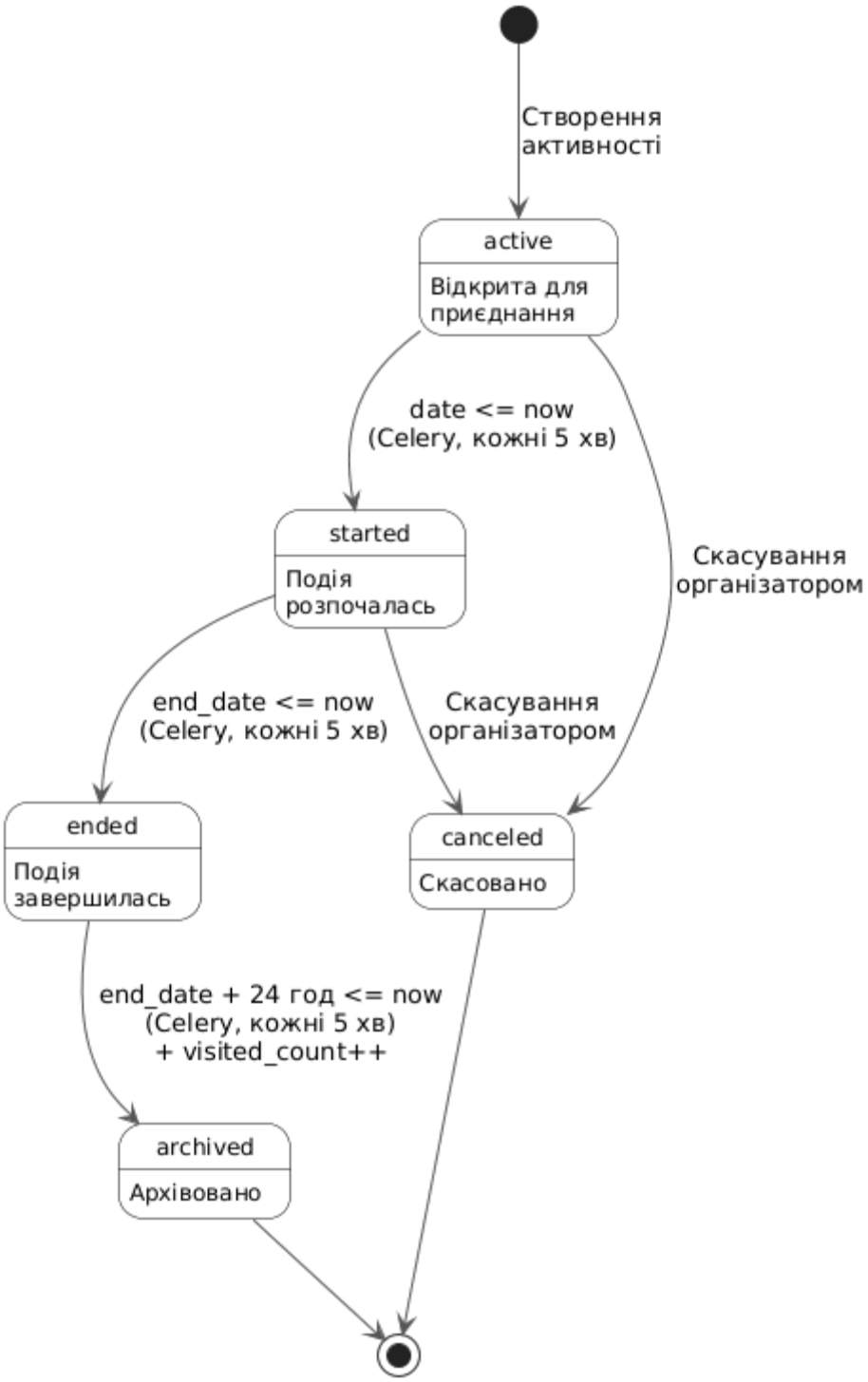

Рисунок 2.4 – Діаграма станів життєвого циклу активності

2.3.2 Алгоритм перевірки перетину розкладів.

Для подій система забороняє одному користувачу брати участь у двох заходах, які відбуваються одночасно. Перевірка виконується при трьох

операціях: подачі заявки на участь, схваленні заявки організатором та редагуванні часу існуючої активності.

Алгоритм працює з часовими відрізками. Кожна активність визначає відрізок [date, end_date]. Два відрізки [A_start, A_end] та [B_start, B_end] перетинаються тоді і тільки тоді, коли виконується умова:

$$A_{start} < B_{end} \wedge B_{start} < A_{end} \quad (2.1)$$

де A_start – Дата та година початку першої активності; A_end – Дата та година закінчення першої активності; B_start – Дата та година початку другої активності; B_end – Дата та година закінчення другої активності.

Ця формула є класичною умовою перетину інтервалів на числовій осі. Сервіс виконує запит до бази даних, який повертає всі активності користувача зі статусами active або started. Для кожної знайденої активності сервіс перевіряє виконання умови (2.1). Якщо хоча б один перетин знайдено, операція відхиляється з повідомленням про конфлікт розкладу.

2.3.3 Алгоритм обмеження частоти запитів.

Система використовує алгоритм ковзного вікна (Sliding Window) для обмеження кількості запитів від одного клієнта за фіксований період часу [6]. На відміну від алгоритму фіксованого вікна, ковзне вікно не має проблеми "сплеску на межі": воно враховує запити безперервно, а не скидає лічильник на початку кожного інтервалу.

Алгоритм реалізований на базі структури даних "відсортована множина" (Sorted Set) у Redis. Кожен запит додає елемент із поточною міткою часу як оцінкою (score). Для кожної пари «клієнт + ендпоїнт» формується унікальний ключ:

$$K = \text{"rate\_limit:"} + client\_id + \text{":"} + endpoint \quad (2.2)$$

де K – ключ відсортованої множини у Redis; rate_limit – службовий префікс, що використовується для групування ключів механізму обмеження запитів; client_id – ідентифікатор клієнта (user_id або IP-адреса); «:» – роздільник між складовими ключа; endpoint – назва ресурсу, що лімітується.

Нижня межа поточного вікна спостереження визначається як:

$$W = T - t \quad (2.3)$$

де T – мітка часу поточного запиту; t – розмір вікна у секундах.

Усі елементи множини K з оцінкою меншою за W видаляються як застарілі. Після очищення підраховується кількість елементів N, що залишились, і перевіряється умова:

$$N \geq L \quad (2.4)$$

де L – встановлений ліміт запитів у вікні. Якщо умова (2.4) виконується, запит відхиляється з кодом відповіді 429 (Too Many Requests); інакше новий елемент з оцінкою T додається до множини K.

Усі операції виконуються атомарно через конвеєр команд Redis, що гарантує коректність при паралельних запитах.

Для WebSocket-чату діє окремий механізм. Він підраховує кількість порушень ліміту і при досягненні порогового значення примусово розриває з'єднання з кодом 4008. Схему роботи алгоритму обмеження частоти запитів наведено на рис. 2.5.

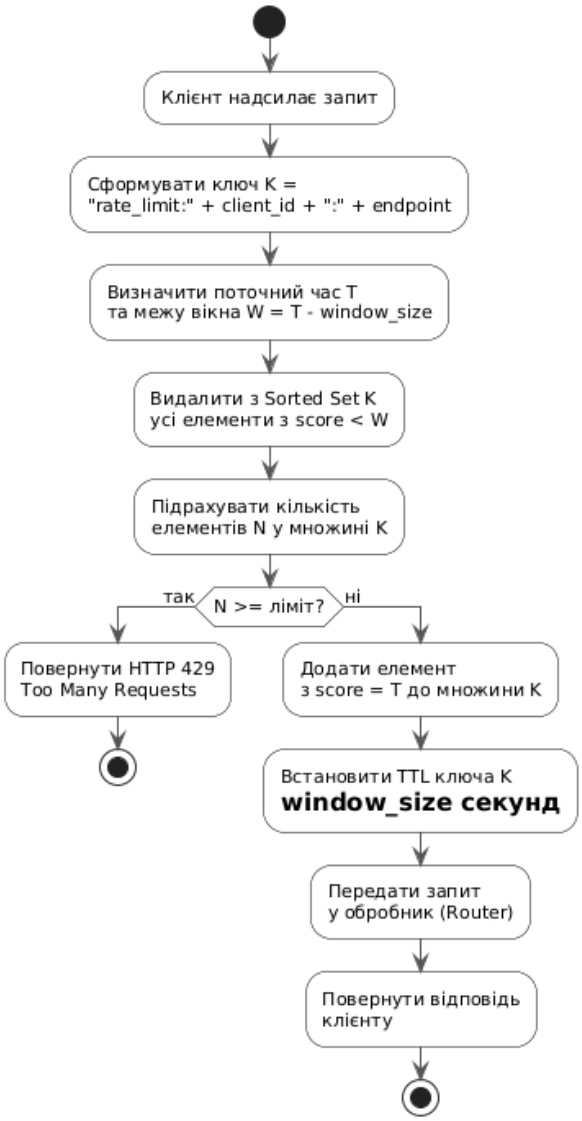

Рисунок 2.5 – Блок-схема алгоритму обмеження частоти запитів (Sliding Window)

2.3.4 Алгоритм компенсуючих транзакцій при завантаженні файлів.

Завантаження мультимедійних файлів (аватарів, банерів) потребує координації між двома сховищами: об'єктним сховищем MinIO та реляційною базою PostgreSQL. Проблема полягає в тому, що ці дві системи не підтримують розподілені транзакції, тому необхідний механізм компенсації у разі часткової помилки.

