

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

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

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

на тему:

**«Вебплатформа для пошуку та підбору партнерів у  
багатокористувацьких онлайн-іграх»**

Виконав студент групи ВТ-22-1  
НАГАЛЬ Олександр Сергійович

Керівник роботи:  
ГРОМСЬКИЙ Олександр Олександрович

Рецензент:  
ВЛАСЕНКО Олег Васильович

Житомир – 2026

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

**«ЗАТВЕРДЖУЮ»**

Зав. кафедри інженерії  
програмного забезпечення

Тетяна Вакалюк

**«28» лютого 2026 р.**

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: **НАГАЛЬ Олександр Сергійович**

Керівник роботи: **ГРОМСЬКИЙ Олександр Олександрович**

Тема роботи: **«Вебплатформа для пошуку та підбору партнерів у багатокористувацьких онлайн-іграх»,**

затверджена Наказом закладу вищої освіти від **«28» лютого 2026 р., №92**

Вихідні дані для роботи: **процес організації та автоматизації підбору партнерів у багатокористувацьких онлайн-іграх. Використання веб-орієнтованих технологій для розробки системи лобі, матчмейкінгу та комунікації гравців у вебплатформі «GameFinder».**

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

| Розділ | Консультант    | Завдання видав | Завдання прийняв |
|--------|----------------|----------------|------------------|
| 1      | Громський О.О. | 01.03.2026р.   | 01.03.2026р      |
| 2      | Громський О.О. | 01.03.2026р    | 01.03.2026р      |
| 3      | Громський О.О. | 01.03.2026р    | 01.03.2026р      |

## РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного продукту – вебплатформи для пошуку та підбору партнерів у багатокористувацьких онлайн-іграх «GameFinder», а також пояснювальної записки. Пояснювальна записка до кваліфікаційної роботи містить 102 сторінки, 54 ілюстрації, 22 таблиці, 3 додатки обсягом 10 сторінок та 30 використаних літературних джерел.

Метою роботи є розробка вебплатформи для пошуку та підбору партнерів у багатокористувацьких онлайн-іграх із використанням сучасних вебтехнологій. Для досягнення мети застосовано методи системного аналізу, об'єктно-орієнтованого проєктування та реляційного моделювання баз даних.

У результаті роботи розроблено серверну архітектуру на базі Symfony 7 та MySQL, створено клієнтський інтерфейс за допомогою Bootstrap 5. Реалізовано інтелектуальну систему матчмейкінгу, систему лобі з фільтрацією, обмін повідомленнями в реальному часі через Mercure, голосовий чат за допомогою WebRTC, а також інтеграції зі Steam API та Stripe.

Отримані результати рекомендуються для використання геймерами з метою швидкого пошуку надійних напарників та покращення досвіду командної гри, а також кіберспортивними організаціями для формування професійних команд.

**КЛЮЧОВІ СЛОВА:** ВЕБПЛАТФОРМА, ЛОБІ, МАТЧМЕЙКІНГ, ОНЛАЙН-ГРА, ЧАТ.

|           |      |                |        |      |                                                                                  |                   |      |        |
|-----------|------|----------------|--------|------|----------------------------------------------------------------------------------|-------------------|------|--------|
|           |      |                |        |      | ІПЗ.КР.Б-121.26-ПЗ                                                               |                   |      |        |
| Змн.      | Арк. | № докум.       | Підпис | Дата | Вебплатформа для пошуку та підбору партнерів у багатокористувацьких онлайн-іграх | Лім.              | Арк. | Аркуші |
| Розроб.   |      | Нагаль О.С.    |        |      |                                                                                  |                   |      |        |
| Перевір.  |      |                |        |      |                                                                                  |                   | 2    |        |
| Керівник  |      | Громський О.О. |        |      |                                                                                  | ФІКТ, Гр. ВТ-22-1 |      |        |
| Н. контр. |      |                |        |      |                                                                                  |                   |      |        |
| Зав. каф. |      |                |        |      |                                                                                  |                   |      |        |

## ABSTRACT

The bachelor's qualification work consists of a software product – a web platform for searching and matchmaking partners in multiplayer online games "GameFinder" – as well as an explanatory note. The explanatory note to the qualification work contains 102 pages, 54 illustrations, 22 tables, 3 appendices spanning 10 pages, and 30 used literary sources.

The aim of the work is to develop a web platform for searching and selecting partners in multiplayer online games using modern web technologies. To achieve the goal, methods of system analysis, object-oriented design, and relational database modeling were applied.

As a result, a server architecture based on Symfony 7 and MySQL was developed, and a client interface was created using Bootstrap 5. An intelligent matchmaking system, a lobby system with filtering, real-time messaging via Mercure, voice chat using WebRTC, as well as integrations with Steam API and Stripe were implemented.

The obtained results are recommended for use by gamers to quickly find reliable partners and improve the team play experience, as well as by esports organizations for forming professional teams.

KEYWORDS: CHAT, LOBBY, MATCHMAKING, ONLINE GAME, WEB PLATFORM.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    | 4    |

## ЗМІСТ

|                                                                                            |           |
|--------------------------------------------------------------------------------------------|-----------|
| <b>РЕФЕРАТ</b> .....                                                                       | <b>3</b>  |
| <b>ABSTRACT</b> .....                                                                      | <b>4</b>  |
| <b>ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ</b> .....                                                     | <b>6</b>  |
| <b>ВСТУП</b> .....                                                                         | <b>7</b>  |
| 1.1.    Постановка задачі.....                                                             | 9         |
| 1.2.    Огляд існуючих аналогічних рішень .....                                            | 10        |
| 1.3.    Вибір архітектурного шаблону та технологій розробки.....                           | 15        |
| 1.4.    Обґрунтування вибору інструментальних засобів .....                                | 18        |
| Висновки до першого розділу .....                                                          | 22        |
| <b>РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ «GAMEFINDER»</b> .....                              | <b>24</b> |
| 2.1.    Визначення варіантів використання та об'єктно-орієнтованої структури системи ..... | 24        |
| 2.2.    Розробка бази даних та математичних алгоритмів системи .....                       | 36        |
| 2.3.    Проектування алгоритмів роботи вебплатформи.....                                   | 46        |
| 2.4.    Опис реалізації функціоналу вебплатформи .....                                     | 50        |
| Висновки до другого розділу .....                                                          | 56        |
| <b>РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ «GAMEFINDER»</b> .....            | <b>58</b> |
| 3.1.    Порядок встановлення та налаштування вебплатформи.....                             | 58        |
| 3.2.    Структура інтерфейсу вебплатформи.....                                             | 62        |
| 3.3.    Тестування роботи програмного продукту .....                                       | 80        |
| Висновки до третього розділу .....                                                         | 85        |
| <b>ВИСНОВКИ</b> .....                                                                      | <b>87</b> |
| <b>СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ</b> .....                                                    | <b>89</b> |
| <b>ДОДАТКИ</b> .....                                                                       | <b>92</b> |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

**БД** – база даних, масив даних, що обробляється за допомогою ЕОМ.

**СКБД** – система керування базами даних, програмне забезпечення для управління БД.

**API** (англ. Application Programming Interface) – набір способів і правил взаємодії програм.

**CRUD** (англ. Create, Read, Update, Delete) – функції: створення, читання, оновлення, видалення.

**CSS** (англ. Cascading Style Sheets) – спеціальна мова для опису зовнішнього вигляду HTML.

**HTML** (англ. HyperText Markup Language) – стандартизована мова розмітки вебсторінок.

**HTTP** (англ. HyperText Transfer Protocol) – протокол прикладного рівня для передачі даних.

**MVC** (англ. Model-View-Controller) – шаблон, що розділяє додаток на модель, вигляд та контролер.

**MySQL** – вільна реляційна система управління базами даних.

**OAuth** (англ. Open Authorization) – протокол доступу до ресурсів без передачі пароля.

**ORM** (англ. Object-Relational Mapping) – технологія зв'язку БД з об'єктно-орієнтованим кодом.

**PHP** (англ. Hypertext Preprocessor) – скриптова мова для розробки серверної частини вебдодатків.

**REST** (англ. Representational State Transfer) – архітектурний стиль взаємодії вебдодатків через HTTP.

**SQL** (англ. Structured Query Language) – мова для управління реляційними базами даних.

**UI** (англ. User Interface) – засіб взаємодії користувача з інформаційною системою.

|      |      |                 |        |      |                    |      |
|------|------|-----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.     |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський. О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.        | Підпис | Дата |                    | 6    |

## ВСТУП

**Актуальність теми.** Сучасна індустрія багатокористувацьких онлайн-ігор стрімко розвивається, об'єднуючи мільйони гравців по всьому світу. Ігри, такі як Counter-Strike 2, Dota 2, Valorant та інші, потребують командної взаємодії, де результат напряму залежить від злагодженості та комунікації між учасниками команди.

Однією з ключових проблем для гравців є пошук надійних та відповідних за рівнем партнерів для спільної гри. Існуючі внутрішньоігрові системи підбору часто не враховують мовні переваги, часовий пояс, вік та рівень досвіду гравців, що призводить до негативного ігрового досвіду та збільшення кількості токсичних взаємодій у команді.

Розробка спеціалізованої вебплатформи, яка дозволить гравцям створювати лобі з детальними фільтрами, переглядати профілі потенційних напарників, планувати ігрові сесії та обмінюватися повідомленнями, є актуальною задачею, що покращить якість командної гри та знизить рівень конфліктів.

**Мета роботи.** Метою випускної роботи є розробка алгоритмів роботи та реалізація програмного комплексу вебплатформи для пошуку та підбору партнерів у багатокористувацьких онлайн-іграх із використанням сучасних вебтехнологій.

**Завдання роботи.** Встановлена мета обумовлює комплекс взаємопов'язаних завдань, першим з яких є проведення ґрунтовного аналізу існуючих рішень для пошуку ігрових партнерів з метою визначення їхніх ключових переваг та архітектурних недоліків. На основі результатів аналізу необхідно розробити оптимальну архітектуру та узагальнену структурну модель майбутньої системи, а також здійснити обґрунтований вибір програмних засобів та технологічного стеку для її реалізації. Наступним важливим етапом виступає проєктування реляційної бази даних, здатної надійно зберігати інформацію про користувачів, ігровий каталог, активні лобі, повідомлення та супутні сутності.

|      |      |                 |        |      |                    |      |
|------|------|-----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.     |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський. О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.        | Підпис | Дата |                    | 7    |

Крім того, вимагається детальне проектування логічних структурних складових та математичних алгоритмів роботи системи, зокрема механізмів управління лобі, інтерактивних чатів, системи відгуків та інтелектуального матчмейкінгу. Завершальні завдання полягають у безпосередній програмній реалізації комплексу вебплатформи з використанням обраних технологій, а також у проведенні комплексного тестування системи з подальшим покриттям її відповідною технічною документацією.

**Об'єкт дослідження.** Об'єктом дослідження є процес організації та автоматизації підбору партнерів у багатокористувацьких онлайн-іграх.

**Предмет дослідження.** Предметом дослідження є використання веб-орієнтованих технологій для розробки системи лобі, матчмейкінгу та комунікації гравців у вебплатформі «GameFinder».

**Методи дослідження.** В роботі було використано методи системного аналізу для формування вимог, об'єктно-орієнтованого проектування для створення архітектури програмного продукту, реляційного моделювання для побудови бази даних, а також методи вебпрограмування та дизайну для реалізації клієнтської частини.

**Практичне значення отриманих результатів.** Розроблений програмний продукт може бути використаний у практичній діяльності геймерських спільнот для організації спільних ігор, турнірів та тренувань. Впровадження платформи дозволить гравцям швидко знаходити партнерів відповідного рівня, що зменшить час очікування та підвищить якість командної взаємодії. Крім того, інструмент є корисним для кіберспортивних організацій при скаутингу нових талантів завдяки наявності прозорості системи рейтингів та відгуків.

|      |      |                 |        |      |                    |      |
|------|------|-----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.     |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський. О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.        | Підпис | Дата |                    | 8    |

# РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ВЕБПЛАТФОРМИ ДЛЯ ПОШУКУ ПАРТНЕРІВ У ОНЛАЙН-ІГРАХ

## 1.1. Постановка задачі

Розробка вебплатформи «GameFinder» вимагає чіткого визначення функціональних та нефункціональних вимог, а також вирішення ключових проблем предметної області. Основний недолік сучасного онлайн-геймінгу полягає у відсутності спеціалізованих інструментів для цілеспрямованого підбору напарників. Існуючі внутрішньоігрові алгоритми (Matchmaking) здебільшого базуються виключно на обчисленні математичного рейтингу гравця (ELO) та затримці мережі (Ping). При цьому повністю ігноруються соціально-поведінкові фактори: бажана мова спілкування, вікова категорія, рівень стресостійкості та ігрові вподобання. Це системно призводить до командних конфліктів і погіршення ігрового досвіду [30]. Отже, першочерговим технічним завданням є розробка підсистеми ігрових лобі. Впровадження розширеного механізму фільтрації дозволить користувачам попередньо формувати команди з оптимальною сумісністю до ініціалізації ігрової сесії [27].

Друге стратегічне завдання полягає у забезпеченні надійної ідентифікації та авторизації користувачів у системі. Платформа має підтримувати класичну автентифікацію через Email та інтеграцію протоколу OAuth 2.0 (Google, Discord) для оптимізації процесу реєстрації [18]. Для підвищення безпеки та верифікації ігрового досвіду передбачається використання Steam API [9]. Шляхом прив'язки Steam-профілю система здатна автоматично синхронізувати бібліотеку ігор користувача та здобуті досягнення. Кожен гравець отримає персоналізований профіль із відображенням ігрової статистики та агрегованого глобального рейтингу довіри, що унеможливило масове створення підроблених акаунтів.

Третім архітектурним компонентом є впровадження надійних засобів комунікації, які функціонують у режимі реального часу. Базовий текстовий чат має обробляти повідомлення асинхронно, без повного перезавантаження

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    | 9    |

сторінки, використовуючи протокол Server-Sent Events [6]. Оскільки для координації дій в кіберспортивних дисциплінах текстового обміну недостатньо, платформа повинна інтегрувати підсистему голосового зв'язку. Використання стандарту WebRTC дозволить встановлювати захищені однорангові (peer-to-peer) з'єднання між клієнтськими браузерами, гарантуючи передачу аудіопотоку з мінімальною мережевою затримкою [7].

Для превентивної протидії деструктивній поведінці ставиться завдання розробки модулів взаємного оцінювання та глобального рейтингу [23]. Після завершення спільної сесії гравці формуватимуть оцінку напарників, що конвертуватиметься у загальний показник репутації (Trust Factor). Облікові записи з критично низьким рейтингом автоматично підпадатимуть під системні обмеження. Крім того, передбачено розробку алгоритмічної бази для розумного матчмейкінгу, що здійснюватиме автоматичний підбір релевантних гравців на основі векторної сумісності їхніх профілів.

З метою забезпечення самоокупності проєкту планується інтеграція модуля Premium-підписки із застосуванням платіжного шлюзу Stripe [10]. Дана підписка надаватиме виключно візуальні косметичні привілеї без впливу на базовий ігровий баланс. Адміністрування платформи здійснюватиметься через захищені панелі керування з інструментами для модерації контенту та обробки скарг. Реалізація зазначеного комплексу вимог забезпечить створення масштабованої вебплатформи, здатної витримувати високі навантаження та функціонувати як надійне середовище для кіберспортивної спільноти.

## 1.2. Огляд існуючих аналогічних рішень

Перед початком проєктування та розробки власної платформи було проведено ґрунтовний аналіз існуючих рішень на ринку, призначених для пошуку ігрових партнерів. Метою цього дослідження було виявлення сильних сторін конкурентів, аналіз їхнього функціоналу та виявлення недоліків, які можна вирішити у новому продукті.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 10   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Одним із популярних рішень є GamerLink – спеціалізований мобільний додаток, орієнтований на пошук геймерів зі схожими ігровими вподобаннями. Головною особливістю застосунку є автоматизована система матчингу, яка підбирає партнерів на основі обраних ігор та спільного часу доступності. Інтерфейс додатку (рис. 1.1) вирізняється інтуїтивно зрозумілою навігацією. З технічної точки зору GamerLink є кросплатформним мобільним додатком (ймовірно, створеним за допомогою React Native). Серверна частина використовує архітектуру REST API з базою даних NoSQL для гнучкого зберігання профілів. Додаток покладається на механізми push-сповіщень для комунікації в реальному часі, однак не використовує повноцінні постійні з'єднання (WebSockets) для безперервних голосових сесій, що є технічним компромісом на користь економії ресурсів мобільних пристроїв. Відсутність повноцінної вебверсії робить платформу менш доступною для ПК-геймерів.