Алгоритм складається з п'яти кроків. На першому кроці сервер декодує завантажений файл через бібліотеку обробки зображень, перевіряючи формат

та розмір. На другому кроці файл зберігається в об'єктне сховище під унікальним ключем. На третьому кроці сервер оновлює запис користувача в реляційній базі, замінюючи старий ключ файлу на новий. На четвертому кроці, якщо оновлення бази пройшло успішно, сервер видаляє старий файл з об'єктного сховища. На п'ятому кроці, якщо оновлення бази завершилось помилкою, сервер запускає фонову компенсуючу задачу, яка видаляє щойно завантажений файл з об'єктного сховища, запобігаючи появі осиротілих об'єктів.

Такий підхід гарантує, що у стабільному стані кожен запис у базі даних відповідає рівно одному файлу в об'єктному сховищі. Навіть при збоях на будь-якому з кроків система повертається до узгодженого стану через компенсуючу операцію.

Висновки до розділу

У цьому розділі виконано аналіз предметної області та спроєктовано інформаційну систему платформи Hangout. Визначено основні об’єкти системи та побудовано інформаційну модуль їхньої взаємодії. Описано вхідні та вихідні дані, а також сформульовано основні обмеження їхньої обробки.

Спроєктовано базу даних а також побудовано об’єктно-орієнтовану модель системи. Обґрунтовано використання багатомовної персистентності з поєднанням реляційного, документоорієнтованого, кешуючого та об’єктного сховищ. Розроблено ER-модель і описано архітектуру серверної та клієнтської частин системи.

Також розроблено основні алгоритми, що забезпечують функціонування платформи. До них належать алгоритм керування життєвим циклом активності, алгоритм перевірки перетину розкладів, алгоритм обмеження частоти запитів та алгоритм узгодження під час завантаження файлів. У сукупності це створює основу для подальшої програмної реалізації платформи Hangout.

РОЗДІЛ 3 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Засоби розробки

Для створення платформи Hangout було обрано набір технологій, який відповідає вимогам до асинхронної обробки запитів, роботи з кількома типами баз даних та побудови інтерактивного інтерфейсу в реальному часі. Розробка серверної та клієнтської частини виконувалася у середовищі PyCharm, що забезпечує зручні засоби для написання, налагодження та тестування проєкту Серверну частину реалізовано мовою Python версії 3.13 з використанням фреймворку FastAPI. Вибір FastAPI зумовлений його вбудованою підтримкою асинхронних операцій, автоматичною генерацією OpenAPI-документації та нативною підтримкою WebSocket-з'єднань [7]. Перелік засобів розробки серверної частини наведено у табл. 3.1.

Таблиця 3.1 – Засоби розробки серверної частини

Технологія	Версія	Призначення у проєкті
Python	3.13	Мова програмування серверної частини
FastAPI	0.136	Веб-фреймворк з підтримкою async та WebSocket
Uvicorn	0.49	ASGI-сервер для запуску FastAPI
SQLAlchemy	2.0	ORM для роботи з PostgreSQL
Alembic	1.18	Управління міграціями реляційної бази
PyMongo	4.17	Асинхронний драйвер MongoDB
Celery	5.6	Черга фонових задач
RabbitMQ	3	Брокер повідомлень для Celery
Pydantic	2.13	Валідація вхідних даних та серіалізація
PyJWT	2.13	Створення та верифікація JWT-токенів
Pillow	12.2	Валідація та обробка зображень
boto3	1.43	SDK для роботи з об'єктним сховищем MinIO

Клієнтську частину реалізовано як односторінковий застосунок на базі фреймворку Vue 3 з використанням Composition API та мови TypeScript. Vue 3 обрано завдяки реактивній системі, яка ефективно оновлює DOM при зміні стану, та модульній архітектурі з підтримкою композитних функцій. Перелік засобів розробки клієнтської частини наведено у табл. 3.2.

Таблиця 3.2 – Засоби розробки клієнтської частини

Технологія	Версія	Призначення у проєкті
Vue 3	3.5	Фреймворк для побудови користувацького інтерфейсу
TypeScript	5.9	Засіб статичної типізації клієнтського коду
Pinia	3.0	Централізоване управління станом застосунку
VueRouter	5.0	Клієнтська маршрутизація та керування доступом до маршрутів
Axios	1.13	HTTP-клієнт для взаємодії з серверним API
Vite	8.0	Засіб збирання проєкту та сервер розробки
Zod	3.25	Опис схем валідації даних
Vee-validate	4.15	Інтеграція валідації з формами Vue

Інфраструктурні компоненти забезпечують розгортання, збереження даних та міжсервісну комунікацію. Перелік інфраструктурних компонентів наведено у табл. 3.3.

Таблиця 3.3 – Інфраструктурні компоненти

Компонент	Версія	Призначення у проєкті
PostgreSQL	16	Реляційна база даних для користувачів та тегів [8]
MongoDB	7	Документна база для активностей, чатів, скарг [9]
Redis	7	Кеш сесій, rate-limit лічильники, дедуплікація
MinIO	latest	S3-сумісне об'єктне сховище для медіафайлів
Docker	–	Контейнеризація та оркестрація всіх сервісів

3.2 Вимоги до технічного та програмного забезпечення

Серверна частина платформи розгортається у контейнеризованому середовищі Docker. Для коректної роботи системи необхідне серверне оточення, яке забезпечує виконання контейнерів баз даних, брокера повідомлень, об’єктного сховища, серверного застосунку, фонових процесів та клієнтської частини. Мінімальні вимогу до серверного оточення наведено у табл. 3.4.

Таблиця 3.4 – Вимоги до серверного оточення

Параметр	Мінімальне значення
Процесор	2 ядра 2.0 ГГц
Оперативна пам’ять	4 ГБ
Дисковий простір	20 ГБ (Рекомендовано SSD)

Операційна система	Windows 10+, macOS 12+, або сучасний Linux-дістрибутив
Програмне забезпечення	Docker Engine 24+, Docker Compose v2
Мережа	Стабільне підключення до мережі Інтернет

Клієнтська частина виконується у браузері користувача та не потребує встановлення додаткового програмного забезпечення. Для коректної роботи інтерфейсу необхідна підтримка JavaScript, WebSocket та сучасних вебстандартів. Мінімальні вимоги до клієнтського оточення наведено у табл. 3.5.

Таблиця 3.5 – Вимоги до клієнтського оточення

Параметр	Мінімальне значення
Браузер	Chrome 100+, Firefox 100+, Safari 15+, Edge 100+
Роздільна здатність екрану	Не менше 1366x768
Підтримка JavaScript	Обов’язкова
Підтримка WebSocket	Обов’язкова
Операційна система	Будь яка ОС, що підтримує сучасні браузери
Мережа	Стабільне підключення до мережі Інтернет

3.3 Опис програмної реалізації

3.3.1 Структура серверного застосунку.

Серверний застосунок організовано відповідно до тришарової архітектури, описаної у п. 2.2.2. Каталог api/ містить 11 модулів маршрутизації. Каталог services/ містить класи бізнес-логіки (табл. 2.3). Каталог core/ містить інфраструктурні модулі: з'єднання з базами даних, криптографію, обмеження частоти запитів та менеджер WebSocket-з'єднань. Каталог tasks/ містить Celery-задачі. Каталог models/ містить SQLAlchemy-моделі для PostgreSQL, а каталог schemas/ – Pydantic-схеми валідації.

Точкою входу є файл main.py, який створює FastAPI-застосунок із асинхронним контекстом життєвого циклу (lifespan). При старті застосунок послідовно перевіряє наявність S3-бакета, доступність Redis, ініціалізує з'єднання з MongoDB та створює необхідні індекси. Кожен ресурс

закривається через AsyncExitStack лише якщо був успішно відкритий. Загальну діаграму розгортання системи наведено на рис. 3.1.

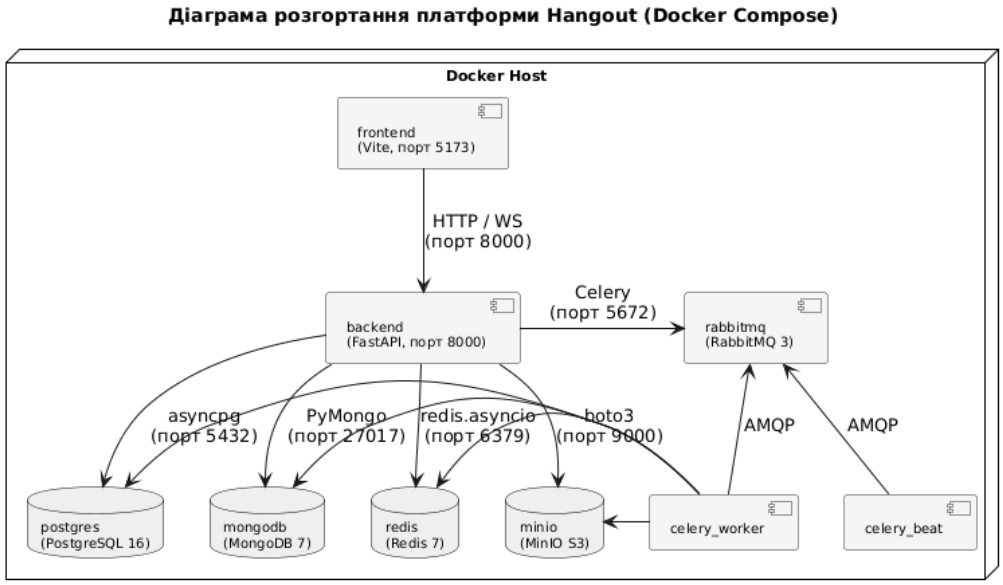

Рисунок 3.1 – Діаграма розгортання системи у середовищі Docker Compose

3.3.2 Реалізація автентифікації та управління сесіями.

Автентифікація реалізована за схемою JWT з двома типами токенів [11]. Токен доступу (access token) підписується алгоритмом HS256 та має час життя 30 хвилин. Він передається у заголовку Authorization кожного HTTP-запиту. Токен оновлення (refresh token) зберігається у httpOnly cookie з прапорцями secure та samesite=lax і має час життя 30 днів. Фрагмент коду створення токенів наведено у лістингу 3.1.

Лістинг 3.1 – Створення access та refresh токенів

```
@staticmethod
def create_access_token(user_id: int, session_version: int = 0) -> str:
    utc_now = datetime.now(timezone.utc)
    payload = {
        "sub": str(user_id),
        "type": "access",
        "ver": int(session_version or 0),
        "exp": utc_now + timedelta(minutes=settings.ACCESS_TTL_MINUTES),
        "iat": utc_now,
    }
    return jwt.encode(
        payload, settings.SECRET_KEY.get_secret_value(), algorithm="HS256"
    )
```

```
@staticmethod
def create_refresh_token(user_id: int, session_version: int = 0) ->
tuple[str, str]:
    utc_now = datetime.now(timezone.utc)
    jti = uuid4().hex
    payload = {
        "sub": str(user_id),
        "exp": utc_now + timedelta(days=settings.REFRESH_TTL_DAYS),
        "jti": jti,
        "type": "refresh",
        "ver": int(session_version or 0),
        "iat": utc_now,
    }
    return jwt.encode(
        payload, settings.SECRET_KEY.get_secret_value(), algorithm="HS256"
    ), jti
```

Кожен refresh-токен має унікальний ідентифікатор jti (JWT ID). При оновленні сесії сервер виконує ротацію: старий jti видаляється з Redis, а новий зберігається. Це гарантує, що кожен refresh-токен може бути використаний лише один раз. Фрагмент коду збереження та споживання сесій наведено у лістингу 3.2.

Лістинг 3.2 – Управління refresh-сесіями через Redis

```
@staticmethod
async def store_refresh_session(
    redis: Redis, jti: str, user_id: int, ttl_seconds: int
) -> None:
    key = AuthService._refresh_key(jti)
    user_session_key = f"user_sessions:{user_id}"
    async with redis.pipeline() as pipe:
        pipe.set(key, str(user_id), ex=ttl_seconds)
        pipe.sadd(user_session_key, jti)
        pipe.expire(user_session_key, ttl_seconds)
        await pipe.execute()

@staticmethod
async def consume_refresh_session(redis: Redis, jti: str) -> bool:
    key = AuthService._refresh_key(jti)
    user_id_str = await redis.getdel(key)
    if user_id_str is None:
        return False
    if isinstance(user_id_str, bytes):
        user_id_str = user_id_str.decode("utf-8")
    await redis.srem(f"user_sessions:{user_id_str}", jti)
    return True
```

Поле session_version у таблиці users інкрементується при зміні пароля або бані. Під час верифікації токена сервер порівнює значення ver у payload з поточним session_version користувача. Якщо значення не збігаються, токен вважається недійсним. Цей механізм дозволяє миттєво інвалідовувати всі видані токени без необхідності зберігати чорний список [12].

3.3.3 Реалізація модуля активностей

Активності є центральною сутністю системи. Валідація вхідних даних виконується через Pydantic-схеми з дискримінованим об'єднанням (discriminated union) для поля extra_data. Кожна з семи категорій має власний клас деталей із полем category типу Literal, що дозволяє Pydantic автоматично обирати потрібну схему. Фрагмент коду наведено у лістингу 3.3.

Лістинг 3.3 – Дискримінована схема активності

```
class GameDetails(BaseModel):
    category: Literal[ActivityCategory.games] = ActivityCategory.games
    game_name: str = Field(min_length=1, max_length=120)
    platform: GamePlatform
    genre: str | None = Field(default=None, min_length=1, max_length=60)
    game_id: int | None = None
    cover_key: str | None = Field(default=None, max_length=255)
    cover_status: CoverStatus = CoverStatus.pending

class BoardGameDetails(BaseModel):
    category: Literal[ActivityCategory.board_games] =
ActivityCategory.board_games
    game_name: str = Field(min_length=1, max_length=120)
    player_count: int | None = None
    complexity: BoardGamesComplexity | None = None

class ActivityBase(BaseModel):
    title: str = Field(min_length=3, max_length=100)
    type: ActivityType = Field(ActivityType.open)
    format: ActivityFormat = Field(ActivityFormat.online)
    category: ActivityCategory
    extra_data: Annotated[
        GameDetails | BoardGameDetails | MovieDetails
        | AnimeDetails | SportDetails | MusicDetails | None,
        Field(discriminator="category"),
    ]
    description: str = Field(min_length=10, max_length=2000)
    date: datetime = Field(...)
```

Такий підхід забезпечує автоматичну валідацію на рівні типів: якщо користувач обирає категорію games, Pydantic вимагає заповнення полів game_name та platform. Для інших категорій ці поля не потрібні. Курсорна пагінація стрічки активностей реалізована через ObjectId, що забезпечує стабільність при одночасних вставках нових записів.

3.3.4 Реалізація WebSocket-чату

Чат реалізовано через нативний WebSocket-протокол. Менеджер з'єднань зберігає реєстр активних сокетів, організований за кімнатами (activity_id) та користувачами. При підключенні учасника сервер додає сокет у реєстр, генерує системне повідомлення про приєднання та транслює оновлений лічильник онлайн-учасників. Фрагмент коду обробки повідомлень із перевіркою rate-limit наведено у лістингу 3.4.

Лістинг 3.4 – Обробка повідомлень у WebSocket-чаті

```
if not await chat_rate_limiter.check_rate_limit(activity_id, user.id):
    violations = await chat_rate_limiter.get_violation_count(
        activity_id, user.id
    )
    if violations >= MAX_VIOLATIONS_BEFORE_DISCONNECT:
        await websocket.close(code=4008)
        break
    else:
        error_envelope = WebSocketEnvelope(
            event=WebSocketEvent.error,
            data=WebSocketErrorData(
                detail="Rate limit exceeded. Please wait."
            ),
        )
        await websocket.send_text(error_envelope.model_dump_json())
        continue

saved_msg = await ChatService.save_message(
    user_id=user.id,
    activity_id=activity_id,
    content=content,
    chat_col=chat_col,
    db=db,
    s3_public_sign=s3_public_sign,
)
msg_envelope = WebSocketEnvelope(
    event=WebSocketEvent.message, data=saved_msg
)
```

```
await connection_manager.broadcast_to_room(activity_id, msg_envelope)
```

Метод broadcast_to_room серіалізує конверт повідомлення у JSON та надсилає його всім активним сокетам у кімнаті. Мертві з'єднання автоматично видаляються з реєстру. Посилання на алгоритм обмеження частоти запитів для WebSocket описано у п. 2.3.3.

3.3.5 Реалізація фонових задач.

Фонові задачі реалізовано через Celery з RabbitMQ як брокером повідомлень [10]. Планувальник (Beat) запускає три періодичні задачі: оновлення статусів життєвого циклу (кожні 5 хвилин), перевірку нагадувань (кожні 15 хвилин) та зняття протермінованих банів (кожні 5 хвилин). Фрагмент коду задачі оновлення статусів наведено у лістингу 3.5.

Лістинг 3.5 – Задача оновлення життєвого циклу активностей

```
async def update_activity_lifecycle_statuses(
    *,
    activities_col: AsyncCollection,
    membership_col: AsyncCollection,
    db: AsyncSession,
    now: datetime | None = None,
) -> dict[str, int]:
    now = now or datetime.now(timezone.utc)
    result = {"started": 0, "ended": 0, "archived": 0}

    start_result = await activities_col.update_many(
        {"status": ActivityStatus.active.value, "date": {"$lte": now}},
        {
            "$set": {
                "status": ActivityStatus.started.value,
                "started_at": now,
                "updated_at": now,
            }
        },
    )
    result["started"] = getattr(start_result, "modified_count", 0) or 0
```

Цей фрагмент демонструє перший крок алгоритму з п. 2.3.1: вибірку всіх активностей зі статусом active, у яких поле date менше або дорівнює

поточному часу, та атомарне оновлення їхнього статусу на started. Аналогічні операції виконуються для переходів started → ended та ended → archived.

Окрім періодичних, система має п'ять одноразових задач: сповіщення про зміну членства, сповіщення за збігом тегів, сповіщення про скасування активності, завантаження обкладинок ігор з RAWG та видалення осиротілих S3-об'єктів.

3.3.6 Реалізація клієнтського застосунку

Клієнтський застосунок побудовано відповідно до шарової архітектури, описаної у п. 2.2.2. Файл main.ts ініціалізує Vue-додаток із Pinia та Router. Маршрутизатор визначає 12 маршрутів з двома типами гардів: requiresAuth (перенаправляє неавторизованих на модалку входу) та requiresModerator (перенаправляє на сторінку 403). Перед кожним переходом маршрутизатор виконує bootstrap-перевірку сесії через refresh-cookie.

Центральним елементом API-шару є перехоплювач (interceptor) у файлі api/index.ts. Він реалізує чергу запитів, що чекають на оновлення токена. Фрагмент коду наведено у лістингу 3.6.

Лістинг 3.6 – Перехоплювач оновлення токена

```
let isRefreshing = false
let failedQueue: Array<{
  resolve: (token: string) => void
  reject: (error: unknown) => void
}> = []

const processQueue = (error: unknown, token: string | null) => {
  failedQueue.forEach((p) => (error ? p.reject(error) : p.resolve(token!)))
  failedQueue = []
}

api.interceptors.response.use(
  (res) => res,
  async (error) => {
    const originalRequest = error.config as RetryableRequestConfig
    const status = error.response?.status
    if (status === 401 && originalRequest && !originalRequest._retry
        && !originalRequest.url?.includes('/user/refresh')) {
      if (isRefreshing) {
        return new Promise<string>((resolve, reject) => {
          failedQueue.push({ resolve, reject })
```

```
        }).then((token) => {
          originalRequest.headers.Authorization = `Bearer ${token}`
          return api(originalRequest)
        })
      }
      originalRequest._retry = true
      isRefreshing = true
      try {
        const { data } = await api.post<TokenResponse>('/user/refresh')
        const newToken = data.access_token
        const authStore = useAuthStore()
        authStore.setToken(newToken)
        originalRequest.headers.Authorization = `Bearer ${newToken}`
        processQueue(null, newToken)
        return api(originalRequest)
      } catch (refreshError) {
        processQueue(refreshError, null)
        const authStore = useAuthStore()
        authStore.clearAuth()
        authStore.setModal('login')
        return Promise.reject(refreshError)
      } finally {
        isRefreshing = false
      }
    }
    return Promise.reject(error)
  },
)
```

Коли кілька запитів одночасно отримують відповідь 401, лише перший з них виконує оновлення токена. Решта запитів додаються у чергу failedQueue та автоматично повторюються після успішного оновлення. Це запобігає зайвим запитам до серверу.