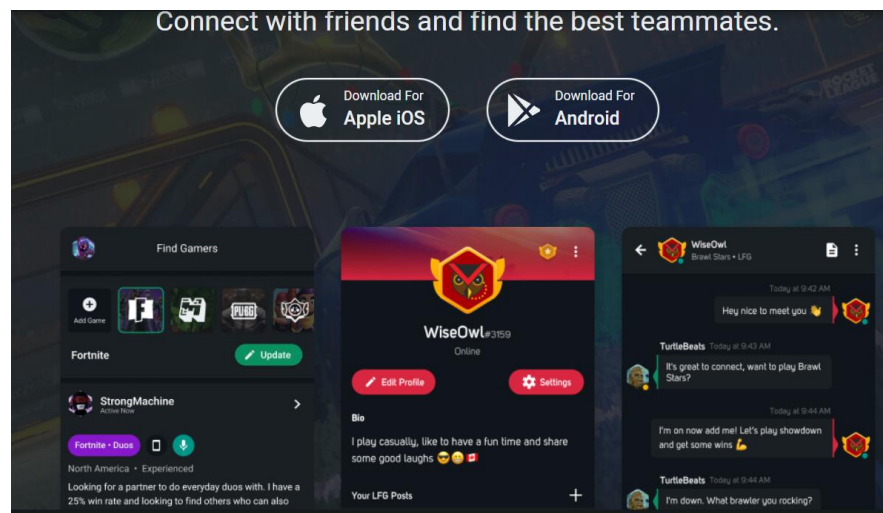


Рисунок 1.1 – Головний екран, профіль та чат мобільного додатка GamerLink

Найбільш поширеним універсальним інструментом для кооперації геймерів на сьогодні є Discord [19]. Ця платформа пропонує децентралізоване рішення у вигляді численних тематичних серверів (Рис. 1.2). Архітектура Discord є еталоном для високонавантажених систем реального часу. Клієнтські додатки побудовані на базі Electron (для ПК) та React Native (для мобільних). Серверна частина складається з сотень мікросервісів, написаних мовами Elixir, Rust та Go

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 11   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

[29]. Обмін повідомленнями в реальному часі відбувається через WebSockets, а голосовий та відеозв'язок базується на спеціально модифікованому стеку WebRTC. Для зберігання гігантських обсягів даних використовується Cassandra. Хоча технічно це надпотужне рішення, воно є не спеціалізованим для LFG (Looking For Group), а універсальним. Процес пошуку команди часто зводиться до хаотичного читання повідомлень у текстових каналах без єдиної системи фільтрації за ігровими параметрами чи рейтингу гравців.

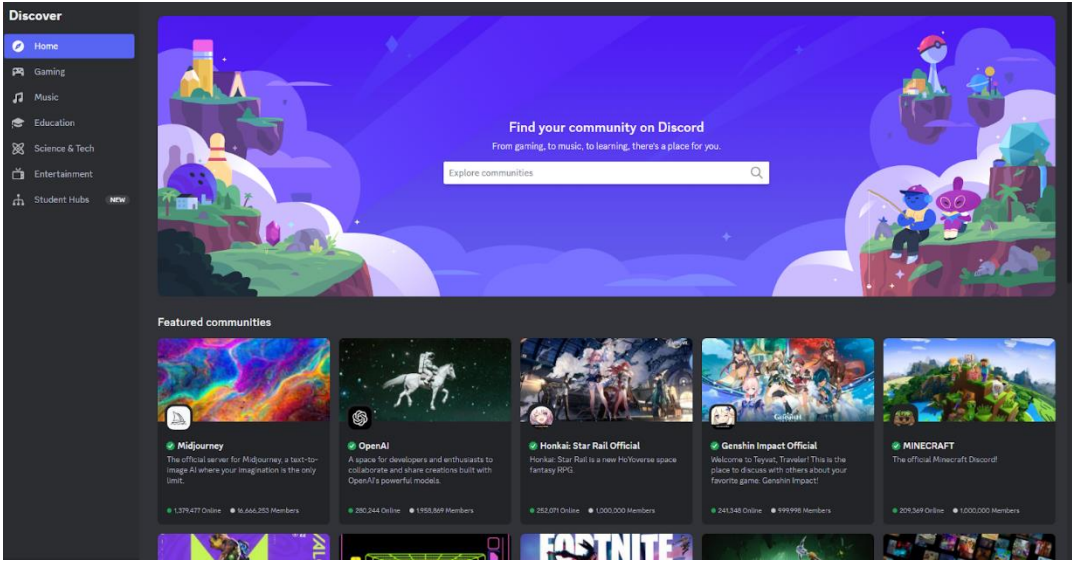


Рисунок 1.2 – Discord сервери(сторінка пошуку груп)

Іншим цікавим підходом є платформа GameTree, яка позиціонує себе як багатофункціональний сервіс для створення ігрових спільнот (Рис. 1.3). З технічного боку GameTree являє собою класичний SPA (Single Page Application). Оскільки платформа робить сильний акцент на психологічний метчинг та визначення сумісності, її інфраструктура інтенсивно використовує алгоритми обробки даних для аналізу графів зв'язків між гравцями. Real-time комунікація забезпечується через WebSockets. Разом з тим, інтерфейс GameTree часто критикують за перевантаженість великою кількістю другорядних соціальних функцій, а значна кількість скриптів негативно впливає на час відгуку інтерфейсу. Реєстрація є занадто тривалою через обов'язкове психологічне тестування.

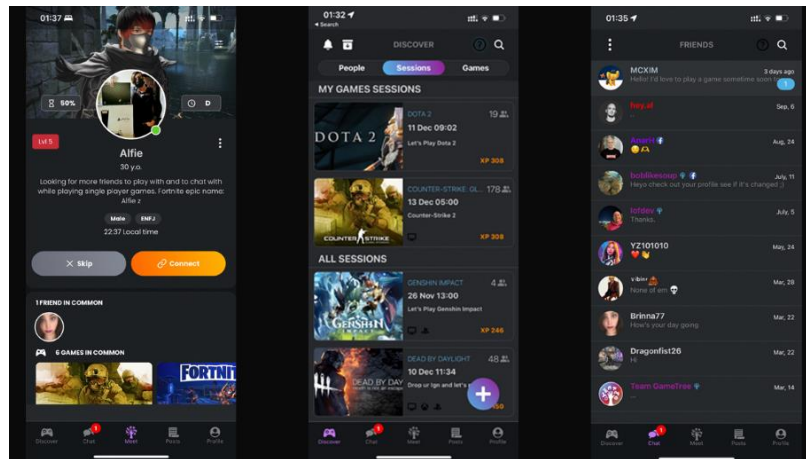


Рисунок 1.3 – Інтерфейс додатку GameTree

Також варто відзначити TeamFind.com – спеціалізовану вебплатформу, орієнтовану переважно на кіберспортивний сегмент (Рис. 1.4). Архітектурно TeamFind виглядає як класичний монолітний вебдодаток попереднього покоління зі стандартним серверним рендерингом. Головним технічним недоліком системи є повна відсутність сучасних підходів до real-time комунікації – замість WebSockets або Server-Sent Events тут застосовується базовий HTTP polling або необхідність ручного оновлення сторінки, що робить комунікацію повільною та незручною [25]. Каталог ігор вкрай обмежений, а дизайн не є адаптивним.

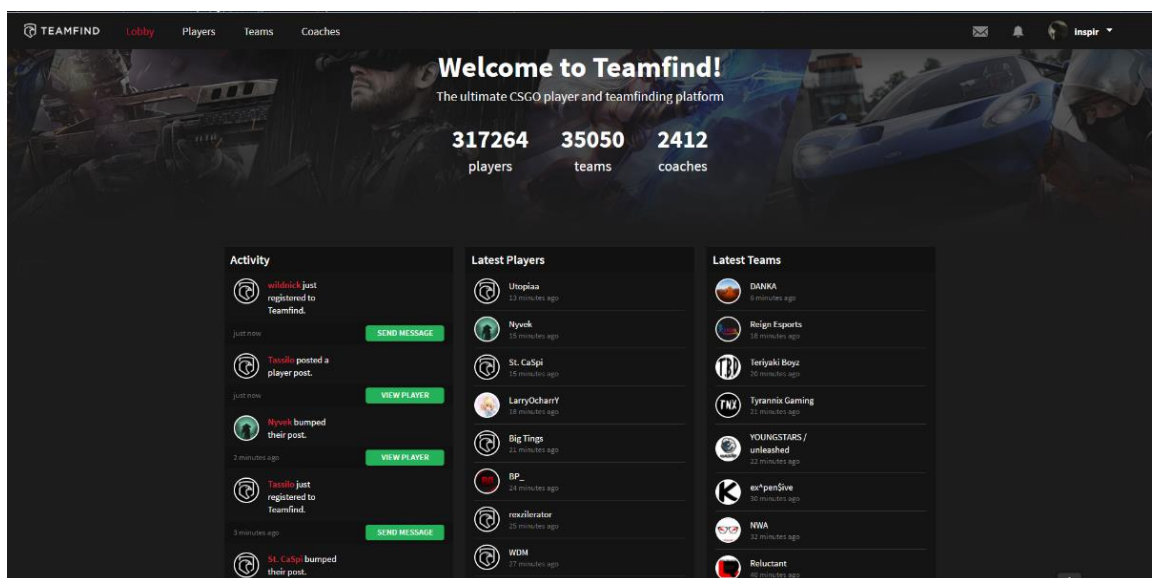


Рисунок 1.4 – Інтерфейс вебплатформи TeamFind

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 13   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Для наочного порівняння функціональних та технічних можливостей розглянутих програмних продуктів та розроблюваної платформи GameFinder було складено зведену таблицю (табл. 1.1).

Таблиця 1.1.

Порівняння існуючих рішень

| Характеристика           | GamerLink       | Discord             | GameTree   | TeamFind     | GameFinder             |
|--------------------------|-----------------|---------------------|------------|--------------|------------------------|
| Архітектура              | Mobile-first    | Мікросервіси        | SPA + API  | Моноліт      | Моноліт (MVC)          |
| Real-time підхід         | Push-сповіщення | WebSockets / WebRTC | WebSockets | HTTP Polling | SSE (Mercure) / WebRTC |
| Вебінтерфейс             | —               | +                   | +          | +            | +                      |
| Система лобі з фільтрами | +/-             | —                   | +          | +/-          | +                      |
| Рейтингова система       | —               | —                   | -          | +/-          | +                      |
| Інтеграція зі Steam      | —               | —                   | +          | —            | +                      |
| Система відгуків         | —               | —                   | -          | —            | +                      |
| Чат (груповий/приватний) | +/-             | +                   | +          | —            | +                      |
| Календар подій           | —               | +/-                 | +          | —            | +                      |
| OAuth авторизація        | +               | +                   | +          | —            | +                      |
| Система друзів           | +               | +                   | +          | —            | +                      |

Базуючись на результатах детального аналізу, було визначено, що жодне з рішень не покриває повною мірою потреб гравців у зручному пошуку напарників із сучасним підходом до технологій. Платформа «GameFinder» має вирішити ці проблеми шляхом реалізації гнучких фільтрів, сучасної архітектури реального часу, інтелектуального алгоритму підбору та прозорої системи рейтингів.

### 1.3. Вибір архітектурного шаблону та технологій розробки

При проектуванні вебплатформи «GameFinder» було розглянуто кілька сучасних архітектурних підходів. Мікросервісна архітектура (Microservices), яка використовується в Discord, передбачає розбиття системи на десятки незалежних сервісів (автентифікація, чати, матчмейкінг), кожен з яких розгортається окремо. Це ідеально для платформ із мільйонами користувачів, оскільки дозволяє масштабувати кожен компонент незалежно. Проте для даного проєкту використання мікросервісів є надлишковим, оскільки суттєво ускладнює розгортання, вимагає налаштування складної інфраструктури та значно збільшує час розробки для невеликої команди. Іншим варіантом була архітектура SPA + API (Single Page Application), де клієнтська частина написана на React або Vue і працює автономно в браузері, а серверна частина виступає виключно як REST API. Цей підхід забезпечує відмінну інтерактивність, однак має недоліки у вигляді необхідності підтримки двох окремих кодових баз та ускладнення SEO-оптимізації. У результаті було обрано монолітну MVC архітектуру (Model-View-Controller). Це класичний підхід, де бекенд і фронтенд інтегровані в єдиній кодовій базі, що забезпечує високу швидкість розробки, легкість тестування та зручність розгортання без надмірного ускладнення інфраструктури.

Визначившись з типом нашого продукту, можна перейти до побудови архітектури серверної та клієнтської частини. Для серверної частини будемо використовувати архітектуру MVC, адже вона є однією із найзручніших для реалізації вебплатформи такого типу.

Model View Controller (MVC) – це фундаментальний патерн, який знайшов застосування в багатьох технологіях, дав розвиток новим підходам і кожен день полегшує життя розробникам. Вперше патерн MVC з'явився досить давно, проте його концепція незмінна: розробники повинні були придумати архітектурне рішення, яке дозволяло б відокремити графічний інтерфейс від бізнес-логіки, а бізнес-логіку – від даних. Таким чином, в класичному варіанті, MVC складається з трьох частин, які і дали йому назву.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 15   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Під Model, зазвичай, розуміється частина, що містить в собі функціональну бізнес-логіку програми. Model повинна бути повністю незалежна від інших частин продукту. Модельний шар нічого не повинен знати про елементи дизайну, і яким чином він буде відображатися. Досягається результат, що дозволяє змінювати View даних, те, як вони відображаються, не чіпаючи саму Model.

Модель представляє собою бізнес-логіку програми, яка володіє знаннями виключно про себе саму і не залежить від контролера чи представлення. У розроблюваній системі цей компонент реалізований через Doctrine ORM, що відповідає за строгу валідацію даних та їх безпечне збереження у реляційну базу даних MySQL [4].

В обов'язки View входить відображення даних, отриманих від Model. Однак, View не може безпосередньо впливати на об'єкти моделі, маючи до них доступ «тільки на читання». У цьому шарі реалізується візуальне представлення інформації будь-яким зручним способом, а також може міститися мінімальний код логіки відображення. Типовими прикладами View у розробленій вебплатформі є використання шаблонізатора Twig у поєднанні з семантичним HTML та CSS-фреймворком Bootstrap 5 [12].

Основна ідея цього патерну в тому, що і Controller, і View залежать від Model, але Model ніяк не залежить від цих двох компонент.

Контролер виступає ключовою сполучною ланкою у системі, приймаючи всі вхідні HTTP-запити від користувача. Він визначає, яке саме представлення має бути показано в даний момент, реагуючи на події з боку інтерфейсу. За необхідності контролер може впливати на модель для оновлення даних та визначати інше представлення для формування кінцевої відповіді клієнту.

Отже, визначившись із архітектурою серверної частини, перейдемо до клієнтської. Оскільки класичний MVC працює за принципом «запит-відповідь», що не підходить для миттєвого обміну повідомленнями у чатах, клієнтська частина платформи була доповнена подієво-орієнтованою архітектурою на базі

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 16   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

протоколу Server-Sent Events (SSE) [8]. Для цього ми будемо використовувати автономний хаб Mercure [6].

Mercure – це відкритий протокол та інструмент, який виконує роль маршрутизатора подій та допомагає керувати потоком даних вашого додатка в реальному часі. Ця архітектура дуже схожа на патерн Publisher/Subscriber і бере те, що ми вже знаємо про WebSockets, але розширює його, щоб бути більш ефективним та простим для нашого монолітного PHP-додатку [1].

Два найбільші вдосконалення, які пропонує Mercure – це єдиний хаб, доступний для всіх клієнтів, на який можна підписатись, і можливість працювати через стандартний HTTP/2 протокол без блокування серверних процесів PHP.