Композитна функція useChatWebSocket інкапсулює логіку підключення до чату з автоматичним перепідключенням. Фрагмент коду наведено у лістингу 3.7.

Лістинг 3.7 – Автоматичне перепідключення WebSocket

```
socket.value.onclose = (event) => {
  isConnected.value = false
  socket.value = null
  connectionState.value = 'closed'

  const nonRetryableCodes = [1000, 1008, 4000, 4003, 4004, 4008]
  if (event.code === 4000) error.value = 'Помилка авторизації чату'
  if (event.code === 4003) error.value = 'Немає доступу до чату'
  if (event.code === 4004) error.value = 'Чат недоступний для цієї
активності'
  if (event.code === 4008) error.value = 'Занадто багато повідомлень'

  if (!nonRetryableCodes.includes(event.code)
```

```
      && retryCount.value < maxRetries) {
    connectionState.value = 'reconnecting'
    const delay = Math.min(1000 * Math.pow(2, retryCount.value), 10000)
    retryTimeout = setTimeout(() => {
      retryCount.value++
      connect()
    }, delay)
  }
}
```

Затримка між спробами зростає експоненційно: 1, 2, 4, 8 та 10 секунд (максимум 5 спроб). Коди 4000, 4003, 4004 та 4008 є неповторюваними – при їх отриманні перепідключення не виконується, а користувач бачить відповідне повідомлення.

3.3.7 Розгортання системи.

Система розгортається через Docker Compose, який визначає дев'ять сервісів та шість іменованих томів. Кожен інфраструктурний контейнер (PostgreSQL, MongoDB, Redis, RabbitMQ) має healthcheck-перевірку. Серверний застосунок та Celery-воркери стартують лише після проходження healthcheck усіх залежностей. Фрагмент конфігурації наведено у лістингу 3.8.

Лістинг 3.8 – Конфігурація серверного контейнера

```
backend:
  build: ./backend
  ports:
    - "8000:8000"
  env_file: .env
  depends_on:
    postgres:
      condition: service_healthy
    mongodb:
      condition: service_healthy
    redis:
      condition: service_healthy
    rabbitmq:
      condition: service_healthy
    minio:
      condition: service_started
  volumes:
    - backend_venv:/app/.venv
  healthcheck:
    test: ["CMD", "curl", "-f", "http://localhost:8000/health"]
    interval: 15s
    timeout: 5s
    retries: 3
```

Для локальної розробки використовується override-файл, який монтує каталог backend/ у контейнер та активує автоматичне перезавантаження сервера при зміні файлів. Frontend-контейнер запускає Vite у режимі розробки з підтримкою HMR (Hot Module Replacement).

3.4 Керівництво користувача

Для початку роботи з платформою користувач проходить реєстрацію. Форма реєстрації вимагає введення електронної пошти, унікального імені та пароля. Пароль має містити мінімум 8 символів, одну велику літеру та одну цифру. Після успішної реєстрації система автоматично виконує вхід та перенаправляє користувача на головну сторінку. Форму реєстрації та входу наведено на рис. 3.2.

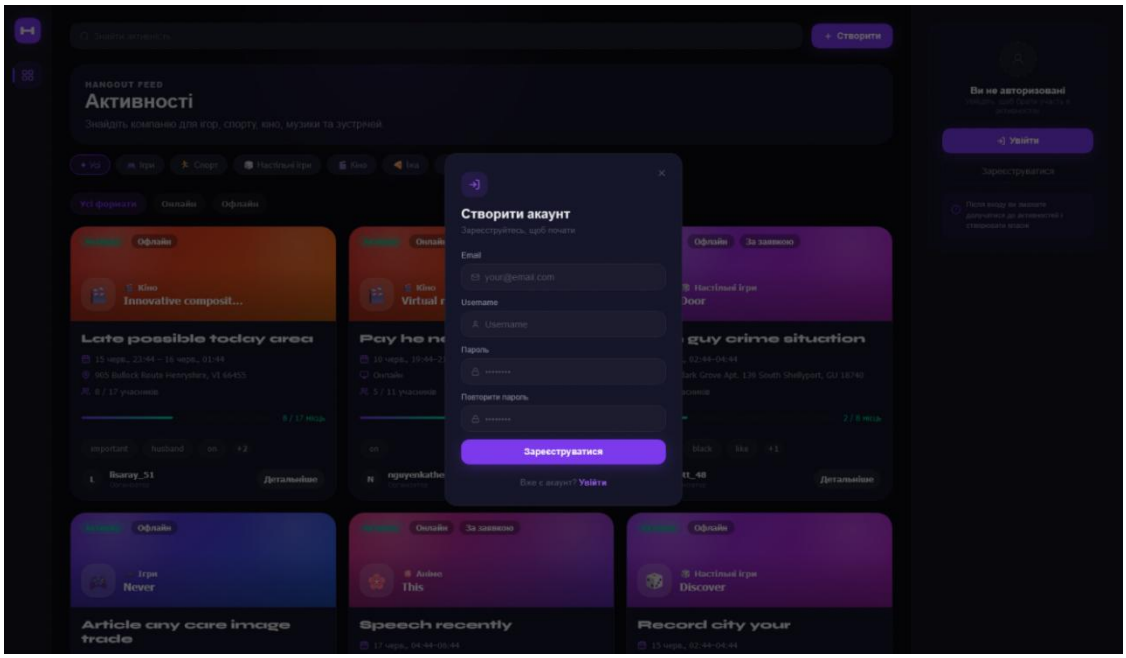

Рисунок 3.2 – Модальне вікно реєстрації та автентифікації

Головна сторінка відображає стрічку всіх активних подій. Користувач може фільтрувати події за категорією та форматом (онлайн/офлайн). Стрічка використовує курсорну пагінацію: нові записи завантажуються автоматично при прокрутці. Кожна картка активності відображає заголовок, категорію,

дату, кількість вільних місць та теги. Головну сторінку з фільтрами наведено на рис. 3.3.

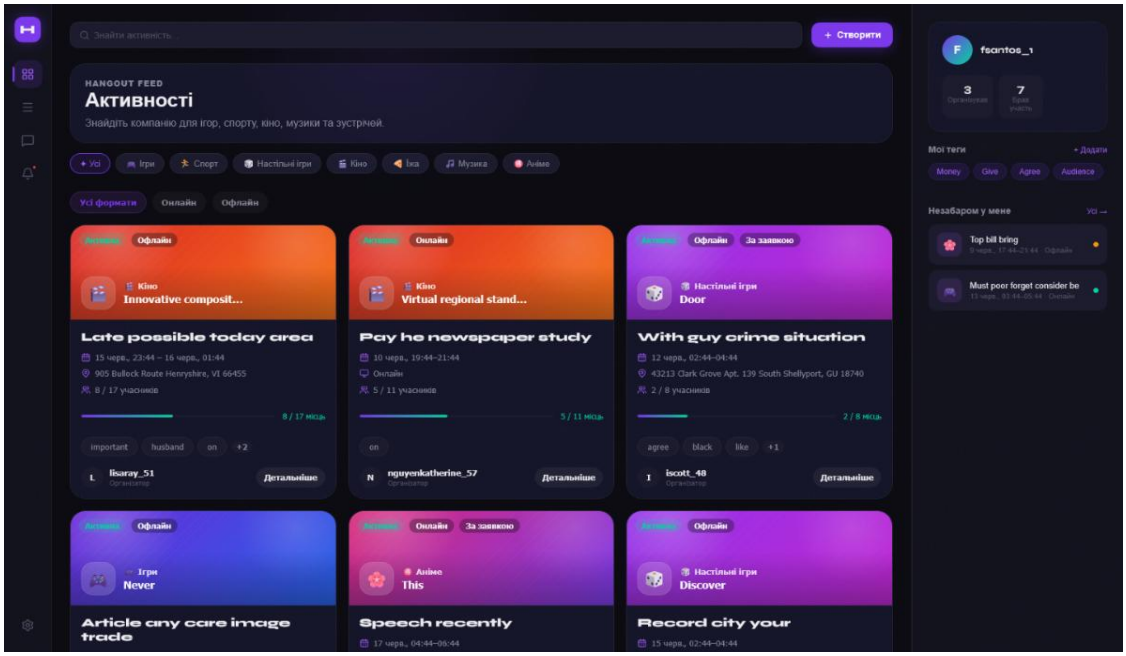

Рисунок 3.3 – Головна стрічка активностей з панеллю фільтрів

Авторизований користувач може створити нову подію. Форма створення адаптується до обраної категорії: для відеоігор з'являється поле з підказками назв ігор через RAWG, для настільних ігор – рівень складності, для спорту – тип заняття. Обов'язковими полями є заголовок, опис, дата, максимальна кількість учасників та теги. Форму створення активності наведено на рис. 3.4.

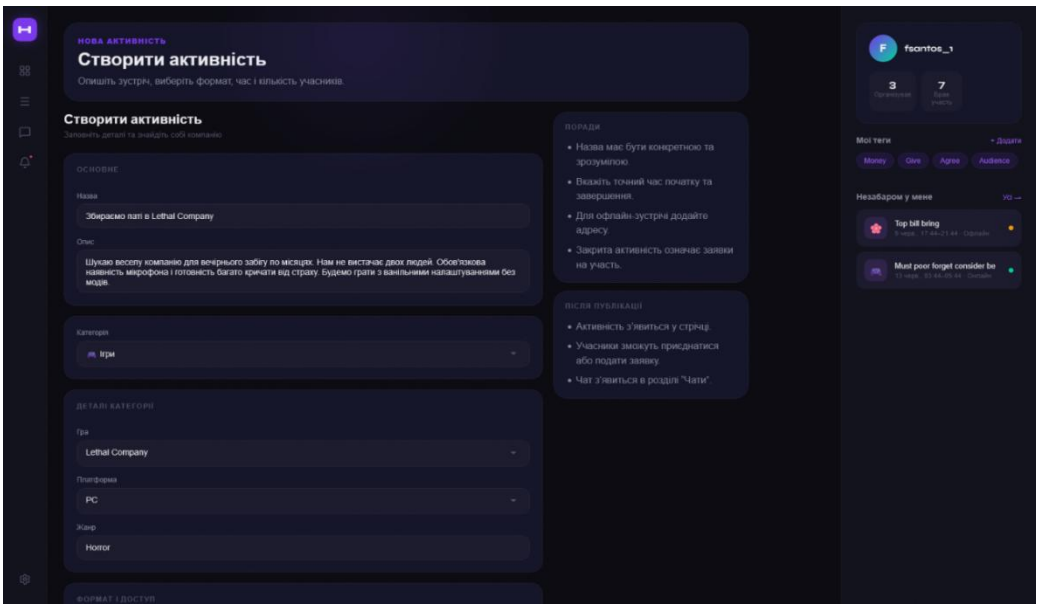

Рисунок 3.4 – Форма створення нової активності (категорія games)

Сторінка деталі відображає повну інформацію про подію: опис, час, формат, учасників та чат. Організатор бачить додаткові елементи управління: кнопки схвалення або відхилення заявок (для закритих подій), вилучення учасників та скасування активності. Сторінку деталі наведено на рис. 3.5.

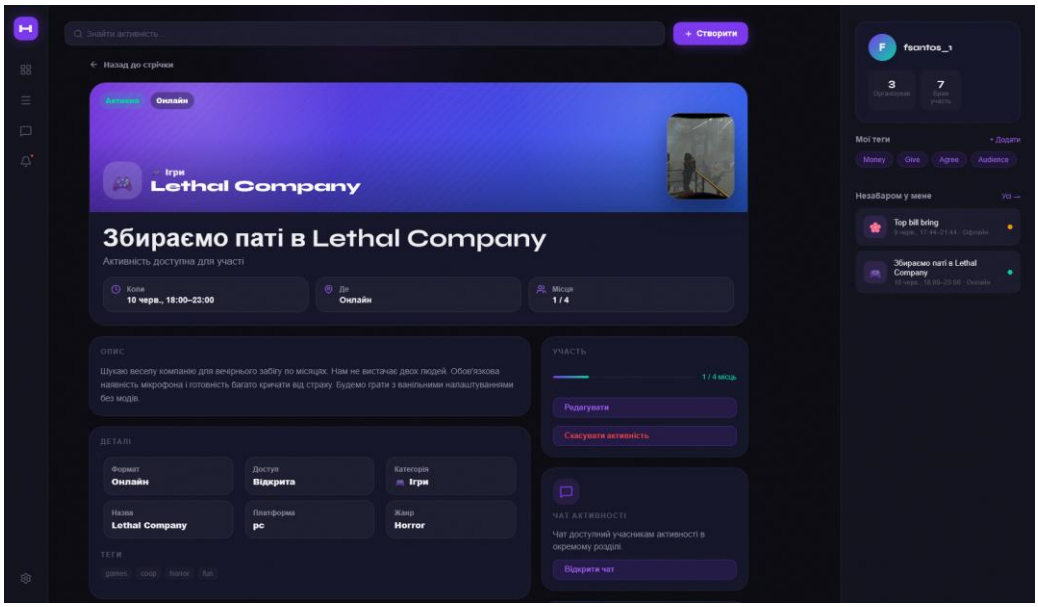

Рисунок 3.5 – Сторінка деталі активності з панеллю учасників

Повідомлення доставляються у реальному часі через WebSocket. Система автоматично генерує сповіщення при приєднанні або виході учасника. У верхній частині чату відображається лічильник онлайн-учасників. Інтерфейс чату наведено на рис. 3.6.

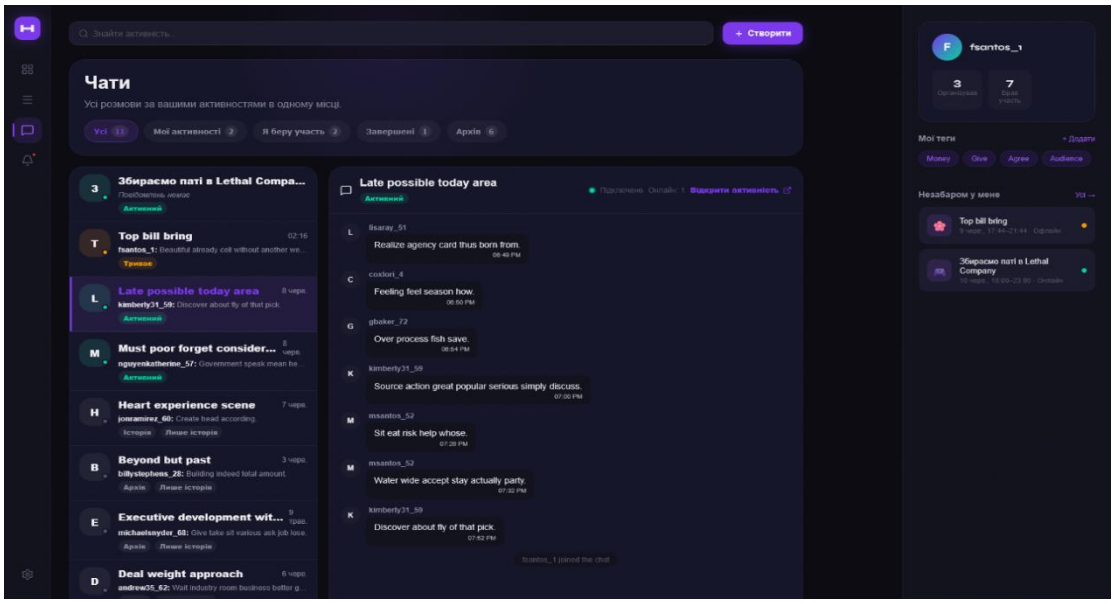

Рисунок 3.6 – WebSocket-чат з системними повідомленнями

Сторінка сповіщень відображає всі події, що стосуються користувача: схвалення та відхилення заявок, нагадування про початок подій, збіги за тегами та результати розгляду скарг. Кожне сповіщення має тип, час та статус прочитання. Користувач може позначити сповіщення як прочитані або видалити їх. Сторінку сповіщень наведено на рис. 3.7.

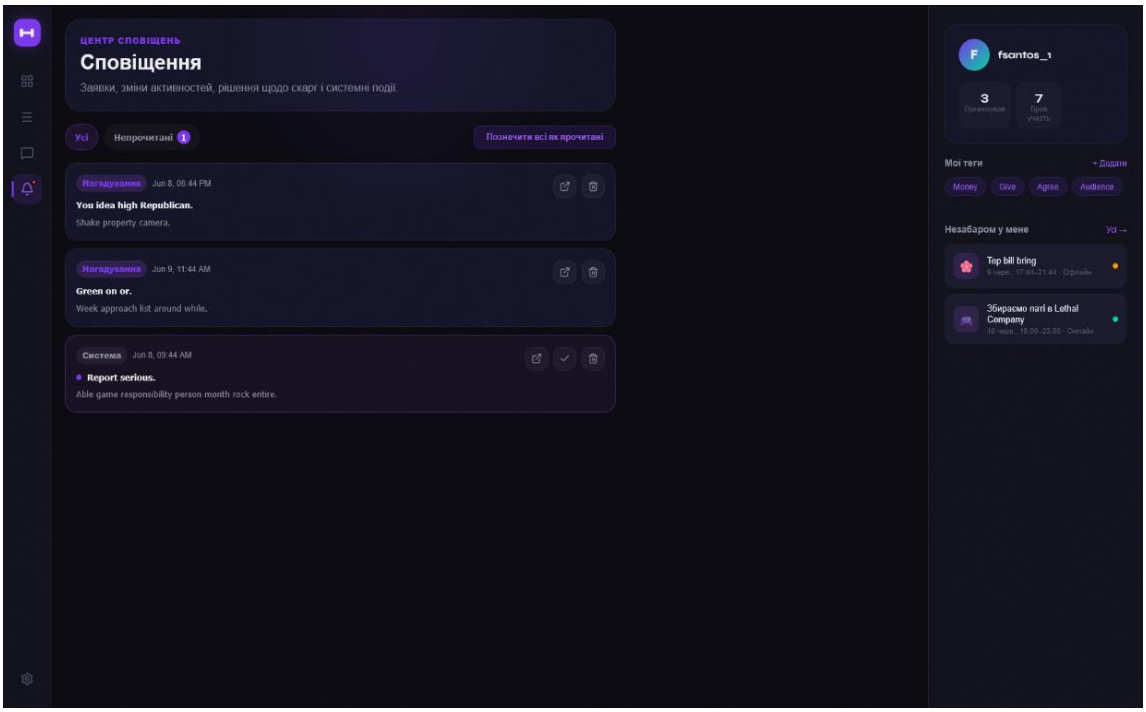

Рисунок 3.7 – Сторінка сповіщень

Модератор має доступ до окремої панелі зі списком скарг. Кожну скаргу можна взяти в роботу, вирішити (з опційним баном порушника) або відхилити. Панель відображає деталі скарги, профіль порушника та історію його попередніх скарг. Панель модератора наведено на рис. 3.8.