Значна частина архітектури Mercure побудована на взаємодії трьох ключових елементів. Першим є видавець (Publisher) – серверний контролер Symfony, який після зміни стану моделі генерує дію та відправляє POST-запит [2]. Другим елементом виступає сам хаб (Hub), що працює як окремий сервіс, приймаючи події від видавця та утримуючи активні HTTP-з'єднання з браузерами. Третім компонентом є підписник (Subscriber), реалізований через клієнтський JavaScript всередині представлення, який підключається до хабу та очікує на конкретні канали. Коли хаб отримує нові дані, він миттєво передає їх усім підписникам без необхідності додаткових запитів. Основна робота Mercure – допомогти системі відстежувати зміни стану і доставляти їх клієнтам без перезавантаження сторінки. А голосовий чат реалізовано за допомогою технології WebRTC, яка встановлює пряме peer-to-peer з'єднання між клієнтами [7].

Архітектурний підхід MVC був обраний не лише через простоту, а й через високу здатність до тестування. Завдяки чіткому розділенню відповідальності між компонентами, забезпечується висока стабільність системи: контролери не містять бізнес-логіки, а моделі не залежать від представлення. Крім того, використання подієво-орієнтованої архітектури (Event-Driven Architecture) у поєднанні з класичним MVC створює гібридну екосистему. З одного боку, Symfony забезпечує надійний захист та строгу обробку даних для маршрутів, а з

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 17   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

іншого – Mercure бере на себе навантаження з підтримки тисяч активних з'єднань, не блокуючи основні PHP-процеси. Такий симбіоз дозволяє уникнути головного недоліку монолітних систем – поганої продуктивності при великій кількості одночасних запитів.

Окрему увагу під час проєктування було приділено організації обміну аудіоданими через WebRTC. На відміну від традиційних медіасерверів, які пропускають весь голосовий трафік через себе та створюють "вузьке місце" (bottleneck) у системі, обрана архітектура встановлює пряме P2P з'єднання між браузерами клієнтів. Сервер платформи виконує лише роль сигнального вузла (Signaling Server) на етапі ініціалізації дзвінка. Після того як користувачі обмінялися необхідними дескрипторами через Mercure, голосові пакети передаються напряму, що зводить затримку до мінімуму і гарантує максимальну якість звуку в ігрових лобі. Це дозволяє суттєво зекономити ресурси сервера та забезпечити комфортне спілкування гравців під час динамічних ігрових сесій.

#### 1.4. Обґрунтування вибору інструментальних засобів

Середовищем розробки нашого продукту буде PhpStorm, так як це потужна платформа для розробки вебсервісів, яка дає змогу підключати та використовувати безліч корисних плагінів. PhpStorm – комерційне кросплатформне інтегроване середовище розробки для PHP. Розробляється компанією JetBrains на основі платформи IntelliJ IDEA. PhpStorm є інтелектуальним редактором для PHP, HTML і JavaScript з можливостями аналізу коду на льоту, запобігання помилок в коді і автоматизованими засобами рефакторингу для PHP і JavaScript. Автодоповнення коду в PhpStorm підтримує специфікації PHP аж до найновіших версій. Є повноцінний SQL-редактор з можливістю редагування отриманих результатів запитів. PhpStorm розроблений на основі платформи IntelliJ IDEA, написаної на Java. Користувачі можуть розширити функціональність середовища розробки за рахунок установки плагінів, розроблених для платформи IntelliJ, або написавши власні плагіни.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 18   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Мовою розробки серверної частини буде PHP (версії 8.2), адже він досить простий та легкий у використанні [1]. Також, PHP має гнучку систему типізації (з можливістю суворої типізації у нових версіях), що робить його потужним інструментом для бекенд-розробки. PHP – це широко використовувана мова сценаріїв загального призначення з відкритим вихідним кодом. Говорячи простіше, PHP — це мова програмування, спеціально розроблена для написання вебдодатків (сценаріїв), що виконуються на вебсервері. Аббревіатура PHP означає «Hypertext Preprocessor (Препроцесор Гіпертексту)». Синтаксис мови бере початок з C, Java і Perl. PHP досить простий для вивчення. Перевагою PHP є надання веб-розробникам можливості швидкого створення динамічно генеруючихся вебсторінок. Значною відмінністю PHP від будь-якого коду, що виконується на стороні клієнта, наприклад, JavaScript, є те, що PHP-скрипти виконуються на стороні сервера. Ви навіть можете налаштувати свій сервер таким чином, щоб клієнти отримували лише готовий результат (HTML-код) і навіть не змогли дізнатися, який код був виконаний на сервері. PHP дозволяє створювати якісні вебдодатки за дуже короткі терміни, отримуючи продукти, які легко модифікуються і підтримуються в майбутньому.

Для вибору системи керування базами даних (СКБД) було проаналізовано декілька популярних реляційних рішень (табл. 1.2).

Таблиця 1.2.

#### Порівняння СКБД

| Характеристика            | MySQL | PostgreSQL | SQLite |
|---------------------------|-------|------------|--------|
| Відкритий код             | +     | +          | +      |
| Підтримка транзакцій      | +     | +          | +/-    |
| Масштабованість           | +     | +          | -      |
| Підтримка JSON            | +/-   | +          | +/-    |
| Інтеграція з Doctrine ORM | +     | +          | +      |
| Швидкість читання         | +     | +/-        | +      |

Базою даних для нас буде слугувати MySQL (версії 8.0), адже основною її перевагою є надійність та безпека [5]. MySQL – вільна реляційна система управління базами даних. Розробку та підтримку MySQL здійснює корпорація «Oracle». Продукт поширюється як під GNU General Public License, так і під власною комерційною ліцензією. Саме завдяки корпоративним замовленням в ній з'явилися потужні механізми на кшталт реплікації. MySQL є відмінним рішенням для малих і середніх вебдодатків. Зазвичай вона входить до складу локальних серверів типу LAMP, WAMP чи XAMPP. З приводу пакету LAMP варто наголосити, що він буває досить проблемним. Час від часу з ним виникають ті чи інші труднощі, якщо щось пішло не так, то потрібно зносити усю систему та встановлювати і налаштовувати її з самого початку. Через таку нестабільність роботи витрачається багато часу. Тому краще ставити усі ці елементи окремо. Завдяки тому, що ми використовуємо Docker, ми зможемо з легкістю встановити та комбінувати усі елементи. У разі виникнення помилок зможемо перевстановити проблемний компонент, не зачепивши інші модулі. Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць. Для нашої платформи критично важливою є таблиця InnoDB, що підтримує транзакції на рівні окремих записів, повнотекстовий пошук (Full-Text) та нативну роботу з JSON-даними.

Одним із головних завдань є забезпечення безпеки даних. Тому найкращим рішенням буде використати Symfony Framework, адже Symfony наділений потужною системою захисту, саме тому він настільки популярний [2]. Symfony – це php-фреймворк, на базі якого можна розробляти складні веб-додатки. У фреймворка Symfony вельми потужна функціональність, продумана архітектура, а також розвинене співтовариство. Symfony – це високопродуктивний фреймворк, написаний на PHP. Фреймворк заснований на базі патерну Model-View-Controller і в його архітектурі дуже активно використовуються інші патерни об'єктно-орієнтованого програмування. Symfony має підтримку безлічі баз даних (MySQL, PostgreSQL, SQLite та інші PDO-сумісні СУБД). Інформація про реляційну базу даних в проєкті зв'язується з об'єктною моделлю за

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 20   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

допомогою ORM інструменту. Базова версія Symfony постачається з потужним ORM: Doctrine.

Для написання клієнтської частини ми будемо використовувати JavaScript у поєднанні з шаблонізатором Twig [13]. Головними їх перевагами є збільшення продуктивності та проста інтеграція з бекендом. Twig забезпечує базову безпеку (автоматичне екранування змінних для захисту від XSS) та має зручний синтаксис. Завдяки вбудованим асинхронним запитам ми маємо змогу оновлювати дані на сторінці без видимого перезавантаження. Для того, щоб пришвидшити розробку нашої платформи, будемо використовувати готові компоненти розмітки Bootstrap 5, адже він є найбільш популярним та зручним у світі CSS-фреймворком для користувачького інтерфейсу [12]. Bootstrap надає готові рішення для швидкої і досить простий веброзробки. Це велика бібліотека, де ключовою частиною є потужна система сіток (grid system) та набір адаптивних компонентів (модальні вікна, навігаційні панелі, кнопки).

Для того, щоб забезпечити динамічне оновлення сторінок без видимого перезавантаження, використаємо «Mercure» [6]. Mercure – це протокол, який дозволяє надсилати оновлення даних до веббраузерів та інших HTTP-клієнтів зручним, швидким, надійним та економічним способом. Особливо корисно публікувати асинхронні та в режимі реального часу оновлення ресурсів, що обслуговуються через веб-API, для реактивних веб-додатків. Також для реалізації безпосереднього голосового спілкування в платформі використовується WebRTC — технологія, що дозволяє передавати аудіо- та відеодані безпосередньо між браузерами (peer-to-peer) [7]. А для обробки Premium-підписок із підтримкою регулярних платежів (recurring billing) інтегровано платіжний шлюз Stripe API [10].

А для того, щоб правильно налаштувати серверну частину, а головне — лише один раз, будемо використовувати Docker. Завдяки чому зможемо швидко тиражувати наш проєкт на хмарній платформі Railway [16]. Docker – це відкрита платформа для розробки, доставки і експлуатації додатків. Docker розроблений для більш швидкого викладання ваших додатків. За допомогою Docker ви

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    | 21   |

можете відокремити ваш додаток від вашої інфраструктури і звертатися з інфраструктурою як з керованим додатком. Docker допомагає викладати ваш код швидше, швидше тестувати і зменшити час між написанням коду і запуском коду. Docker робить це за допомогою легковагової платформи контейнерної віртуалізації. У своєму ядрі Docker дозволяє запускати практично будь-який додаток, безпечно ізольований в контейнері. Безпечна ізоляція дозволяє вам запускати на одному хості багато контейнерів одночасно. Легка природа контейнера, який запускається без додаткового навантаження гіпервізора, дозволяє вам добиватися більшого від вашого апаратного забезпечення. Docker прекрасно підходить для організації циклу розробки. Docker дозволяє розробникам використовувати локальні контейнери з додатками і сервісами. Для забезпечення безперервної інтеграції та доставки (CI/CD) у проєкті налаштовано автоматизовані пайплайни. Зокрема, використовується GitHub Actions (.github/workflows/mirror.yml) для дзеркалювання репозиторію та GitLab CI (.gitlab-ci.yml) для тестування і подальших етапів розгортання [17].

Щодо технічних засобів, мінімальні вимоги для серверу включають наявність будь-якої Unix-подібної операційної системи (наприклад, Ubuntu 22.04 LTS), двоядерного процесора з тактовою частотою від 2.0 GHz, мінімум 2 Gb оперативної пам'яті (проте для стабільної роботи бази даних рекомендовано 4 Gb), а також не менше 20 Gb вільного дискового простору для серверного додатку та зберігання даних. Вимоги до клієнтської частини обмежуються наявністю сучасного веббраузера: підтримуються Google Chrome (не нижче версії 80.0) [20], Mozilla Firefox (від версії 88.0), Opera (від версії 51) та Safari (не нижче версії 14.0).

### Висновки до першого розділу

У процесі виконання першого розділу було проаналізовано предметну область та поставлено задачу на кваліфікаційну роботу. Визначено основні

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 22   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

проблеми, які повинен вирішувати програмний продукт: забезпечення ефективного пошуку напарників для онлайн-ігор.

Проведено дослідження аналогічних систем (GamerLink, Discord, GameTree, TeamFind) з технічної точки зору. Виявлено їхні архітектурні підходи та недоліки в реалізації real-time функцій. Обґрунтовано відмову від мікросервісної та SPA архітектур на користь перевіреного підходу MVC (фреймворк Symfony) з використанням технологій SSE (Mercure) та WebRTC для швидкої і надійної реалізації інтерактивних можливостей без надмірного ускладнення інфраструктури.

Обрано технологічний стек інструментів: PHP 8.2; СКБД MySQL 8.0 (обрано завдяки підтримці JSON, повнотекстового пошуку та реплікації); клієнтські технології Bootstrap 5 та JavaScript; хмарні сервіси Railway та Cloudinary. Сформовано структуру програмного комплексу та вимоги до апаратних засобів, що дозволяє перейти безпосередньо до етапу проектування системи.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 23   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

## РОЗДІЛ 2. ПРОЄКТУВАННЯ ВЕБПЛАТФОРМИ «GAMEFINDER»

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

Платформа «GameFinder» повинна вирішувати такі основні проблеми гравців: пошук надійних та відповідних за рівнем партнерів, відсутність зручного засобу комунікації до початку гри, неможливість оцінити потенційного напарника заздалегідь, а також відсутність інструментів для планування ігрових сесій. На платформі передбачено п'ять типів користувачів. Для детального відображення їхніх можливостей побудовано окремі діаграми варіантів використання (Use Case) для кожної ролі. Роль «Гість» представляє неавторизованого користувача, який має доступ до перегляду загальнодоступних сторінок платформи, таких як головна сторінка, лобі, лідерборд, події та каталог ігор. Крім того, гість має змогу створити новий обліковий запис або увійти в існуючий.



Рисунок 2.1 – Діаграма варіантів використання для ролі «Гість»

Основною аудиторією платформи є роль «Користувач». Цей авторизований користувач може заповнювати свій профіль, користуватися автоматичним підбором напарників, створювати та шукати лобі за допомогою системи фільтрів, спілкуватися в текстових та голосових чатах, оцінювати напарників після гри та підключати Steam для синхронізації ігрових досягнень.

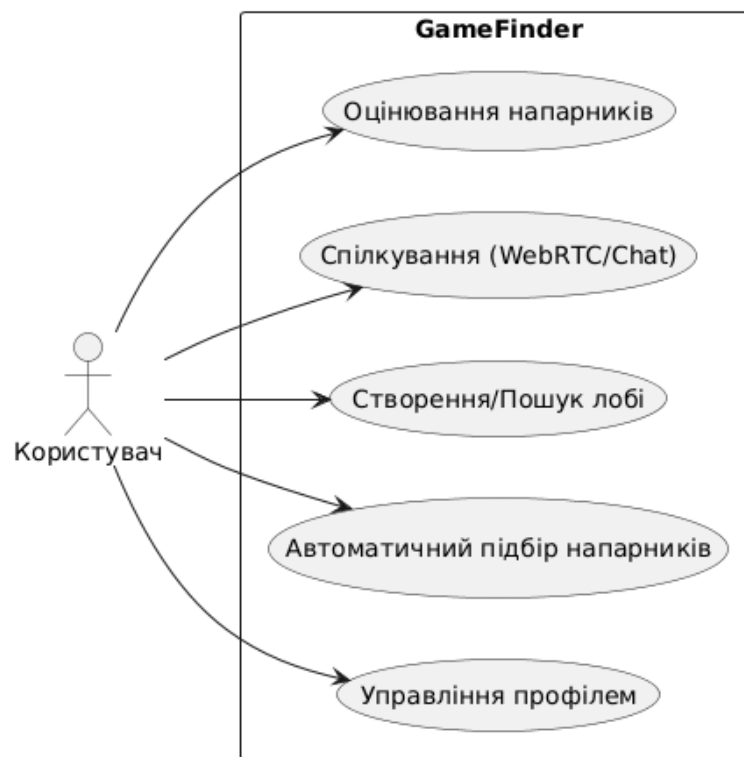


Рисунок 2.2 – Діаграма варіантів використання для ролі «Користувач»

Роль «Premium-користувач» представляє авторизованого гравця із розширеними можливостями у вигляді платної підписки. Ця роль повністю успадковує всі права звичайного користувача, але додатково отримує ексклюзивні косметичні предмети, такі як анімована рамка профілю та стікери для чату, а також функціональні переваги, зокрема пріоритет у підборі лобі.



Рисунок 2.3 – Діаграма варіантів використання для ролі «Premium-користувач»

Роль «Модератор» відповідає за підтримку порядку на платформі. Модератор має доступ до інструментів управління скаргами, завдяки яким може тимчасово блокувати порушників, видаляти неприйнятні повідомлення, примусово закривати лобі та ігрові події, видаляти неправдиві відгуки. Також він має доступ до модераторської статистики, що включає кількість скарг, користувачів та інформацію про останні дії в системі.



Рисунок 2.4 – Діаграма варіантів використання для ролі «Модератор»

Роль «Адміністратор» має найвищий та повний доступ до платформи, повністю успадковуючи всі права модератора. Адміністратор управляє глобальним каталогом ігор, що включає їх додавання, видалення та імпорт зі Steam, керує ролями користувачів системи та має доступ до базової глобальної статистики для перегляду загальної кількості користувачів та активних ігор.