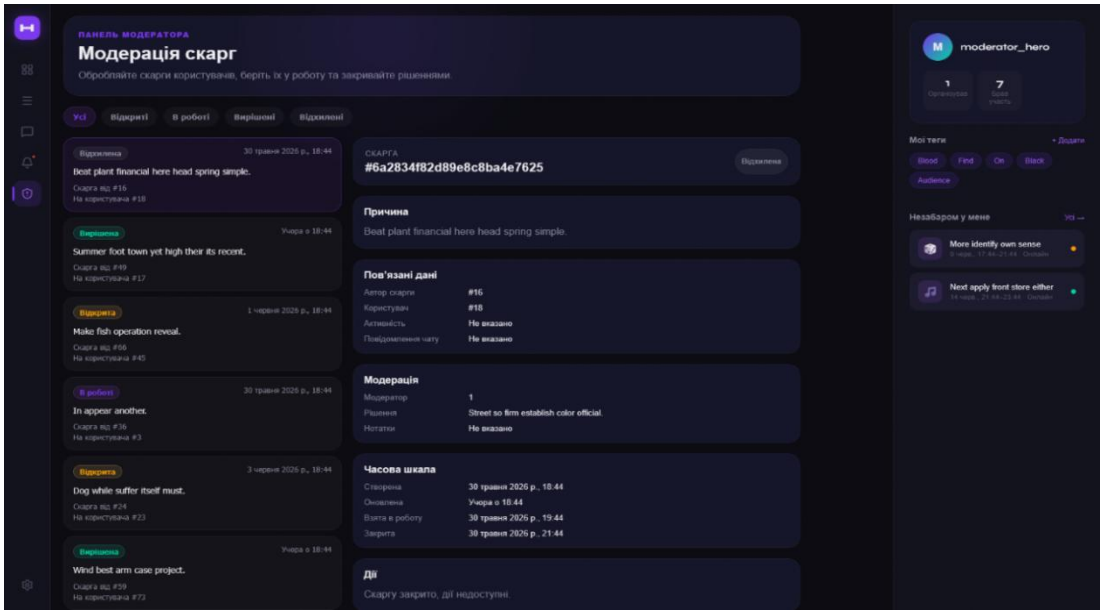

Рисунок 3.8 – Панель модератора зі списком скарг

Сторінка профілю дозволяє змінити ім'я, біографію, аватар та фонове зображення. Сторінку профілю наведено на рис. 3.9.

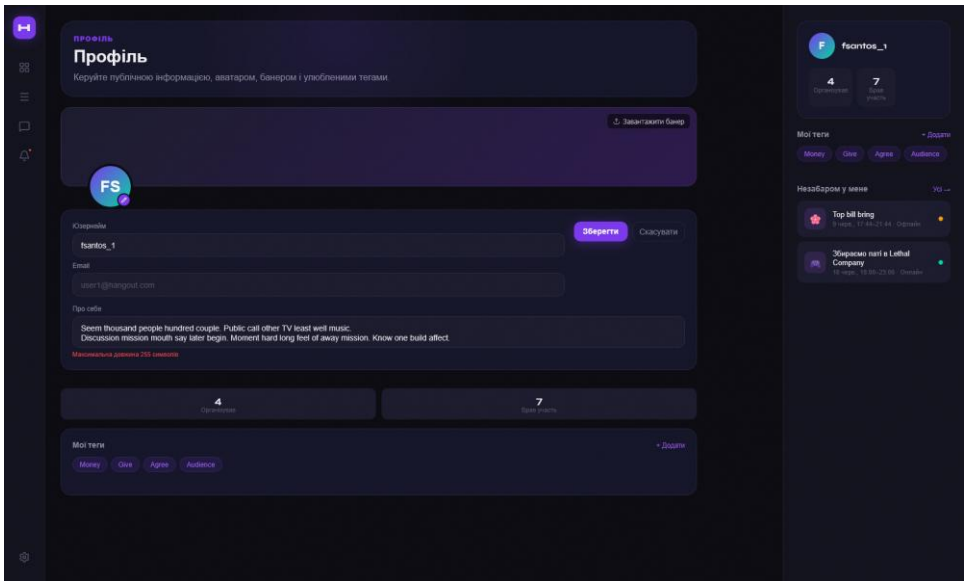

Рисунок 3.9 – Сторінка редагування профілю

Система обробляє помилки та відображає зрозумілі повідомлення. При спробі створити активність із датою менше ніж за дві години від поточного часу система відхиляє запит та відображає помилку валідації (рис. 3.10). При перевищенні ліміту запитів на реєстрацію (більше 3 спроб за хвилину) сервер повертає код 429 (рис. 3.11). При спробі приєднатись до події, час якої перетинається з іншою подією користувача, система повідомляє про конфлікт розкладу (рис. 3.12).

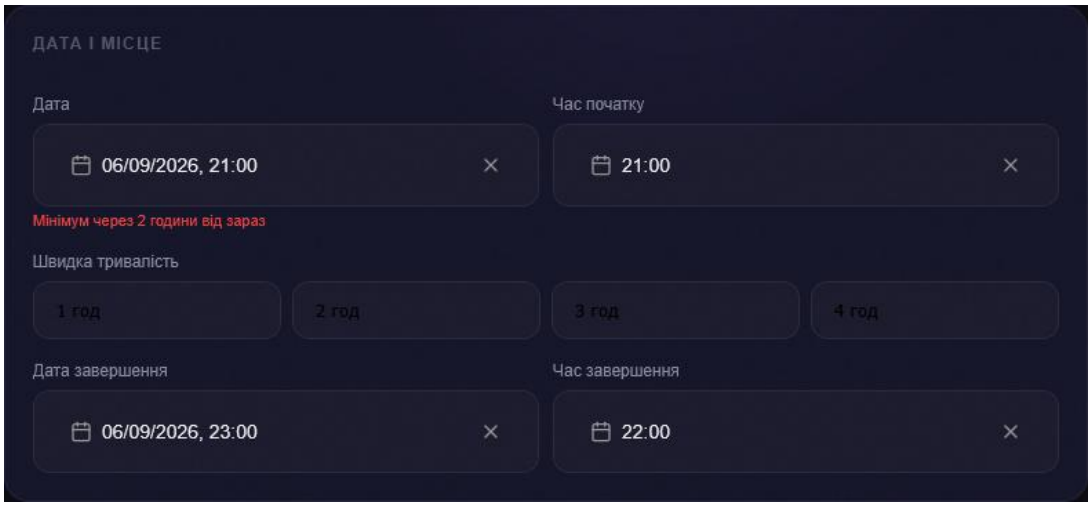

Рисунок 3.10 – Помилка валідації дати при створенні активності

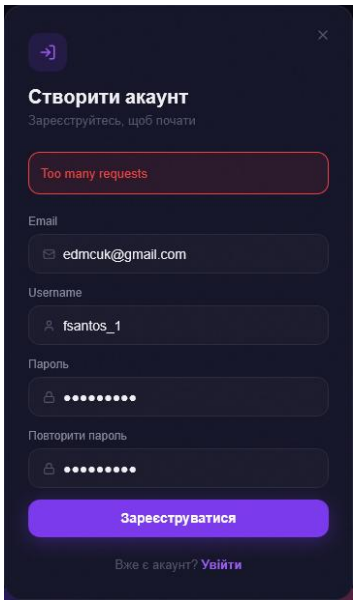

Рисунок 3.11 – Повідомлення про перевищення ліміту запитів (HTTP 429)

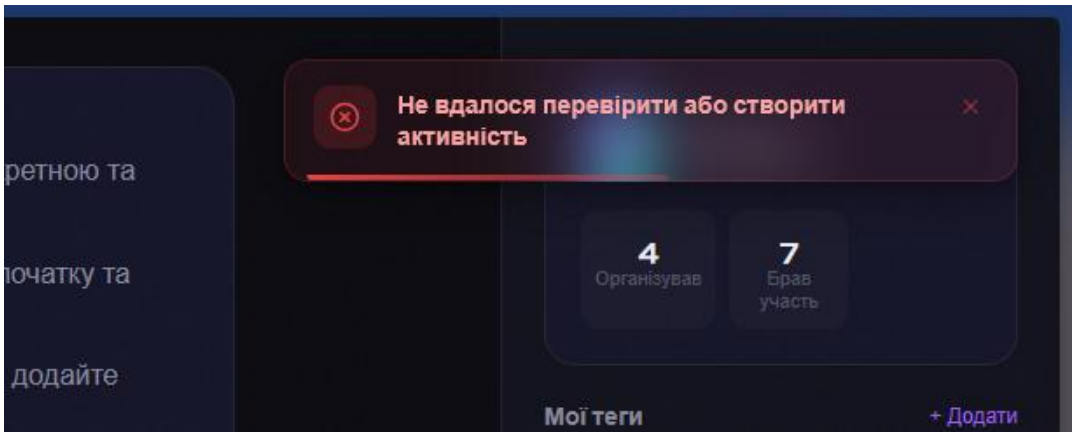

Рисунок 3.12 – Повідомлення про конфлікт розкладу подій

Висновки до розділу

У цьому розділі було обґрунтовано вибір засобів розробки. Серверну частину реалізовано мовою Python 3.13 з фреймворком FastAPI, клієнтську – на Vue 3 з TypeScript. Для збереження даних використано PostgreSQL, MongoDB, Redis та MinIO. Оркестрацію забезпечує Docker Compose.

Також було визначено мінімальні вимоги до серверного обладнання та клієнтського робочого місця.