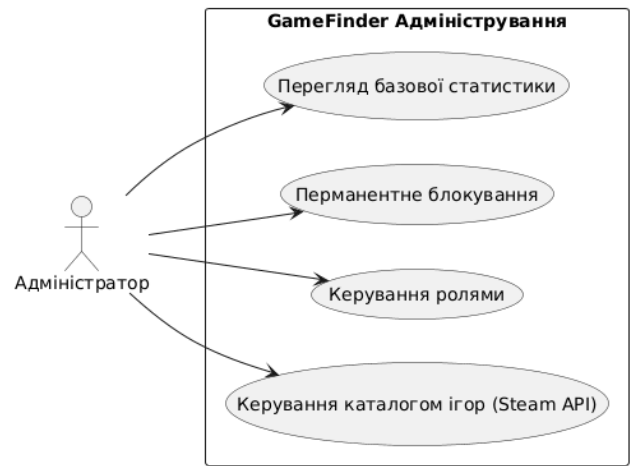


Рисунок 2.5 – Діаграма варіантів використання для ролі «Адміністратор»

Для наглядної взаємодії між елементами додатку було створено Діаграму варіантів використання (рис. 2.6).



Рисунок 2.6 – Діаграма взаємодії варіантів використання для різних ролей

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.7.

Таблиця 2.1.

Опис прецеденту «Автентифікація»

| Характеристика              | Опис                                                                                                                                         |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Автентифікація                                                                                                                               |
| Основні актори              | Гість                                                                                                                                        |
| Короткий опис               | Користувачі можуть зареєструватися через Email або увійти в систему за допомогою існуючого акаунту (пароль, Google, Discord через OAuth 2.0) |

Таблиця 2.2.

Опис прецеденту «Управління профілем та інтеграція»

| Характеристика              | Опис                                                                                                                                                                 |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Управління профілем                                                                                                                                                  |
| Основні актори              | Користувач                                                                                                                                                           |
| Короткий опис               | Користувачі можуть редагувати свої особисті дані, вибирати аватари та теми оформлення, а також підключати Steam ID для синхронізації та перегляду ігрових досягнень. |

Таблиця 2.3.

Опис прецеденту «Взаємодія з лобі та чатом»

| Характеристика              | Опис                                                                                                                                                                                                                 |
|-----------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Лобі та комунікація                                                                                                                                                                                                  |
| Основні актори              | Користувач, Premium-користувач                                                                                                                                                                                       |
| Короткий опис               | Гравці можуть створювати нові лобі, шукати існуючі за фільтрами, подавати заявки у приватні кімнати. Усередині лобі доступний текстовий чат, обмін файлами, використання стікерів та голосовий зв'язок через WebRTC. |

Таблиця 2.4.

## Опис прецеденту «Матчмейкінг та події»

| Характеристика              | Опис                                                                                                                                                                                                                                |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Матчмейкінг та планування подій                                                                                                                                                                                                     |
| Основні актори              | Користувач                                                                                                                                                                                                                          |
| Короткий опис               | Система забезпечує автоматичний підбір релевантних гравців, дозволяє оцінювати напарників, формує глобальний лідерборд.<br>Користувачі можуть створювати ігрові події у календарі, приєднуватися до них та видаляти створені події. |

Таблиця 2.5.

## Опис прецеденту «Premium-функції»

| Характеристика              | Опис                                                                                                                                                  |
|-----------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Управління Premium-підпискою                                                                                                                          |
| Основні актори              | Premium-користувач                                                                                                                                    |
| Короткий опис               | Користувачі можуть придбати платну підписку для отримання пріоритету при пошуку лобі, унікальних стікерів, тем оформлення та анімовану рамку профілю. |

Таблиця 2.6.

## Опис прецеденту «Модерація контенту»

| Характеристика              | Опис                                                                                                                                                                                                |
|-----------------------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Модерація                                                                                                                                                                                           |
| Основні актори              | Модератор                                                                                                                                                                                           |
| Короткий опис               | Забезпечення порядку на платформі шляхом перегляду скарг гравців, видалення неприйнятних повідомлень чи відгуків, примусового закриття лобі, видалення подій та видачі тимчасових банів порушникам. |

Таблиця 2.7.

## Опис прецеденту «Адміністрування платформи»

| Характеристика              | Опис                                                                                                                                                                                                 |
|-----------------------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Назва варіанту використання | Адміністрування                                                                                                                                                                                      |
| Основні актори              | Адміністратор                                                                                                                                                                                        |
| Короткий опис               | Глобальне управління системою, включаючи налаштування каталогу ігор, імпорт нових ігор через Steam API, керування ролями користувачів, аналіз загальної статистики та видачу перманентних блокувань. |

Основна ціль розробки платформи «GameFinder» полягає в створенні зручного онлайн-середовища, яке дозволяє гравцям швидко знаходити надійних напарників. Платформа спрямована на вирішення проблеми токсичності та несумісності, забезпечуючи ефективну комунікацію, оцінювання партнерів та автоматичний підбір за інтересами. Основне завдання полягає у формуванні здорової спільноти шляхом залучення геймерів до використання спеціалізованих інструментів організації ігрового процесу. Платформа дозволяє створювати детальні профілі, об'єднуватися в лобі, спілкуватися через чати та планувати події. Завдяки цьому створюються умови для комфортного ігрового досвіду та загального покращення соціальної взаємодії між гравцями.

Серед ключових бізнес-вимог першочерговим є збільшення залученості гравців (Retention), що досягається за рахунок системи рейтингу, досягнень та зручного пошуку напарників. Для забезпечення стабільного доходу передбачено монетизацію через інтеграцію платіжного шлюзу Stripe для продажу Premium-підписок, що надають косметичні привілеї [10]. Також важливою вимогою є мінімізація токсичності, яка реалізується шляхом формування здорової спільноти завдяки впровадженню системи репутації та ефективних інструментів модерації. Окрім цього, архітектура системи повинна бути масштабованою та витримувати велику кількість одночасних з'єднань (real-time chat, WebRTC), що дозволить платформі безперебійно зростати [7].

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 30   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Вимоги користувачів можна розділити відповідно до їхніх ролей. Адміністратор повинен мати можливості керування каталогом ігор (додавання, імпорт зі Steam), управління ролями, видачі перманентних блокувань та перегляду глобальної статистики. Модератор забезпечує порядок, обробляючи скарги, видаляючи неприйнятні повідомлення і відгуки, блокуючи порушників та закриваючи лобі. Premium-користувач очікує пріоритет під час підбору лобі, використання ексклюзивних тем оформлення та стікерів, а також управління підпискою. Звичайний користувач потребує безпечної авторизації (Email, OAuth) [19], редагування профілю, синхронізації зі Steam, а також створення лобі, управління подіями, друзями та залишення відгуків після гри. Гість має обмежуватися переглядом публічних сторінок та доступом до функцій реєстрації і входу [18].

Серед основних функціональних вимог виділено забезпечення надійної реєстрації та авторизації (Email, OAuth), а також повноцінне управління профілями. Важливою складовою є алгоритмічна система підбору напарників (Matchmaking) на основі ігрових вподобань, поточного рейтингу та мови спілкування [24]. Система повинна підтримувати створення, пошук та управління лобі із розширеними фільтрами. Для забезпечення комунікації вимагається реалізація чатів у реальному часі (текстове листування та голосовий зв'язок WebRTC). Інші функції включають систему відгуків, інтеграцію Steam API для імпорту досягнень, календар подій та розсилку автоматичних сповіщень [9]. Також передбачена наявність адміністративної панелі та можливість придбання Premium-підписки.

Нефункціональні вимоги визначають критерії якості: адаптивний інтерфейс, продуктивність (час відповіді до 0,5 секунди), стабільна робота чату без затримок. Системні вимоги до клієнтської частини: сучасний браузер (Chrome 90+, Firefox 80+), пристрої від 2 ГБ ОЗУ, 1.5 ГГц процесор та мікрофон. Серверна частина вимагає PHP 8.2 (Symfony 7) та бази даних MySQL 8.0+ [1]. Для роботи функцій реального часу необхідний налаштований сервер Mercure Hub та інтеграція зі сховищем Cloudinary [6].

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 31   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Аналіз функціональних вимог дозволив виділити основні сутності доменної моделі (рівень Entity), що забезпечать програмну реалізацію платформи «GameFinder». На рисунку 2.7 наведено діаграму класів, яка використовується для її роботи.

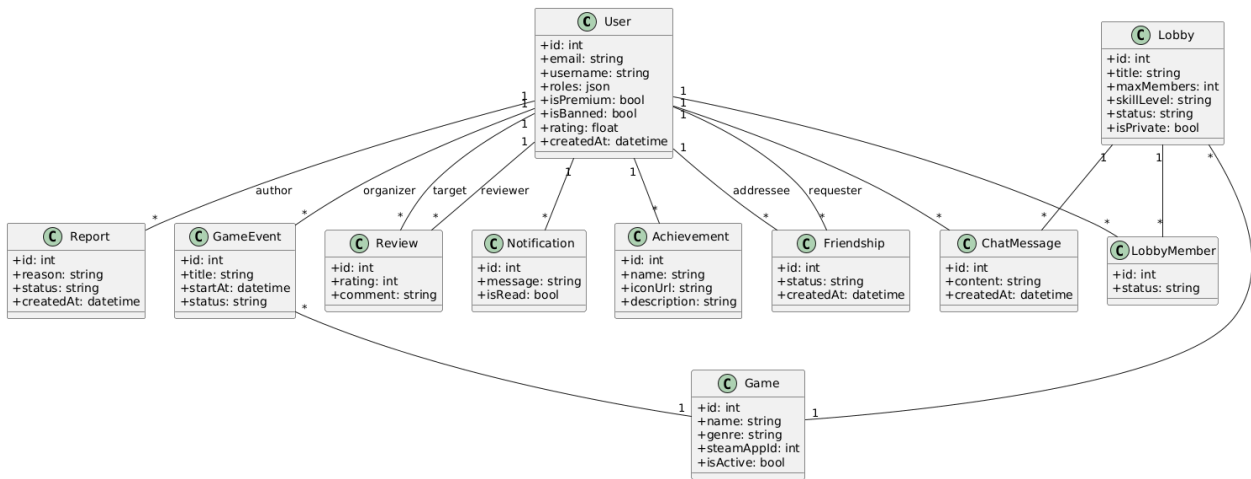


Рисунок 2.7 – Діаграма класів доменної моделі (рівень Entity)

Опишемо основні сутності, зображені на діаграмі класів. Сутність User представляє авторизованого гравця і містить базові атрибути (ідентифікатор, електронну пошту, ім'я користувача, ролі) та специфічні ігрові дані (наявність Premium-підписки, статус блокування, рейтинг, дата реєстрації), що дозволяють ідентифікувати користувача, визначати його рівень доступу та соціальний рейтинг. Сутність Game створена для збереження базової інформації про відеоігри, доступні на платформі, включаючи назву, жанр, ідентифікатор у Steam та статус активності, що використовується для категоризації ігрових сесій. Для об'єднання гравців зі схожими навичками використовується сутність Lobby, яка представляє ігрову кімнату для пошуку напарників та містить параметри максимальної кількості учасників, необхідного рівня гри, статусу і приватності. Допоміжною є сутність LobbyMember, яка відповідає за відстеження поточного статусу та ролі конкретних учасників у створеному лобі. Окрім лобі, передбачена сутність GameEvent, яка описує заплановані кіберспортивні події або турніри із зазначенням конкретного часу їх проведення та поточного статусу.

Для забезпечення комунікації реалізована сутність ChatMessage, яка представляє повідомлення, відправлені учасниками у чаті. Оцінювання напарників після спільної гри фіксується за допомогою сутності Review, що безпосередньо впливає на рейтинг гравців. Інформування користувачів про важливі події відбувається через сутність Notification, яка зберігає системні або персональні сповіщення. Крім цього, архітектура включає ще три важливі сутності: Friendship для встановлення соціальних зв'язків між користувачами (списки друзів), Achievement для синхронізації та фіксації індивідуальних ігрових досягнень, а також Report для реєстрації скарг на неприйнятний контент. Ця об'єктно-орієнтована багатоваріантова структура з 11 ключових сутностей забезпечує надійне зберігання інформації, високу гнучкість, цілісність даних та підтримку всього функціоналу системи завдяки чітко визначеним зв'язкам.

Шар бізнес-логіки системи реалізується через набір класів-сервісів (Services), які інкапсулюють у собі складні алгоритми обробки даних та зовнішні інтеграції. На рисунку 2.8 наведено діаграму класів сервісного шару.

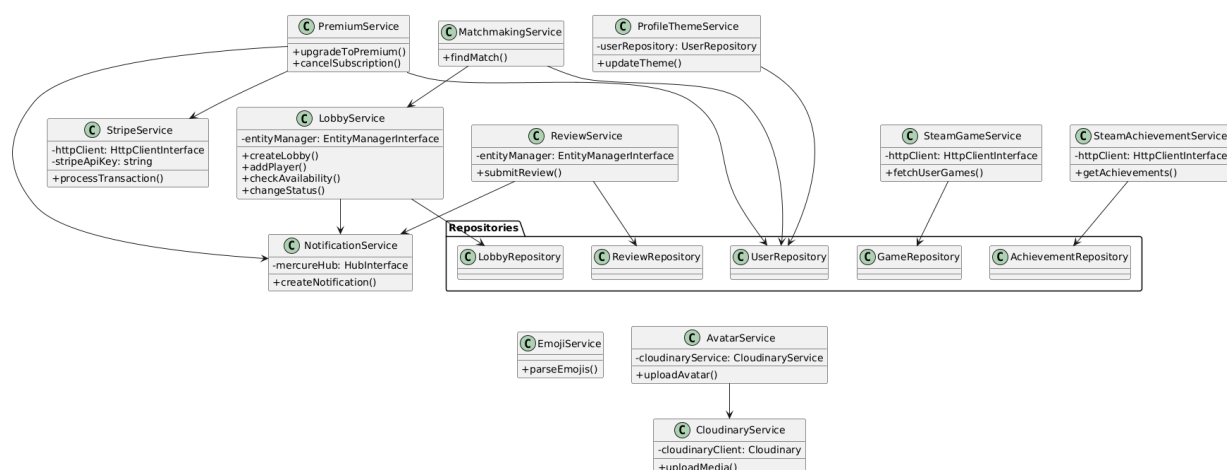


Рисунок 2.8 – Діаграма класів бізнес-логіки (рівень Service)

Основні класи діаграми бізнес-логіки можна описати наступним чином. Клас LobbyService відповідає за життєвий цикл ігрових кімнат: метод createLobby() відповідає за створення нового лобі, addPlayer() — за додавання гравців, checkAvailability() перевіряє наявність вільних місць, а changeStatus()

змінює поточний статус лобі. Клас MatchmakingService забезпечує функціонал автоматичного підбору гравців, використовуючи метод findMatch() для аналізу рейтингу та пошуку команди. Управління підписками реалізовано в PremiumService за допомогою методів upgradeToPremium() та cancelSubscription(). Для взаємодії із зовнішнім платіжним шлюзом Stripe [10] створено StripeService з методом processTransaction().

Інтеграція зі Steam API інкапсульована у класах SteamGameService (метод fetchUserGames() для імпорту ігор) та SteamAchievementService (метод getAchievements() для отримання досягнень). Клас NotificationService використовує метод createNotification() для прив'язки сповіщень до користувачів. Для роботи з медіафайлами створено CloudinaryService (uploadMedia()) та AvatarService (uploadAvatar() для передачі аватарів до Cloudinary [11]). Обробка емодзі в чаті виконується через EmojiService (parseEmojis()), а кастомізація профілю — через ProfileThemeService (updateTheme()). Процес оцінювання напарників управляється класом ReviewService за допомогою методу submitReview().

Шар маршрутизації та обробки запитів платформи «GameFinder» реалізовано за допомогою класів-контролерів (Controllers). Цей рівень забезпечує прийом HTTP-запитів, валідацію вхідних даних та формування HTTP-відповідей. На рисунку 2.9 наведено діаграму класів рівня контролерів.

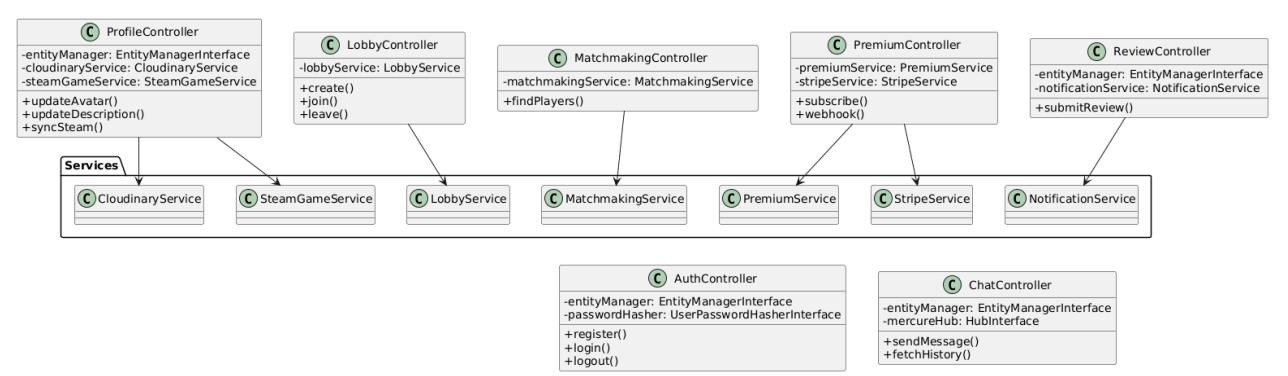


Рисунок 2.9 – Діаграма класів маршрутизації та обробки запитів (рівень Controller)

Опишемо основні класи діаграми контролерів. Клас AuthController відповідає за процеси ідентифікації: він містить методи register() для реєстрації нових користувачів, login() для авторизації та logout() для виходу з системи. Взаємодія з ігровими кімнатами здійснюється через LobbyController, де методи create(), join() та leave() обробляють запити на створення лобі, приєднання до нього та вихід відповідно. За обмін повідомленнями відповідає ChatController з методами sendMessage() для збереження нових повідомлень та fetchHistory() для вибірки історії чату.

Класи, пов'язані з підписками та профілями, включають PremiumController (методи subscribe() та webhook() для обробки платежів від Stripe) та ProfileController (методи updateAvatar(), updateDescription() та syncSteam() для управління даними користувача). Запити на автоматичний пошук напарників обробляє MatchmakingController через метод findPlayers(), а система оцінювання керується класом ReviewController за допомогою методу submitReview().

Загальна архітектура системи базується на класичному паттерні багатошарової архітектури, де чітко розділені обов'язки між шарами представлення, бізнес-логіки та доступу до даних. На рисунку 2.10 наведено загальну узагальнену діаграму класів, що демонструє ієрархію та потік даних між ключовими компонентами системи.

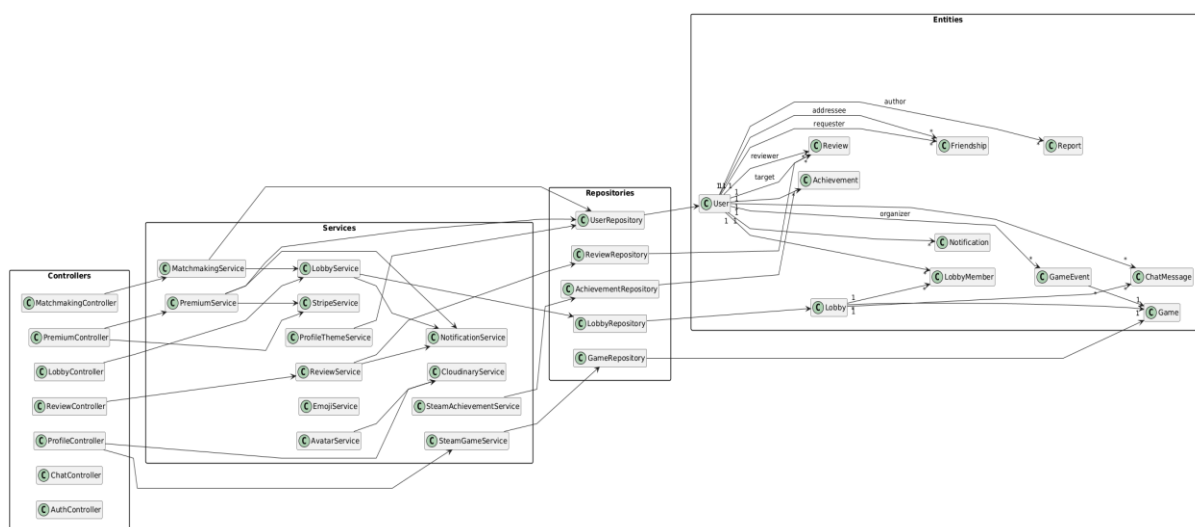


Рисунок 2.10 – Загальна діаграма класів архітектури системи

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 35   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Ця об'єктно-орієнтована багатоваріантова структура забезпечує надійне зберігання інформації, високу гнучкість, цілісність даних та легке масштабування всього функціоналу завдяки жорсткому дотриманню принципу єдиної відповідальності (SRP) та впровадженню залежностей (Dependency Injection).

## 2.2. Розробка бази даних та математичних алгоритмів системи

Для збереження даних у додатку використовується MySQL – система управління реляційними базами даних з відкритим вихідним кодом [5]. MySQL забезпечує надійне зберігання структурованих даних, підтримує складні запити, транзакції та забезпечує цілісність даних через систему зовнішніх ключів та обмежень. Додаток розгорнуто на хмарній платформі Railway, яка надає керований екземпляр MySQL як сервіс [16]. Railway автоматично забезпечує масштабування, резервне копіювання та моніторинг бази даних. Взаємодія з базою даних здійснюється через ORM Doctrine, яка дозволяє працювати з даними на рівні об'єктів PHP, автоматично генерувати SQL-запити та забезпечувати міграції схеми бази даних [1].

Таблиця 2.8.

Таблиці бази даних

| Назва таблиці | Призначення                                 |
|---------------|---------------------------------------------|
| User          | таблиця, яка зберігає дані користувачів     |
| Game          | таблиця, яка зберігає інформацію про ігри   |
| Lobby         | таблиця, яка зберігає лобі для спільної гри |
| LobbyMember   | таблиця, яка зберігає учасників лобі        |
| ChatMessage   | таблиця, яка зберігає повідомлення чату     |
| Notification  | таблиця, яка зберігає сповіщення            |

продовж. табл. 2.8

|             |                                                        |
|-------------|--------------------------------------------------------|
| Friendship  | таблиця, яка зберігає зв'язки дружби між користувачами |
| Review      | таблиця, яка зберігає відгуки про користувачів         |
| Achievement | таблиця, яка зберігає досягнення користувачів          |
| Report      | таблиця, яка зберігає скарги на контент                |
| GameEvent   | таблиця, яка зберігає ігрові події                     |

Опис кожної з таблиць бази даних наведений у таблицях 2.9 – 2.19.

Таблиця 2.9.

Таблиця «User»

| Поле      | Тип          | Призначення                      |
|-----------|--------------|----------------------------------|
| id        | int          | Ідентифікатор                    |
| email     | varchar(180) | Email користувача (унікальний)   |
| username  | varchar(50)  | Нікнейм користувача (унікальний) |
| roles     | json         | Ролі користувача                 |
| password  | varchar      | Хешований пароль                 |
| bio       | varchar(500) | Опис профілю                     |
| avatar    | varchar(255) | Посилання на аватар              |
| city      | varchar(100) | Місто користувача                |
| language  | varchar(10)  | Мова користувача                 |
| age       | int          | Вік користувача                  |
| googleId  | varchar(255) | Ідентифікатор Google OAuth       |
| discordId | varchar(255) | Ідентифікатор Discord            |

продовж. табл. 2.9

|                  |              |                             |
|------------------|--------------|-----------------------------|
| steamId          | varchar(255) | Ідентифікатор Steam         |
| createdAt        | datetime     | Дата реєстрації             |
| lastLoginAt      | datetime     | Дата останнього входу       |
| isPremium        | boolean      | Статус преміум-підписки     |
| premiumExpiresAt | datetime     | Дата закінчення преміуму    |
| isVerified       | boolean      | Статус верифікації email    |
| isBanned         | boolean      | Статус блокування           |
| bannedUntil      | datetime     | Дата закінчення блокування  |
| profileTheme     | varchar(30)  | Тема оформлення профілю     |
| rating           | float        | Рейтинг користувача         |
| totalReviews     | int          | Загальна кількість відгуків |

Таблиця 2.10.

Таблиця «Game»

| Поле        | Тип          | Призначення                   |
|-------------|--------------|-------------------------------|
| id          | int          | Ідентифікатор                 |
| name        | varchar(100) | Назва гри                     |
| slug        | varchar(255) | URL-дружній ідентифікатор     |
| genre       | varchar(50)  | Жанр гри                      |
| description | varchar(500) | Опис гри                      |
| imageUrl    | varchar(255) | Посилання на зображення       |
| maxPlayers  | int          | Максимальна кількість гравців |
| steamAppId  | int          | Ідентифікатор гри в Steam     |
| isActive    | boolean      | Статус активності гри         |

Таблиця «Lobby»

| Поле        | Тип          | Призначення                                                 |
|-------------|--------------|-------------------------------------------------------------|
| id          | int          | Ідентифікатор                                               |
| title       | varchar(150) | Назва лобі                                                  |
| description | varchar(500) | Опис лобі                                                   |
| maxMembers  | int          | Максимальна кількість учасників                             |
| skillLevel  | varchar(20)  | Рівень навичок (beginner, intermediate, advanced, pro, any) |
| minAge      | int          | Мінімальний вік учасників                                   |
| maxAge      | int          | Максимальний вік учасників                                  |
| language    | varchar(10)  | Мова спілкування                                            |
| city        | varchar(100) | Місто для локального пошуку                                 |
| status      | varchar(20)  | Статус лобі (open, full, in_game, closed)                   |
| isPrivate   | boolean      | Приватність лобі                                            |
| voiceChat   | boolean      | Наявність голосового чату                                   |
| createdAt   | datetime     | Дата створення                                              |
| scheduledAt | datetime     | Запланований час гри                                        |
| game_id     | int          | Зовнішній ключ на таблицю Game                              |
| owner_id    | int          | Зовнішній ключ на таблицю User                              |

Таблиця 2.12.

Таблиця «LobbyMember»

| Поле     | Тип         | Призначення                                 |
|----------|-------------|---------------------------------------------|
| id       | int         | Ідентифікатор                               |
| status   | varchar(20) | Статус участі (pending, accepted, rejected) |
| role     | varchar(20) | Роль в лобі (owner, member)                 |
| joinedAt | datetime    | Дата приєднання                             |
| lobby_id | int         | Зовнішній ключ на таблицю Lobby             |
| user_id  | int         | Зовнішній ключ на таблицю User              |

Таблиця 2.13.

Таблиця «Notification»

| Поле      | Тип          | Призначення                                                               |
|-----------|--------------|---------------------------------------------------------------------------|
| id        | int          | Ідентифікатор                                                             |
| type      | varchar(50)  | Тип сповіщення (lobby_invite, friend_request, game_start, review, system) |
| message   | varchar(500) | Текст сповіщення                                                          |
| link      | varchar(255) | Посилання для переходу                                                    |
| isRead    | boolean      | Статус прочитання                                                         |
| createdAt | datetime     | Дата створення                                                            |
| user_id   | int          | Зовнішній ключ на таблицю User                                            |

Таблиця 2.14.

Таблиця «ChatMessage»

| Поле          | Тип          | Призначення                                 |
|---------------|--------------|---------------------------------------------|
| id            | int          | Ідентифікатор                               |
| content       | text         | Вміст повідомлення                          |
| type          | varchar(20)  | Тип повідомлення (text, image, file)        |
| attachmentUrl | varchar(255) | Посилання на вкладення                      |
| isPrivate     | boolean      | Приватне повідомлення                       |
| isRead        | boolean      | Статус прочитання                           |
| createdAt     | datetime     | Дата відправлення                           |
| sender_id     | int          | Зовнішній ключ на таблицю User (відправник) |
| lobby_id      | int          | Зовнішній ключ на таблицю Lobby             |
| event_id      | int          | Зовнішній ключ на таблицю GameEvent         |
| recipient_id  | int          | Зовнішній ключ на таблицю User (отримувач)  |

Таблиця 2.15.

Таблиця «FriendShip»

| Поле       | Тип         | Призначення                                          |
|------------|-------------|------------------------------------------------------|
| id         | int         | Ідентифікатор                                        |
| status     | varchar(20) | Статус дружби (pending, accepted, rejected, blocked) |
| createdAt  | datetime    | Дата створення запиту                                |
| acceptedAt | datetime    | Дата прийняття запиту                                |

продовж. табл. 2.15

|              |     |                                               |
|--------------|-----|-----------------------------------------------|
| requester_id | int | Зовнішній ключ на таблицю User (хто надіслав) |
| receiver_id  | int | Зовнішній ключ на таблицю User (хто отримав)  |

Таблиця 2.16.

Таблиця «Review»

| Поле       | Тип          | Призначення                            |
|------------|--------------|----------------------------------------|
| id         | int          | Ідентифікатор                          |
| isPositive | boolean      | Позитивний чи негативний відгук        |
| comment    | varchar(500) | Текст відгуку                          |
| createdAt  | datetime     | Дата створення                         |
| author_id  | int          | Зовнішній ключ на таблицю User (автор) |
| target_id  | int          | Зовнішній ключ на таблицю User (ціль)  |
| lobby_id   | int          | Зовнішній ключ на таблицю Lobby        |

Таблиця 2.17.

Таблиця «Achievement»

| Поле        | Тип          | Призначення         |
|-------------|--------------|---------------------|
| id          | int          | Ідентифікатор       |
| name        | varchar(200) | Назва досягнення    |
| description | varchar(500) | Опис досягнення     |
| iconUrl     | varchar(255) | Посилання на іконку |

продовж. табл. 2.17

|                    |              |                                  |
|--------------------|--------------|----------------------------------|
| steamAchievementId | varchar(100) | Ідентифікатор досягнення в Steam |
| unlockedAt         | datetime     | Дата отримання                   |
| user_id            | int          | Зовнішній ключ на таблицю User   |
| game_id            | int          | Зовнішній ключ на таблицю Game   |

Таблиця 2.18.

Таблиця «Report»

| Поле               | Тип          | Призначення                                     |
|--------------------|--------------|-------------------------------------------------|
| id                 | int          | Ідентифікатор                                   |
| reason             | varchar(500) | Причина скарги                                  |
| status             | varchar(20)  | Статус скарги (pending, reviewed, resolved)     |
| moderatorNote      | varchar(500) | Нотатка модератора                              |
| createdAt          | datetime     | Дата створення                                  |
| resolvedAt         | datetime     | Дата вирішення                                  |
| reporter_id        | int          | Зовнішній ключ на таблицю User (хто подав)      |
| reportedUser_id    | int          | Зовнішній ключ на таблицю User (на кого скарга) |
| reportedMessage_id | int          | Зовнішній ключ на таблицю ChatMessage           |
| reportedReview_id  | int          | Зовнішній ключ на таблицю Review                |

продовж. табл. 2.18

|               |     |                                              |
|---------------|-----|----------------------------------------------|
| resolvedBy_id | int | Зовнішній ключ на таблицю User (хто вирішив) |
|---------------|-----|----------------------------------------------|

Таблиця 2.19.

Таблиця «GameEvent»

| Поле            | Тип          | Призначення                                          |
|-----------------|--------------|------------------------------------------------------|
| id              | int          | Ідентифікатор                                        |
| title           | varchar(200) | Назва події                                          |
| description     | text         | Опис події                                           |
| startAt         | datetime     | Дата та час початку                                  |
| endAt           | datetime     | Дата та час закінчення                               |
| maxParticipants | int          | Максимальна кількість учасників                      |
| status          | varchar(20)  | Статус події (planned, active, completed, cancelled) |
| createdAt       | datetime     | Дата створення                                       |
| organizer_id    | int          | Зовнішній ключ на таблицю User                       |
| game_id         | int          | Зовнішній ключ на таблицю Game                       |

Зв'язки між таблицями реалізовані через зовнішні ключі (Foreign Keys), які забезпечують цілісність посилань на рівні СУБД. При створенні запису MySQL перевіряє існування пов'язаних записів у відповідних таблицях – наприклад, при додаванні лобі перевіряється наявність вказаного користувача та гри [5]. Для зв'язку "багато до багатьох" між подіями та учасниками використовується

проміжна таблиця `event_participants`. Унікальні обмеження запобігають дублюванню даних: пара (`requester_id`, `receiver_id`) у таблиці `Friendship` та трійка (`author_id`, `target_id`, `lobby_id`) у таблиці `Review`.