Було наведено опис програмної реалізації. Описано структуру серверного застосунку з тришаровою архітектурою, реалізацією JWT-автентифікації, модуль активностей, реалізація real-time чату з автоматичним видаленням мертвих з’єднань, підсистему фонових задач на Celery та клієнтський застосунок з перехоплювачем оновлення токенів та експоненційним пере підключенням WebSocket.

Також було продемонстровано роботу платформу з позиції користувача та модератора, що показує повну функціональність продукту та готовність до повноцінної експлуатації.

РОЗДІЛ 4 ОХОРОНА ПРАЦІ

4.1 Організаційно-правові основи забезпечення безпеки праці

Охорона праці є одним із важливих напрямів державної політики будь-якої країни, оскільки спрямована на збереження життя, здоров'я та працездатності працівників у процесі трудової діяльності. У сучасному світі роль охорони праці набуває дедалі більшого значення через розвиток технологій, автоматизацію виробництва, цифровізацію економіки та появу нових професійних ризиків.

За даними міжнародних організацій, наприклад, [13], щороку у світі мільйони працівників зазнають виробничих травм або професійних захворювань. Основними причинами нещасних випадків залишаються недотримання вимог безпеки, недостатня підготовка персоналу, несправність обладнання, людський фактор та відсутність належного контролю за умовами праці. Тому більшість розвинених країн приділяє значну увагу створенню ефективних систем управління охороною праці та постійному вдосконаленню нормативно-правової бази.

Важливу роль у формуванні сучасних підходів до охорони праці відіграють міжнародні організації. Насамперед це Міжнародна організація праці (МОП), яка розробляє міжнародні конвенції та рекомендації, в яких висвітлюються положення щодо створення та забезпечення безпечних і здорових умов праці. Значний вплив також мають документи Європейського Союзу, зокрема Рамкова директива 89/391/ЄЕС, яка визначає основні принципи запобігання професійним ризикам, оцінювання небезпек та відповідальності роботодавців за безпеку працівників.

В Україні питання охорони праці регулюються системою законодавчих і нормативно-правових актів. Основу законодавства становить Конституція України, положеннями якої визначається право кожного на належні, безпечні та здорові умови праці. Головним спеціалізованим нормативним документом

є Закон України «Про охорону праці», в якому визначаються та закріплюються права та обов’язки працівників і роботодавців, порядок організації роботи з охорони праці та відповідальність за порушення встановлених вимог.

Важливу роль у законодавстві України також відіграють Кодекс законів про працю України, Кодекс цивільного захисту України, державні будівельні норми, державні санітарні правила та інші нормативні документи, що регламентують безпечну експлуатацію обладнання, організацію робочих місць, забезпечення пожежної безпеки та створення сприятливих умов праці.

Сучасний розвиток охорони праці в Україні спрямований на гармонізацію національного законодавства з вимогами Європейського Союзу та міжнародними стандартами. Значну роль у цьому процесі відіграє впровадження стандарту ISO 45001, який передбачає ризик-орієнтований підхід до управління безпекою праці та постійне вдосконалення заходів із запобігання виробничому травматизму.

Законодавство виступає основним інструментом забезпечення належного рівня охорони праці як у світі, так і в Україні. Саме завдяки нормативному регулюванню встановлюються обов’язкові вимоги щодо безпеки праці, визначаються права працівників та відповідальність роботодавців, а також створюються умови для зниження виробничого травматизму та професійних захворювань.

4.2 Характеристика об’єкта та виявлення потенційних небезпек

У даному розділі об’єктом дослідження виступатиме робоче місце адміністратора соціальної платформи для пошуку людей та організації спільних активностей. Основним призначенням адміністратора є забезпечення стабільної роботи платформи, контроль за її функціонуванням, взаємодія з користувачами та підтримання актуальності інформації.

Робоче місце адміністратора може розташовуватися в окремому офісному приміщенні або в адміністративному кабінеті організації, яка забезпечує

функціонування соціальної платформи. Приміщення повинно відповідати санітарно-гігієнічним вимогам щодо освітлення, мікроклімату, вентиляції та організації робочого простору. Для забезпечення комфортних умов праці приміщення обладнується системою штучного освітлення, засобами опалення та вентиляції, а також меблями, що відповідають ергономічним вимогам.

Робоче місце, як правило, обладнане персональним комп'ютером або ноутбуком, монітором, клавіатурою, комп'ютерною мишею та засобами доступу до мережі Інтернет. За необхідності можуть використовуватися додатковий монітор, гарнітура для проведення онлайн-консультацій, вебкамера, джерело безперебійного живлення та інші допоміжні технічні засоби. Усі пристрої підключаються до електромережі та забезпечують безперервний доступ до інформаційної системи.

Соціальна платформа призначена для об'єднання користувачів за спільними інтересами та організації різноманітних активностей, таких як спортивні заходи, туристичні поїздки, культурні події, навчальні зустрічі та інші форми спільного дозвілля. Адміністратор забезпечує коректне функціонування веб-платформи, контролює роботу окремих модулів системи та здійснює моніторинг активності користувачів.

До основних функціональних обов'язків адміністратора належать:

- контроль працездатності соціальної платформи та своєчасне виявлення технічних збоїв;
- керування обліковими записами користувачів;
- перевірка достовірності інформації, що розміщується на платформі;
- модерація повідомлень, коментарів та оголошень користувачів;
- обробка звернень та консультація користувачів щодо роботи системи;
- моніторинг статистики відвідуваності та активності користувачів;
- забезпечення захисту персональних даних і контролю доступу до інформаційних ресурсів;
- оновлення інформаційного наповнення платформи та підтримання актуальності сервісів;

- взаємодія з технічними спеціалістами у разі виникнення несправностей програмного забезпечення або серверного обладнання.

Робота адміністратора має переважно інтелектуальний характер та виконується у сидячому положенні з використанням комп'ютерної техніки протягом більшої частини робочого часу. У процесі роботи працівник здійснює пошук, обробку та аналіз інформації, спілкується з користувачами через електронні канали зв'язку, приймає рішення щодо модерації контенту та забезпечує безперервну роботу інформаційної системи.

Незважаючи на те, що така діяльність не пов'язана з виконанням важких фізичних робіт або експлуатацією складного виробничого обладнання, під час виконання посадових обов'язків працівник може зазнавати впливу низки небезпечних і шкідливих факторів виробничого середовища. Виявлення таких факторів є необхідним етапом оцінювання умов праці, оскільки дозволяє визначити потенційні ризики для здоров'я працівника та розробити заходи щодо їх попередження або мінімізації.

Таблиця 4.1 – Виявлення потенційних небезпек на робочому місці адміністратора соціальної платформи (на основі [14])

№	Потенційна небезпека	Джерело виникнення	Можливі наслідки
1	Надмірне навантаження на органи зору	Тривале спостереження за екраном монітора, робота з текстовою та графічною інформацією	Погіршення гостроти зору, сухість очей, головний біль, швидка втомлюваність
2	Незручне положення тіла протягом робочого дня	Тривале перебування у сидячій позі, недостатня ергономічність меблів	Біль у спині та шиї, порушення постави, дискомфорт у суглобах і м'язах
3	Перевтома внаслідок високої концентрації уваги	Постійний контроль роботи платформи, аналіз повідомлень та звернень користувачів	Зниження працездатності, помилки в роботі, підвищена втома
4	Емоційне перенапруження	Робота зі скаргами, конфліктними ситуаціями та проблемними користувачами	Стрес, нервове виснаження, погіршення психологічного стану

5	Небезпека ураження електричним струмом	Комп’ютерне обладнання, мережеві пристрої, електричні кабелі та розетки	Травмування працівника, опіки, порушення роботи серцево-судинної системи
6	Виникнення пожежонебезпечної ситуації	Перевантаження електромережі, пошкодження ізоляції проводів, несправність обладнання	Загоряння техніки, пошкодження майна, загроза життю та здоров’ю людей
7	Недостатній рівень освітлення робочої зони	Невдала організація освітлення або недостатня кількість світильників	Підвищене навантаження на зір, зниження продуктивності праці
8	Відблиски та засвічення на екрані	Неправильне розташування монітора відносно джерел світла або вікон	Дискомфорт під час роботи, погіршення сприйняття інформації, втома очей
9	Несприятливі параметри мікроклімату	Недостатня вентиляція, надмірна температура або сухість повітря в приміщенні	Погіршення самопочуття, сонливість, зниження концентрації уваги
10	Шумове навантаження	Робота комп’ютерної техніки, серверного обладнання, офісної оргтехніки	Дратівливість, швидка втомлюваність, зниження концентрації
11	Ризик втрати або пошкодження інформації	Збої програмного забезпечення, відключення електроживлення, помилки користувачів	Втрата даних, необхідність повторного виконання роботи, стресові ситуації
12	Небезпека спотикання або падіння	Кабелі живлення та мережеві дроти, розміщені в зоні пересування працівників	Забої, травми кінцівок, пошкодження обладнання
13	Перенавантаження кистей рук	Тривале використання клавіатури та комп’ютерної миші	Больові відчуття у кистях, дискомфорт у суглобах, розвиток професійних захворювань
14	Підвищене інформаційне навантаження	Одночасна робота з великою кількістю повідомлень, заявок та інформаційних ресурсів	Психічна втома, зниження уважності, збільшення кількості помилок
15	Відсутність регулярних перерв у роботі	Тривала безперервна робота за комп’ютером	Загальна перевтома організму, погіршення самопочуття, зниження ефективності праці

4.3 Дослідження ризику реалізації потенційних небезпек на об’єкті проєктування та розробка заходів щодо їх попередження

Одним із найважливіших етапів забезпечення безпечних умов праці є оцінювання професійних ризиків. Під ризиком в охороні праці розуміють імовірність виникнення небезпечної події та тяжкість можливих наслідків для працівника внаслідок впливу небезпечних або шкідливих виробничих факторів [15].