Таким чином, розроблена база даних складається з одинадцяти взаємопов'язаних таблиць та повністю забезпечує функціонування системи – від реєстрації користувачів та інтеграції із зовнішніми сервісами до пошуку лобі, комунікації в чаті, соціальної взаємодії та модерації контенту.

Важливою складовою платформи є алгоритм інтелектуального підбору напарників (`Matchmaking`), який забезпечує автоматизований пошук оптимальних партнерів для гравця [26]. Цей алгоритм обчислює загальний бал сумісності (`Score`) для кожного потенційного кандидата на основі ряду критеріїв за адитивною моделлю. Базова логіка підбору відразу виключає з пошуку самого користувача, а також користувачів, які вже є у списку друзів або мають заблокований акаунт. Нарахування балів відбувається за наступними правилами: наявність опису профілю (`Bio`) додає 5 балів, наявність аватара — 3 бали; збіг мови спілкування додає 30 балів; збіг міста проживання (геолокація) додає 25 балів. Схожий вік також відіграє роль: різниця у віці до 3 років додає 25 балів, до 5 років — 20 балів, до 10 років — 10 балів. За кожну спільну гру нараховується 15 балів. Якщо у кандидата 3 або більше відгуків, додається бонус до 20 балів пропорційно його рейтингу довіри. Інтеграція зі `Steam` додає 5 балів, а активність гравця (наявність хоча б одного лобі) додає ще 10 балів. Кандидати з оцінкою більше 0 сортуються за спаданням балу. Додатково для кожного знайденого напарника формується список причин збігу (наприклад, «Спільні ігри», «Та сама мова»), які відображаються у користувацькому інтерфейсі [28].

Для забезпечення чесності та прозорості оцінювання гравців розроблено відповідну бізнес-логіку системи відгуків та рейтингу, яка підпорядковується суворим правилам. По-перше, виконується перевірка участі у спільному лобі: користувач може залишити відгук іншому гравцеві лише тоді, коли вони були учасниками одного і того ж ігрового лобі. По-друге, встановлено вимоги до статусу та тривалості лобі: оцінювання стає доступним виключно після того, як

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 45   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

лобі переходить у статус завершеного. Мінімальний час існування лобі становить 1 хвилину для запобігання накрутці. По-третє, діє захист від дублювання та самооцінювання: заборонено залишати відгуки самому собі, а для іншого гравця дозволяється мати лише один активний відгук (при повторному оцінюванні він оновлюється). Відгуки є бінарними (позитивний або негативний), а загальний рейтинг гравця розраховується як відсоток позитивних відгуків від загальної кількості. Отримане значення рейтингу округлюється до одного десяткового знака та зберігається у базі даних профілю користувача.

### 2.3. Проектування алгоритмів роботи вебплатформи

Додаток розроблявся з урахуванням сучасних стандартів UX/UI-дизайну, що робить його максимально простим у використанні та гарантує, що всі функціональні можливості залишаються інтуїтивно зрозумілими навіть для нових користувачів. Головною метою при проектуванні архітектури взаємодії було створення безшовного досвіду, де кожна дія гравця логічно впливає з попередньої, формуючи цілісний алгоритм роботи з платформою. Цей узагальнений алгоритм детально відображено на рисунку 2.11 у вигляді UML-діаграми активності, яка візуалізує переходи між різними станами системи залежно від ініційованих подій.

В основі бізнес-логіки платформи лежить сувора рольова модель управління доступом (Role-Based Access Control). Залежно від поточної ролі користувача, алгоритм його взаємодії з системою суттєво відрізняється, надаючи або обмежуючи доступ до певних модулів. Найвищий рівень привілеїв у системі належить адміністратору. Після проходження процедури безпечної авторизації адміністратор автоматично перенаправляється до спеціалізованої адміністративної панелі (Dashboard). Цей простір слугує єдиним центром управління всією екосистемою GameFinder. В адміністративній панелі реалізовано збір та візуалізацію глобальної статистики: поточну кількість зареєстрованих гравців, відсоток активних користувачів (MAU/DAU), загальну кількість створених лобі в реальному часі, а також агреговані дані щодо ігрових

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    | 46   |

подій та надісланих скарг. Адміністратор отримує повні та ексклюзивні можливості управління глобальним каталогом ігор. Для підтримки актуальності бібліотеки реалізована пряма інтеграція зі Steam API, завдяки чому адміністратор може виконувати пошук нових ігрових тайтлів безпосередньо в базі Steam та здійснювати масовий імпорт їх метаданих (назва, жанр, офіційна обкладинка). У випадку необхідності ігри можна додавати вручну або повністю видаляти з системи, причому останнє супроводжується каскадним оновленням пов'язаних лобі. Крім управління контентом, адміністратор виступає верховним арбітром системи: він має повноваження застосовувати глобальне блокування (перманентний бан) до акаунтів, які критично порушили правила платформи, а також розблоковувати їх у разі задоволення апеляцій. Важливою функцією є управління кадрами – лише адміністратор здатний розширювати команду модераторів або призначати нових адміністраторів шляхом зміни ролей безпосередньо в базі даних користувачів. Слід зазначити, що адміністратор за замовчуванням повністю успадковує всі повноваження модератора, маючи змогу безпосередньо брати участь у розборі складних конфліктів.

У свою чергу, алгоритм роботи модератора зосереджений виключно на підтримці здорового мікроклімату спільноти та забезпеченні дотримання користувацької угоди. Після авторизації модератор отримує доступ до панелі модерації, яка функціонує за принципом тікет-системи (Ticket System). Його основне щоденне завдання полягає у швидкому реагуванні на скарги (репорти), що генеруються автоматичними тригерами або надходять безпосередньо від гравців платформи через систему репортування в профілях чи чатах. Аналізуючи контекст порушення, модератор приймає рішення та застосовує дисциплінарні заходи, найпоширенішим з яких є тимчасове обмеження доступу (Time-out бан). Система дозволяє гнучко вказувати тривалість такого бану (наприклад, 24 години, тиждень або місяць) залежно від тяжкості порушення. Окрім обмеження доступу до акаунтів, модератор здійснює премодерацію та постмодерацію користувацького контенту. Він має необхідний інструментарій для примусового видалення образливих чи спамних повідомлень із текстових чатів, а також

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 47   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

видалення неправдивих або токсичних відгуків, що штучно занижують репутацію гравців. Для боротьби з деструктивними кімнатами модератор має право ініціювати примусове закриття (Force Close) лобі, якщо в ньому фіксується масова токсична поведінка, або скасовувати заплановані ігрові події, якщо їхній опис чи мета не відповідають політиці спільноти.

Для неавторизованого користувача (гостя) передбачено максимально відкритий алгоритм ознайомлення з платформою, що є критично важливим для органічного залучення нової аудиторії (SEO-оптимізації). Гість має безперешкодний доступ до перегляду ключових загальнодоступних сторінок платформи. Він може досліджувати головну сторінку, переглядати список відкритих лобі (без можливості бачити приватні чати), слідкувати за лідерами глобального рейтингу (Leaderboard), переглядати розклад майбутніх ігрових подій та шукати ігри у загальному каталозі. Такий підхід створює ефект прозорості та дозволяє потенційному користувачу оцінити функціонал системи ще до передачі персональних даних. З будь-якої публічної сторінки інтерфейс пропонує гостю логічно завершити ознайомлення переходом до реєстрації. Система підтримує як класичну реєстрацію через форму (з обов'язковою валідацією Email та пароля), так і миттєвий вхід через сторонні провайдери (Discord, Google) за протоколом OAuth 2.0, що значно підвищує конверсію нових відвідувачів.

Алгоритм роботи звичайного авторизованого користувача охоплює основний та наймасштабніший функціонал платформи. Одразу після створення акаунту користувач отримує доступ до індивідуального профілю, який він може гнучко налаштовувати, заповнюючи розділ "Про себе", встановлюючи аватар та вибираючи улюблені ігрові жанри. Важливою частиною налаштування профілю є можливість інтеграції зі Steam, що дозволяє автоматично синхронізувати власні ігрові досягнення, поточну бібліотеку ігор та рівень акаунта напряму на платформу GameFinder. З точки зору ігрового процесу, користувач має повну свободу дій: він може вручну шукати напарників шляхом створення власних лобі з унікальними правилами або приєднуватися до вже існуючих відкритих кімнат

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 48   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

за допомогою розвиненої панелі багатфакторних фільтрів (пошук за конкретною грою, мовою спілкування чи бажаним рівнем навичок). Якщо ж користувач не бажає витратити час на ручний пошук, він запускає систему інтелектуального підбору (Matchmaking). Алгоритм матчмейкінгу виконує складний математичний аналіз бази даних гравців, вираховуючи коефіцієнт сумісності за десятками критеріїв, та формує персоналізовану стрічку найбільш підходящих кандидатів, з якими можна миттєво розпочати комунікацію. Для організації зв'язку всередині лобі система пропонує багатофункціональний текстовий чат із підтримкою відправки медіафайлів (скріншотів, гіфок) через інтеграцію з Cloudinary, а також прямий голосовий зв'язок на базі технології WebRTC, що усуває необхідність використання сторонніх додатків на кшталт Discord. Для довгострокового планування гравці можуть переглядати загальнодоступний календар ігрових подій, реєструватись як учасники в турнірах або створювати власні події, об'єднуючи великі групи людей на певну дату та час. Після завершення кожної спільної ігрової сесії алгоритм зобов'язує або пропонує учасникам залишити взаємні відгуки. Ці оцінки миттєво обробляються системою і впливають на показник Trust Factor (рівень довіри), що напряду формує глобальний рейтинг користувача і впливає на його майбутню видимість у системі підбору. Додатково користувач може керувати своїми соціальними зв'язками, надсилаючи та приймаючи запити на додавання у друзі, та підтримувати приватний текстовий зв'язок через особисті повідомлення.

Окремим сценарієм є алгоритм роботи користувача зі статусом Premium (платна підписка). Придбавши підписку через платіжний шлюз Stripe, гравець миттєво розширює свої базові можливості. Ключовою алгоритмічною перевагою Premium-статусу є зміна ваги користувача в базі даних: такі акаунти отримують абсолютний пріоритет у черзі відображення в системі пошуку лобі та інтелектуальному підборі, що значно прискорює пошук ідеального напарника. З візуальної точки зору Premium-користувач отримує доступ до панелі розширеної кастомізації профілю, яка дозволяє обирати ексклюзивні CSS-теми оформлення (динамічна зміна кольорів інтерфейсу) та застосовувати анімовані рамки для

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 49   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

свого аватара, що виділяє його на фоні стандартних гравців. Під час спілкування в чаті підписники отримують розширені набори унікальних емодзі та виділяються спеціальним статусним бейджем PRO, який відображається поряд із нікнеймом на всіх сторінках системи.

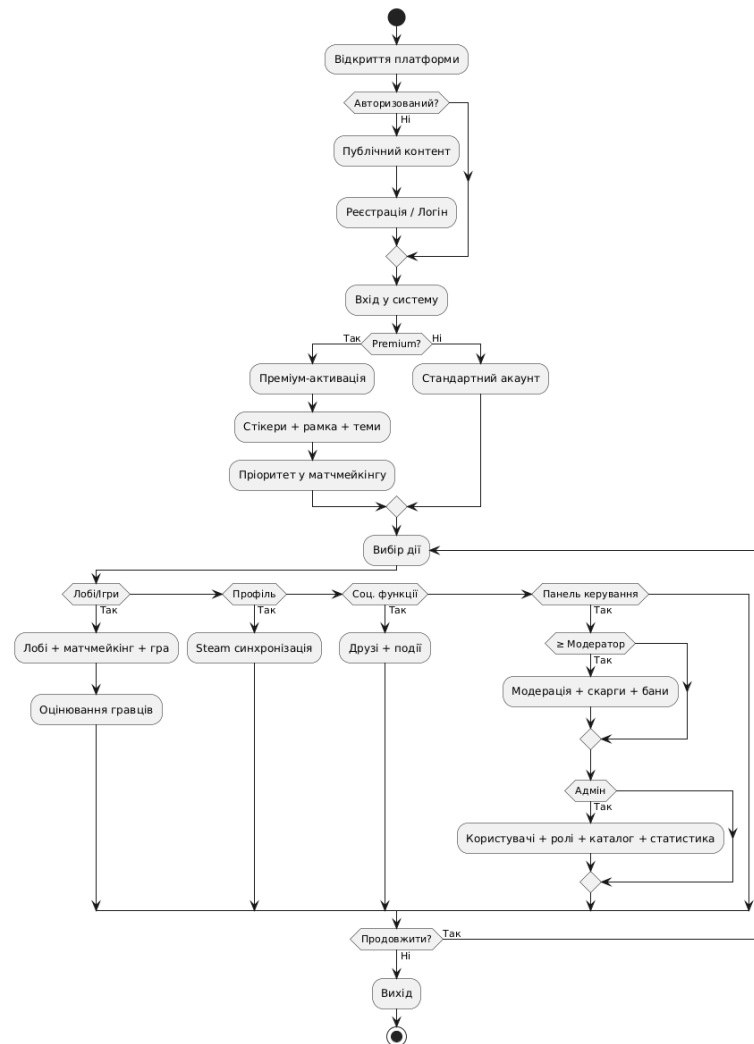


Рисунок 2.11 – Діаграма активності

## 2.4. Опис реалізації функціоналу вебплатформи

Розглянемо реалізацію ключового функціоналу. При проєктуванні системи активно використовувалися структурні патерни програмування Repository (через Doctrine ORM), Service Layer та Dependency Injection [4].

Базовою сутністю системи є ігрове лобі (кімната), яке дозволяє гравцям об'єднуватися для спільних ігрових сесій. Першим етапом є створення лобі. Користувач, маючи зареєстрований акаунт, заповнює форму створення кімнати. Процес створення лобі обробляється у контролері `LobbyController`. Замість написання "товстих контролерів", де бізнес-логіка змішується з обробкою HTTP-запитів, контролер делегує збереження абстракції `EntityManager` (рис. 2.12).

```
#[Route('/lobby/create', name: 'app_lobby_create')]
public function create(Request $request, EntityManagerInterface $em):
Response
{
    $lobby = new Lobby();
    $lobby->setHost($this->getUser());
    $lobby->setStatus('open');
    $lobby->setCreatedAt(new \DateTimeImmutable());

    $form = $this->createForm(LobbyType::class, $lobby);
    $form->handleRequest($request);

    if ($form->isSubmitted() && $form->isValid()) {
        $em->persist($lobby);
        $em->flush();

        $this->addFlash('success', 'Лобі успішно створено!');
        return $this->redirectToRoute('app_lobby_show', ['id' => $lobby-
>getId()]);
    }

    return $this->render('lobby/create.html.twig', [
        'form' => $form->createView(),
    ]);
}
```

Рисунок 2.12 – Функція створення нового ігрового лобі

Цей фрагмент коду демонструє використання патерну `Dependency Injection` — ми отримуємо екземпляр `EntityManagerInterface` у параметрах методу. У функції створюється новий екземпляр класу `Lobby`, якому автоматично призначається поточний користувач як хост (власник). Далі, використовуючи патерн `Repository` та `Unit of Work`, після

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 51   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

успішної валідації форми зміни фіксуються в базу даних MySQL за допомогою persist() та flush().

Після створення лобі користувачі мають змогу комунікувати між собою за допомогою вбудованого чату. Логіка відправки повідомлень реалізована у ChatController. Функція підтримує не лише текстові повідомлення, але й відправку зображень та голосових повідомлень (рис. 2.13).

```
#[Route('/lobby/{id}/chat/send', name: 'app_lobby_chat_send', methods:
['POST'])]
public function sendLobbyMessage(Lobby $lobby, Request $request,
EntityManagerInterface $em, CloudinaryService $cloudinary): Response
{
    $content = trim($request->request->get('message', ''));
    $voiceFile = $request->files->get('voice');

    $message = new ChatMessage();
    $message->setSender($this->getUser());
    $message->setLobby($lobby);
    $message->setContent($content ?: '');
    if ($voiceFile && $voiceFile->isValid()) {
        $voiceUrl = $this->storeVoice($voiceFile, $cloudinary);
        $message->setType('voice');
        $message->setAttachmentUrl($voiceUrl);
        $message->setContent('[Голосове повідомлення]');
    }
    $em->persist($message);
    $em->flush();
    return new JsonResponse(['status' => 'ok', 'message' => $this-
    >formatMessage($message)]);
}
```

Рисунок 2.13 – Функція відправки повідомлень у чаті лобі

У цій функції реалізовано перевірку наявності прикріплених файлів. Якщо користувач надсилає голосове повідомлення, воно автоматично завантажується у хмарне сховище за допомогою сервісу CloudinaryService. Відповідь повертається у форматі JSON для асинхронного оновлення інтерфейсу клієнта без перезавантаження сторінки.

Щодо процесу реалізації інтелектуального підбору (Matchmaking), цей алгоритм повністю винесено у сервісний шар (MatchmakingService). Виконання

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 52   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

підбору виглядає як послідовність дій, відображена на UML-діаграмі послідовності (рис. 2.14).

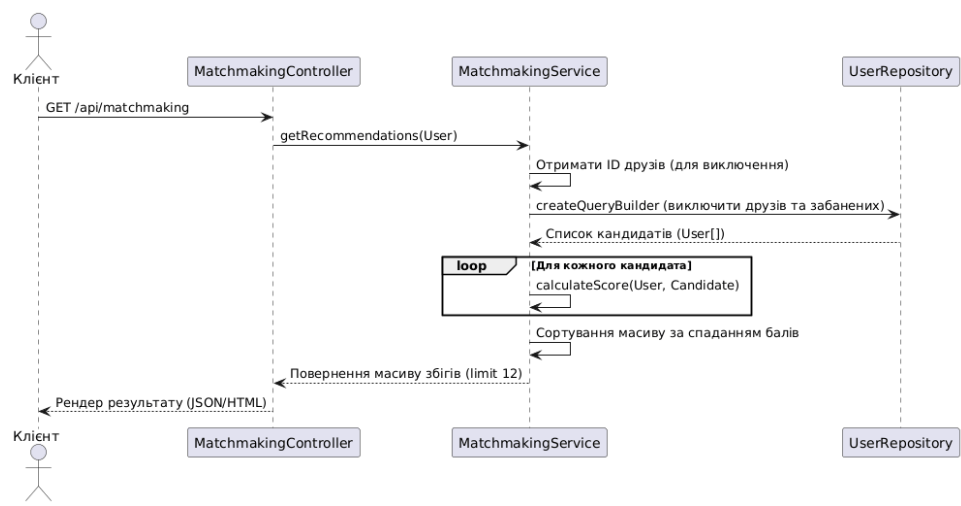


Рисунок 2.14 – UML-діаграма послідовності алгоритму Matchmaking

Для швидкого витягування кандидатів сервіс використовує оптимізований QueryBuilder, який попередньо відсіює невідповідних за мовою кандидатів безпосередньо на рівні СУБД (рис. 2.15).

```
public function getRecommendations(User $user, int $limit = 12): array
{
    // Отримання ID друзів для виключення з пошуку
    $friendIds = $this->getFriendIds($user);

    // Запит до БД: вибір усіх активних користувачів, крім друзів
    $qb = $this->userRepo->createQueryBuilder('u')
        ->where('u.id NOT IN (:excludeIds)')
        ->andWhere('u.isBanned = false')
        ->setParameter('excludeIds', $friendIds);

    $candidates = $qb->getQuery()->getResult();

    // Обчислення балів (Score) для кожного кандидата
    $scored = [];
    foreach ($candidates as $candidate) {
        $score = $this->calculateScore($user, $candidate, $userGameIds);
        if ($score > 0) {
            $scored[] = [
                'user' => $candidate,
                'score' => $score,
            ];
        }
    }

    // Сортування масиву за спаданням балів
    $scored = $this->sort($scored);

    // Повернення масиву збірів (limit 12)
    return $scored;
}
```

```

        'reasons' => $this->getMatchReasons($user, $candidate,
$userGameIds),
    ];
    }
}
// Сортування за балами та повернення лімітованої кількості
usort($scored, fn($a, $b) => $b['score'] <=> $a['score']);
return array_slice($scored, 0, $limit);
}

```

Рисунок 2.15 – Лістинг методу отримання кандидатів для матчмейкінгу

Окремо варто відзначити інтеграцію голосового зв'язку WebRTC. Сигналінг, тобто обмін технічними даними для встановлення P2P з'єднання, здійснюється через протокол Mercure за допомогою патерну Publisher/Subscriber [6].

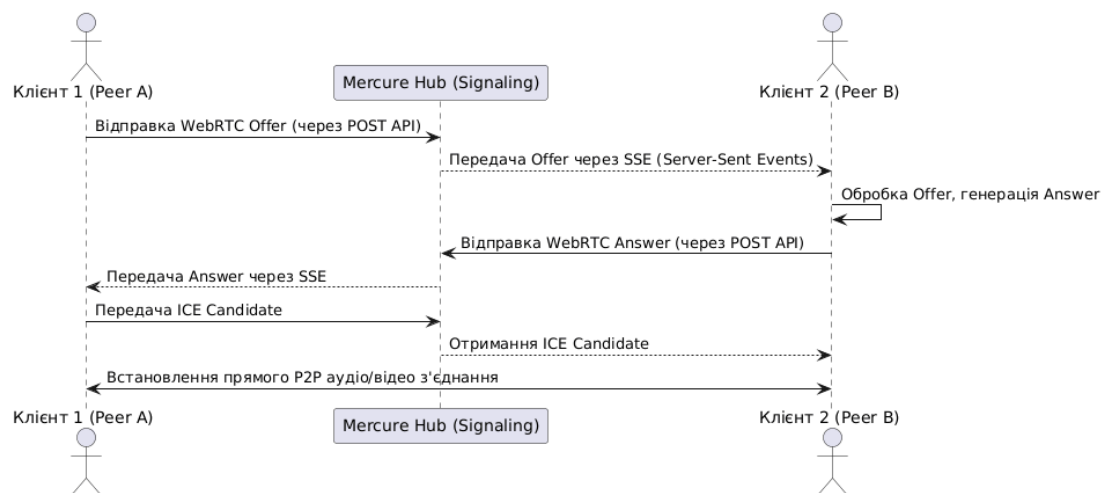


Рисунок 2.16 – UML-діаграма послідовності встановлення P2P з'єднання через WebRTC

Ця архітектура дозволяє зняти навантаження з основного PHP-сервера, оскільки безпосередній голосовий чи відео трафік йде напряму між клієнтами (P2P), а PHP-додаток (Symfony) та Mercure Hub лише допомагають їм обмінятися технічною інформацією для початкового підключення [2].

Для забезпечення гнучкості, масштабованості та зручності підтримки коду, компонентна архітектура платформи GameFinder побудована за модульним

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 54   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

принципом. Взаємодію основних структурних блоків системи відображено на UML-діаграмі компонентів (рис. 2.17).

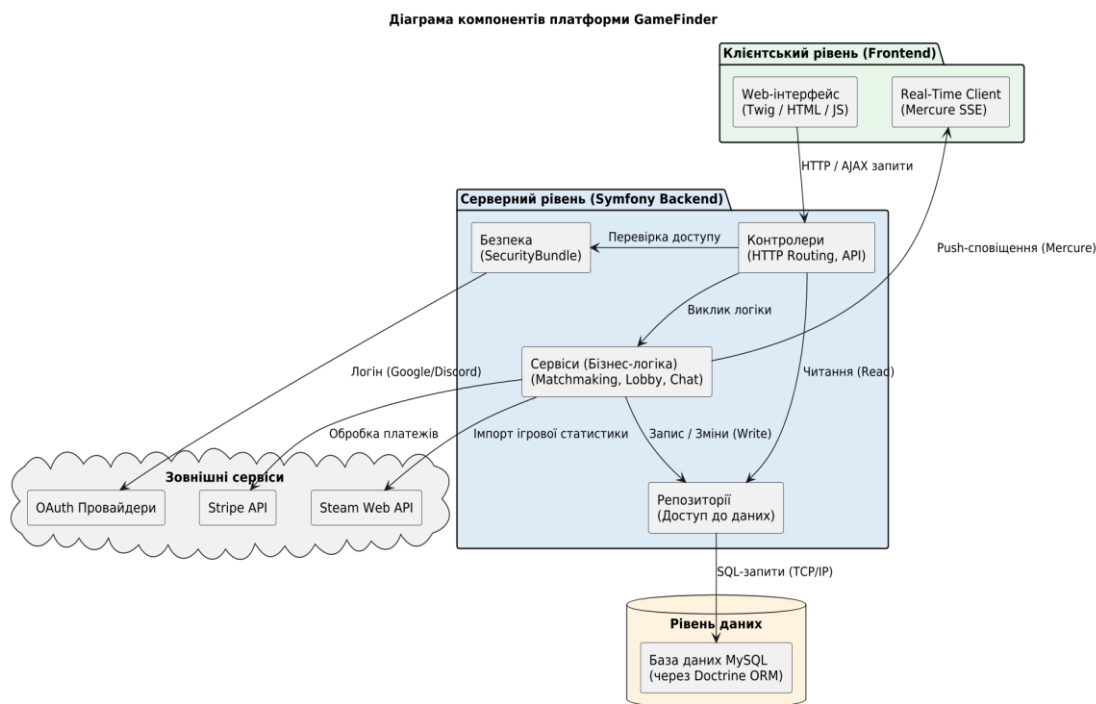


Рисунок 2.17 – Діаграма компонентів системи

Архітектура платформи розділена на чотири логічні рівні, що забезпечує гнучкість та простоту підтримки.

Клієнтський рівень (Frontend) забезпечує інтерактивну взаємодію з користувачем через браузер. Він включає веб-інтерфейс, побудований на шаблонізаторі Twig для швидкого серверного рендерингу (SSR), що покращує SEO та зменшує час первинного завантаження [13]. Для візуалізації динамічних елементів, таких як модальні вікна відгуків, використовується JavaScript (ES6) та ванільний CSS, що дозволяє уникнути перевантаження проєкту важкими SPA-фреймворками. Також інтегровано Real-Time клієнт (Mercure SSE), який підтримує постійне з'єднання для миттєвого отримання повідомлень у чаті та WebRTC сигналінгу без необхідності постійного опитування сервера [6].

Серверний рівень (Backend) є ядром системи, розробленим на фреймворку Symfony [2]. Він складається з "тонких" контролерів для обробки HTTP-запитів

та маршрутизації, а також компонента SecurityBundle для безпечної авторизації та рольового доступу (RBAC) [3]. Основна бізнес-логіка інкапсульована в спеціалізованих сервісах (Matchmaking, Lobby, Chat), які легко тестувати та масштабувати завдяки вбудованому Dependency Injection контейнеру. Доступ до даних абстрагований через репозиторії, що генерують оптимізовані запити. Рівень даних представлений реляційною СУБД MySQL. Взаємодія з нею здійснюється виключно через Doctrine ORM, що абстрагує таблиці бази даних в PHP-об'єкти, гарантуючи надійний захист від SQL-ін'єкцій через використання підготовлених виразів [4]. База даних оптимізована для високих навантажень: налаштовано композитні індекси для швидкого пошуку лобі та каскадні видалення для підтримання цілісності даних.

Зовнішні сервіси складають четвертий рівень, де платформа глибоко інтегрована зі сторонніми API для розширення функціоналу. Використовуються OAuth-провайдери (Google, Discord) для швидкої реєстрації без паролів, Steam Web API для підтвердження ігрового досвіду через синхронізацію досягнень, а також платіжний шлюз Stripe [9]. Останній бере на себе всю обробку транзакцій Premium-підписки, дотримуючись стандартів PCI-DSS, тому конфіденційні дані карток не зберігаються на серверах платформи. Такий підхід повністю відповідає патерну MVC (Model-View-Controller) та принципам SOLID, гарантуючи сувору ізоляцію обов'язків. Контролери не містять жодної бізнес-логіки, сервіси працюють незалежно від HTTP-контексту, а репозиторії відповідають виключно за персистенцію стану системи. Це робить код легко читабельним та готовим до подальшого розширення.

## Висновки до другого розділу

Спроектовано структуру бази даних під управлінням MySQL, що складається з 11 взаємопов'язаних таблиць (зокрема «User», «Lobby», «ChatMessage», «GameEvent», «Review»), які

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 56   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

повністю покривають потреби системи у зберіганні даних. Взаємодія з базою реалізована через безпечний прошарок ORM Doctrine.

Детально опрацьовано архітектурні рівні додатка (Entity, Service, Controller). Розроблено та описано ключові алгоритми роботи платформи, серед яких: створення ігрових лобі, функціонування алгоритмічної системи розумного підбору гравців (Matchmaking), обмін повідомленнями в реальному часі через систему опитування та Mercure, завантаження медіафайлів за допомогою Cloudinary, а також інтеграція зі сторонніми API (Steam для імпорту ігор та Stripe для обробки Premium-підписок).

Підсумком проєктування стала побудова високорівневої діаграми компонентів, яка наочно демонструє чітке розділення відповідальності між клієнтською частиною, сервером на базі фреймворку Symfony, базою даних та зовнішніми хмарними сервісами. Це гарантує надійність, продуктивність та високу масштабованість розробленої ігрової екосистеми.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 57   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

## РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБПЛАТФОРМОЮ «GAMEFINDER»

### 3.1. Порядок встановлення та налаштування вебплатформи

Встановлення та налаштування платформи вимагає розгортання всіх її компонентів та налаштування як кожного елементу окремо, так і загальної системи в цілому. Для опису архітектури розгортання платформи було створено діаграму розгортання (рис. 3.1).

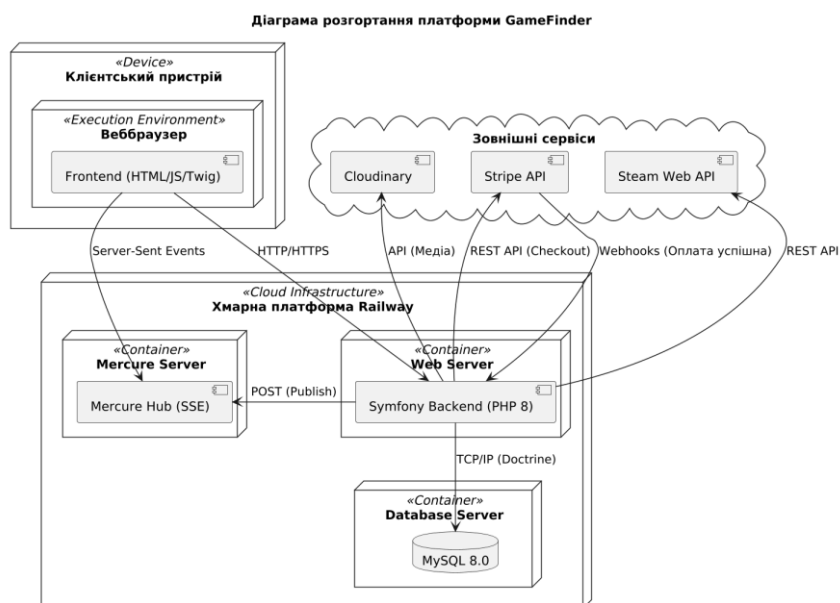


Рисунок 3.1 – Діаграма розгортання платформи GameFinder

Щодо архітектури розгортання, основним середовищем для продуктового хостингу розробленого додатку виступає сучасна хмарна платформа Railway. Вибір саме цього сервісу обґрунтований його надійністю, масштабованістю та повною підтримкою парадигми "Infrastructure as Code" (IaC). Railway забезпечує автоматизоване розгортання проєкту безпосередньо з основного репозиторію контролю версій GitHub за моделлю безперервної доставки (Continuous Delivery) [16]. Кожного разу, коли розробник зливає новий стабільний код у головну гілку, Railway автоматично запускає процес збирання (Build process), завантажуючи всі

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 58   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

необхідні залежності та компілюючи фінальний образ додатку, після чого прозоро замінює стару версію на нову без переривання доступу користувачів (Zero-downtime deployment).