Основною метою оцінювання ризиків є своєчасне виявлення потенційних небезпек, визначення рівня їх небезпечності та розроблення ефективних заходів щодо їх усунення або зменшення негативного впливу на працівників. Завдяки проведенню оцінки ризиків можна не лише попередити виникнення нещасних випадків і професійних захворювань, а й створити більш комфортні та безпечні умови праці [15].

Процес оцінювання ризиків передбачає послідовне виконання декількох етапів. Спочатку визначаються всі можливі небезпеки, які можуть виникати на робочому місці або під час виконання працівником своїх функціональних обов’язків. Після цього аналізуються причини виникнення небезпек, визначається ймовірність їх реалізації та оцінюється тяжкість можливих наслідків. На основі отриманих результатів встановлюється рівень ризику та приймаються рішення щодо необхідності впровадження додаткових заходів безпеки [16].

Результати оцінювання ризиків дозволяють визначити найбільш небезпечні фактори виробничого середовища, встановити пріоритетність заходів безпеки та підвищити рівень захищеності працівників.

Оцінку ризиків для деяких з визначених для робочого місця адміністратора небезпек будемо проводити за допомогою матриці оцінки ризиків. Результати оцінки та рекомендації щодо підвищення рівня безпеки наведені нижче.

Таблиця 4.2 – Результати оцінки ризиків 1

Показник	Характеристика
Небезпека	Надмірне навантаження на органи зору
Джерело небезпеки	Тривала робота за монітором
Категорія серйозності	III – критична
Характеристика наслідків	Тимчасове погіршення самопочуття та працездатності, розвиток проблем із зором
Рівень ймовірності	В – імовірна
Характеристика ймовірності	Небезпека регулярно виникає під час роботи
Індекс ризику	3В
Рівень ризику	Небажаний

Рекомендації щодо зниження ризику:

- забезпечити правильне розташування монітора відносно очей працівника;
- використовувати достатнє природне та штучне освітлення;
- робити короткі перерви кожні 45-60 хвилин роботи;
- виконувати вправи для очей;
- використовувати монітори з якісним відображенням інформації.

Таблиця 4.3 – Результати оцінки ризиків 2

Показник	Характеристика
Небезпека	Тривале перебування у сидячому положенні
Джерело небезпеки	Робота за комп’ютером протягом більшої частини робочого дня
Категорія серйозності	III – критична
Характеристика наслідків	Порушення постави, біль у спині, шиї та суглобах
Рівень ймовірності	В – імовірна
Характеристика ймовірності	Постійно присутня під час виконання роботи
Індекс ризику	3В
Рівень ризику	Небажаний

Рекомендації щодо зниження ризику:

- використовувати ергономічне офісне крісло;
- правильно налаштувати висоту столу та монітора;
- періодично змінювати положення тіла;
- виконувати розминку під час перерв;

- забезпечити достатній простір для ніг та зручне розташування обладнання.

Таблиця 4.4 – Результати оцінки ризиків 3

Показник	Характеристика
Небезпека	Нервово-психічне перенапруження
Джерело небезпеки	Велика кількість звернень користувачів, необхідність швидкого прийняття рішень та модерації контенту
Категорія серйозності	III – критична
Характеристика наслідків	Стрес, перевтома, зниження концентрації уваги та продуктивності праці
Рівень ймовірності	В – імовірна
Характеристика ймовірності	Виникає регулярно в процесі роботи
Індекс ризику	3В
Рівень ризику	Небажаний

Рекомендації щодо зниження ризику:

- раціонально розподіляти робоче навантаження;
- дотримуватися режиму праці та відпочинку;
- уникати тривалої безперервної роботи;
- автоматизувати частину рутинних операцій;
- створити комфортні умови праці та сприятливий психологічний клімат у колективі.

Висновки до розділу

У розділі з охорони праці проведено аналіз стану охорони праці, визначено, що на світовому рівні це питання викликає занепокоєння спільнот, що підтверджується статистикою, яку ведуть визнані міжнародні організації. Зазначено, що формування та удосконалення як міжнародного, так і національного законодавства з охорони праці необхідно для забезпечення більш високого рівня безпеки праці працівників, задіяних у різних секторах

економічної діяльності країн. Розглянуто основні національні законодавчі документи, які стосуються сфери охорони праці.

У розділі приділено увагу аналізу робочого місця адміністратора соціальної платформи, за результатами якого складено деталізований перелік можливих небезпек, джерел їх виникнення на можливих наслідків, що є важливою складовою подальшої оцінки ризиків, виконання якої дає можливість пріоритезувати дії щодо підвищення рівня охорони та безпеки праці. Із використанням матриці оцінки ризиків визначено рівень ризиків для декількох небезпек та наведено загальні рекомендації щодо попередження або мінімізації їх впливу.

ЗАГАЛЬНІ ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено веб-платформу Hangout, призначену для пошуку компанії та організації спільного проведення часу.

Було проведено аналіз предметної області, який виявив потребу в сучасних цифрових інструментах для подолання проблеми соціальної ізоляції. Аналіз існуючих аналогів, зокрема Meetup та Eventbrite, дозволив сформувати технічні вимоги до платформи, орієнтовані на підтримку локальних офлайн-зустрічей та короткотривалих онлайн-активностей.

Спроєктовано архітектуру програмної системи з використанням підходу Polyglot Persistence. Застосування реляційної бази PostgreSQL для транзакційних даних користувачів і документоорієнтованої бази MongoDB для гнучких даних активностей, чатів і сповіщень забезпечило баланс між надійністю та масштабованістю зберігання даних. Розроблено комплекс UML- та ER-діаграм, що описують структурні й динамічні аспекти платформи.

Програмно реалізовано серверну частину системи мовою Python із використанням асинхронного фреймворку FastAPI. Імплементовано механізми безпеки, включно з JWT-автентифікацією з ротацією токенів, а також систему керування станом активностей за допомогою фонових задач Celery. Реалізовано обмін повідомленнями в реальному часі на базі WebSocket. Також створено клієнтський вебзастосунок на основі Vue 3 з використанням Composition API та стейт-менеджера Pinia. Розроблено адаптивний інтерфейс, що забезпечує зручну взаємодію користувача з функціоналом платформи: пошуком, фільтрацією активностей, управлінням заявками та онлайн-чатом.

Отже, поставлені на початку роботи цілі було досягнуто. Розроблена платформа відповідає визначеним вимогам, має практичне значення та може бути використана як інструмент соціальної взаємодії для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Our Epidemic of Loneliness and Isolation: The U.S. Surgeon General's Advisory on the Healing Effects of Social Connection and Community / Office of the U.S. Surgeon General. – Washington, D.C., 2023. – 82 с. – Режим доступу: https://www.hhs.gov/sites/default/files/surgeon-general-social-connection-advisory.pdf
2. Adding Years to Life and Life to Years: The Global State of Health and Well-being [Електронний ресурс] / McKinsey Health Institute. – 2023. – Режим доступу: https://www.mckinsey.com/mhi/our-insights/adding-years-to-life-and-life-to-years
3. 2024 Meetup Measurement Report: Community Trends and Platform Data [Електронний ресурс] / Meetup. – 2024. – Режим доступу: https://www.meetup.com/blog/
4. TRNDS 2024: Event Trends Report [Електронний ресурс] / Eventbrite. – 2024. – Режим доступу: https://www.eventbrite.com/blog/event-trends/
5. Richards M. Fundamentals of Software Architecture: An Engineering Approach / M. Richards, N. Ford. – Sebastopol, CA : O'Reilly Media, 2020. – 419 p. – ISBN 978-1492043454.
6. Madden N. API Security in Action / N. Madden. – Shelter Island, NY : Manning Publications, 2020. – 576 p. – ISBN 978-1617296024.
7. Ramírez S. FastAPI: Modern, Fast (high-performance) Web Framework for Building APIs with Python [Електронний ресурс] / S. Ramírez. – 2025. – Режим доступу: https://fastapi.tiangolo.com/
8. PostgreSQL 17 Documentation [Електронний ресурс] / The PostgreSQL Global Development Group. – 2025. – Режим доступу: https://www.postgresql.org/docs/17/
9. MongoDB 8.0 Manual [Електронний ресурс] / MongoDB, Inc. – 2025. – Режим доступу: https://www.mongodb.com/docs/manual/

10. Celery — Distributed Task Queue: Documentation [Електронний ресурс] / Celery Project. – 2025. – Режим доступу: https://docs.celeryq.dev/
11. Jones M. B. JSON Web Token (JWT) : RFC 7519 / M. B. Jones, J. Bradley, N. Sakimura. – Internet Engineering Task Force (IETF), 2015. – Режим доступу: https://datatracker.ietf.org/doc/html/rfc7519
12. OWASP Top 10:2025 — The Ten Most Critical Web Application Security Risks [Електронний ресурс] / OWASP Foundation ; кер. проекту O. Katz. – 2025. – Режим доступу: https://owasp.org/Top10/
13. Statistics on Safety and Health at Work [Електронний ресурс] / International Labour Organization. – Режим доступу: https://ilostat.ilo.org/topics/safety-and-health-at-work/
14. Гігієнічна класифікація праці за показниками шкідливості та небезпечності факторів виробничого середовища, важкості та напруженості трудового процесу [Електронний ресурс] : наказ МОЗ України від 08.04.2014 № 248. – Режим доступу: https://zakon.rada.gov.ua/laws/show/z0472-14#Text
15. Оцінка ризиків на робочих місцях [Електронний ресурс] / Полтавський обласний центр зайнятості. – Режим доступу: https://pol.dcz.gov.ua/novyna/ocinka-ryzykiv-na-robochyh-miscyah
16. Основи оцінки ризиків: як проводити та застосовувати результати [Електронний ресурс] / LinkedIn. – Режим доступу: https://ua.linkedin.com/pulse/risk-assessment-basics-how-conduct-apply-findings-zdt5e?tl=uk