Серверна частина платформи GameFinder побудована на базі потужного PHP-фреймворку Symfony із використанням найновішої версії мови програмування PHP 8.2 [2]. У конфігурації продуктового сервера Symfony працює у режимі production, що активує строгі політики кешування (OpCache, Doctrine Query Cache), мінімізацію шаблонів Twig та відключення інструментів відлагодження, гарантуючи максимальну продуктивність та захист від витoku технічної інформації. Цей бекенд відповідає за обробку складної бізнес-логіки системи, криптографічний захист паролів за допомогою алгоритму Argon2i, управління сесіями користувачів, обробку лобі-запитів та виконання математичних обчислень алгоритму матчмейкінгу.

Для постійного, структурованого та надійного збереження всіх даних екосистеми — від профілів користувачів та історії особистих повідомлень до відгуків та глобальних статистичних показників — було інтегровано реляційну систему управління базами даних MySQL 8.0 [5]. Використання кодування utf8mb4 забезпечує коректну підтримку всіх мов та емодзі в текстових чатах. Управління структурою бази даних здійснюється виключно через систему міграцій (Doctrine Migrations), що гарантує версіонування схеми та виключає можливість розсинхронізації структури бази даних між локальним середовищем розробника та продуктовим сервером.

Зважаючи на те, що комунікація є ядром платформи GameFinder, стандартна модель "клієнт-сервер" із регулярним опитуванням (HTTP Polling) виявилася неефективною. Щоб забезпечити миттєву доставку системних сповіщень та текстових повідомлень у чатах лобі без надмірного навантаження на основний сервер та виснаження з'єднань MySQL, було розгорнуто спеціалізований автономний сервер Mercure Hub. Даний мікросервіс написаний на мові Go і працює виключно за технологією Server-Sent Events (SSE) [6]. Впровадження Mercure дозволило налагодити односпрямовану асинхронну

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 59   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

трансляцію оновлень (Push notifications) напряму у браузері клієнтів. Комунікація між Symfony (Publisher) та Mercure надійно захищена підписаними JSON Web Tokens (JWT), що унеможлиблює несанкціоновану відправку сповіщень.

Складна архітектура GameFinder активно взаємодіє з низкою зовнішніх хмарних сервісів через прикладні програмні інтерфейси (API), знімаючи навантаження з власного сервера. По-перше, для зберігання, обробки та оптимізації медіафайлів (наприклад, аватарів користувачів чи зображень, прикріплених у чаті лобі) використовується хмарне сховище Cloudinary. За допомогою його API зображення "на льоту" обрізаються, стискаються та конвертуються у сучасний легкий формат WebP, що значно покращує швидкість завантаження інтерфейсу клієнта. По-друге, обробка фінансових транзакцій та управління життєвим циклом платних Premium-підписок (створення сесій оплати, скасування, перевірка строків дії) повністю делегована захищеному фінансовому шлюзу Stripe. Синхронізація підписок відбувається через захищені Webhooks, цифрові підписи яких суворо перевіряються Symfony сервером. По-третє, для глибокої інтеграції ігрового контексту застосовується офіційний Steam Web API [9]. Цей інтерфейс дозволяє здійснювати перевірку справжності Steam-акаунтів користувачів через OAuth-авторизацію та автоматично синхронізувати власні ігрові досягнення, бібліотеки куплених ігор та загальний рівень акаунта на платформу GameFinder.

Що стосується локального середовища розробки та процесу тестування, встановлення проєкту є повністю автоматизованим завдяки використанню інструменту контейнеризації Docker Compose. Даний підхід вирішує проблему сумісності середовищ "it works on my machine" і вимагає від розробника мінімальних попередніх налаштувань: наявності встановленого ядра Docker, локального середовища PHP 8.2 та пакетного менеджера Composer [1]. На початковому етапі розгортання виконується клонування вихідного коду проєкту з віддаленого репозиторію та ініціалізується завантаження всіх необхідних PHP-залежностей (бібліотек) за допомогою Composer. Наступним, критично

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 60   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

важливим кроком є налаштування локального конфігураційного файлу змінних середовища. У ньому розробник обов'язково прописує автентифікаційні дані у вигляді DSN-рядка для підключення до локальної бази даних MySQL, генерує власні криптографічні секретні ключі безпеки для запуску локального сервера Mercure Hub, прописує параметри інтеграції протоколу OAuth 2.0 (Client ID та Client Secret для коректної роботи авторизації через платформи Google та Discord), а також додає набір унікальних API ключів для всіх вищезгаданих сторонніх хмарних інтеграцій.

Після успішної генерації конфігураційного файлу виконується ініціалізація Docker-контейнерів. В результаті піднімаються повністю налаштовані ізольовані служби для бази даних, Mercure Hub та, за потреби, Mailcatcher для перехоплення і тестування локальних електронних листів. Далі розробник застосовує міграції бази даних через консоль Symfony, що призводить до автоматизованого створення всієї структури таблиць, індексів та зовнішніх ключів. Розгортання завершується компіляцією фронтенд-ассетів (CSS та JS) через Webpack Encore та ініціалізацією локального вебсервера (зазвичай Symfony CLI), після чого платформа GameFinder стає повністю працездатною за локальною адресою, готовою до тестування нового функціоналу або подальшої розробки.

У системі налаштовано суворе розмежування доступу на основі ролей за допомогою компонента Symfony Security. Це гарантує високий рівень захисту конфіденційних даних та запобігає несанкціонованому втручанню в роботу платформи. Неавторизовані гості мають змогу переглядати лише публічну інформацію, таку як загальні лобі чи каталог ігор, а також сторінки реєстрації та логіну. Після авторизації користувачі отримують базовий доступ до створення власних кімнат, спілкування в чатах, управління списком друзів та відправки відгуків. Важливим аспектом є механізм перевірки активності акаунту: користувачі з неповноцінно заповненими профілями можуть зіткнутися з обмеженнями функціоналу матчмейкінгу. Для власників платної підписки передбачено статус Premium-користувача, що надає пріоритет під час

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    | 61   |

алгоритмічного підбору напарників та відкриває доступ до ексклюзивного візуального оформлення профілю, включаючи анімовані рамки та стікери [10]. За безпеку та дотримання правил відповідають модератори, які мають спеціальну панель для оперативної обробки скарг, приховування неприйняттого контенту, закриття токсичних лобі та тимчасового блокування порушників. Найвищий рівень доступу мають адміністратори, що керують глобальним каталогом ігор, налаштовують інтеграції зі Steam API, відслідковують загальну статистику активності та призначають ролі для всього персоналу. Завдяки такій чіткій ієрархії платформа забезпечує масштабованість адміністративних процесів та гнучке управління багатотисячною спільнотою.

### 3.2. Структура інтерфейсу вебплатформи

Взаємодія користувачів із системою відбувається через зручний, сучасний та адаптивний вебдодаток. Інтерфейс побудовано з використанням CSS-фреймворку Bootstrap 5, що гарантує коректне відображення як на десктопних, так і на мобільних пристроях [12]. Завдяки гнучким налаштуванням доступу (Public Access), неавторизований користувач при першому відкритті сайту потрапляє на головну сторінку (рис. 3.2), яка виступає в ролі публічного дашборду. Тут гість має змогу вільно переглядати список відкритих лобі, каталог ігор, ігрові події та глобальний рейтинг (Leaderboard). Однак для безпосередньої взаємодії з екосистемою (створення лобі, відправки повідомлень чи оцінювання гравців) необхідно увійти в акаунт, натиснувши кнопку «Увійти» або «Зареєструватись». Сторінка авторизації містить форму з валідацією для вводу Email та пароля, а також кнопки інтеграції зі сторонніми сервісами (Discord, Google) через протокол OAuth для миттєвого входу без необхідності запам'ятовувати паролі [18].

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 62   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

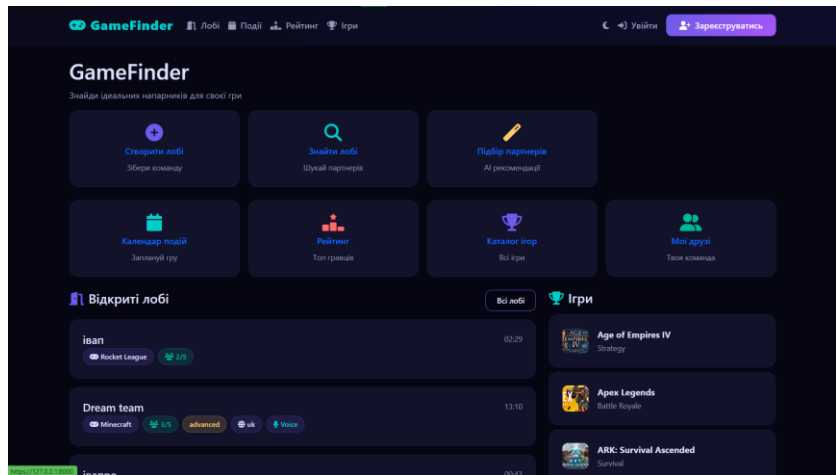


Рисунок 3.2 – Головна сторінка для неавторизованого користувача

Якщо у гравця ще немає акаунту, він переходить до процесу реєстрації (рис. 3.3). Ця сторінка містить динамічну форму, яка перевіряє унікальність введеного нікнейму та надійність пароля ще до відправки даних на сервер.

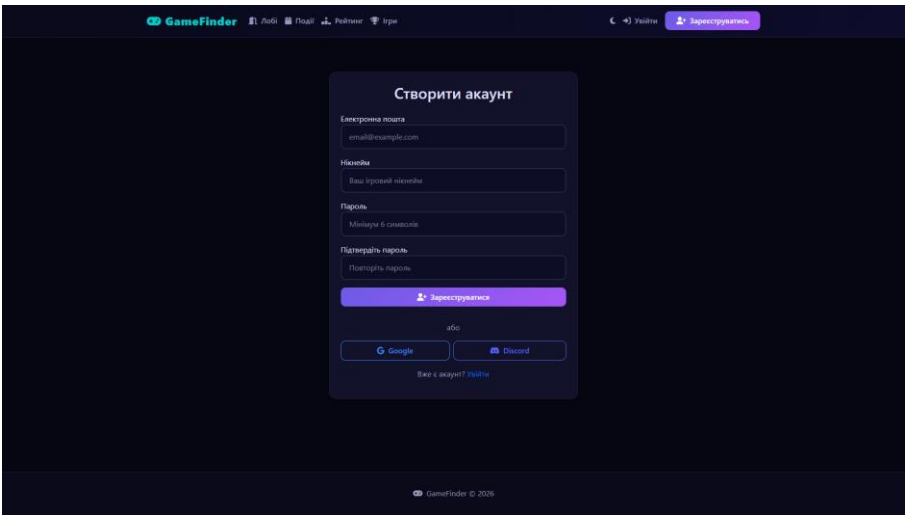


Рисунок 3.3 – Сторінка реєстрації

Після успішної реєстрації система автоматично перенаправляє користувача на сторінку авторизації (рис. 3.4). Тут він може увійти у свій новостворений акаунт, безпечно ввівши електронну пошту та пароль, або ж скористатися швидким входом через сторонні сервіси (Google, Discord) за допомогою OAuth-токенів.

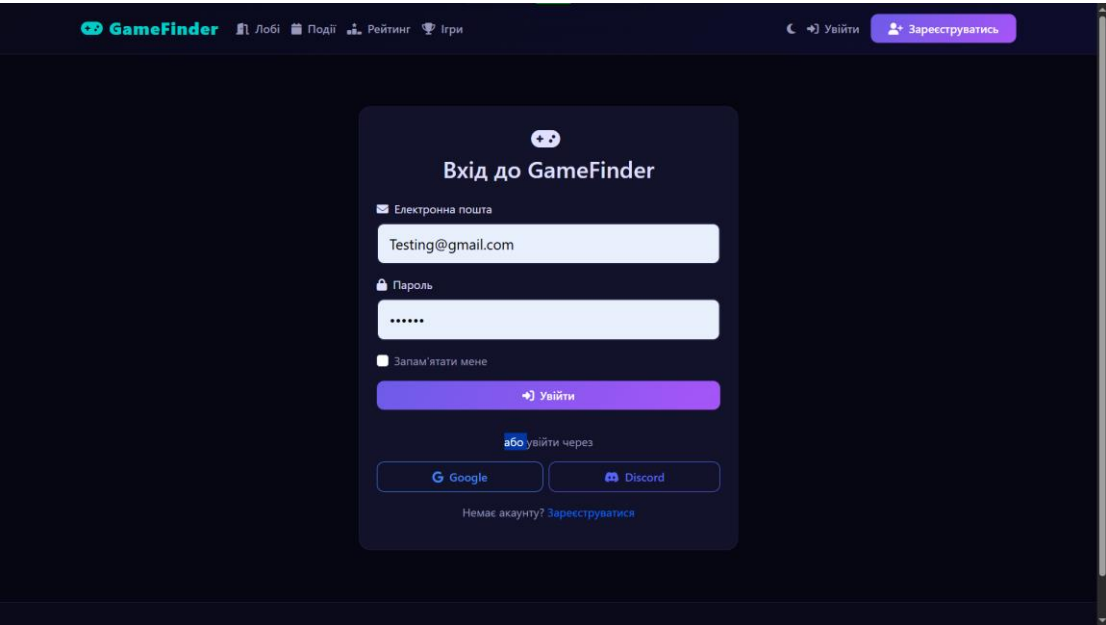


Рисунок 3.4 – Сторінка авторизації

Після успішної авторизації гравець перенаправляється на головну сторінку (дашборд) системи (рис. 3.5). Тут відображається його поточний статус, швидкі посилання на останні лобі, список активних друзів онлайн, а також актуальні новини платформи. Інтерфейс розроблено таким чином, щоб користувач міг у 1-2 кліки перейти до будь-якої критичної функції платформи.

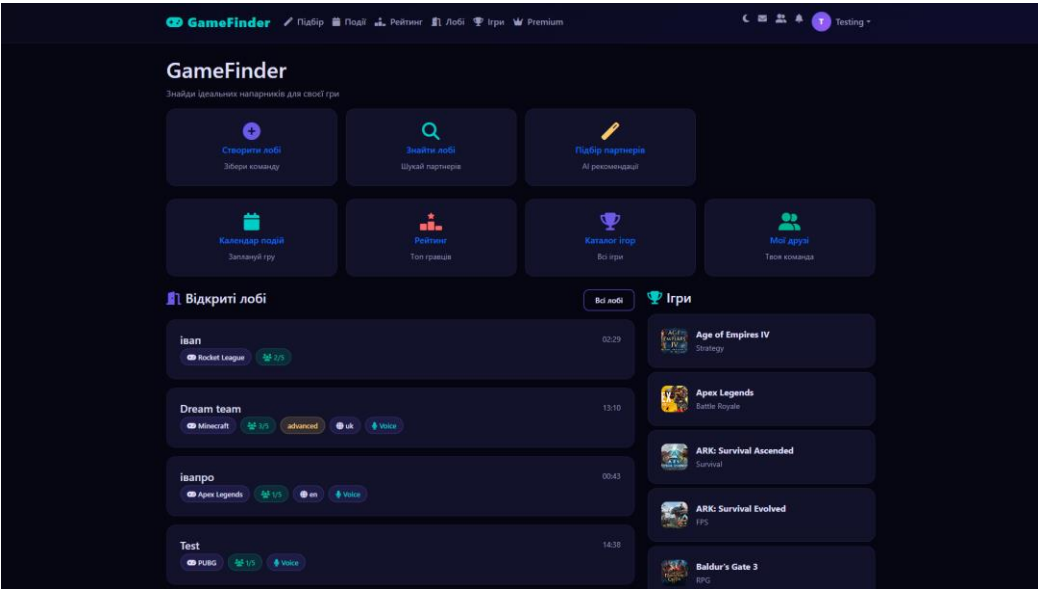


Рисунок 3.5 – Головна сторінка

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 64   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Якщо користувач бажає знайти ідеальних напарників для гри, він може скористатися інструментом інтелектуального підбору (рис. 3.6) [24]. Ця сторінка містить інтерактивний інтерфейс, де результати роботи багатофакторного скорингового алгоритму відображаються у вигляді динамічної сітки карток кандидатів. Картки автоматично сортуються за релевантністю — від найвищого відсотка сумісності до найнижчого. На кожній картці виведено загальний бал сумісності (наприклад, 95%) та набір візуальних тегів із піктограмами, що пояснюють ключові причини збігу (наприклад, "Спільні ігри", "Одне місто", "Схожий вік" або "Високий рейтинг"). Безпосередньо з картки гравця можна в один клік переглянути його розширений профіль, надіслати запит на додавання в друзі або миттєво запросити до свого активного лобі.

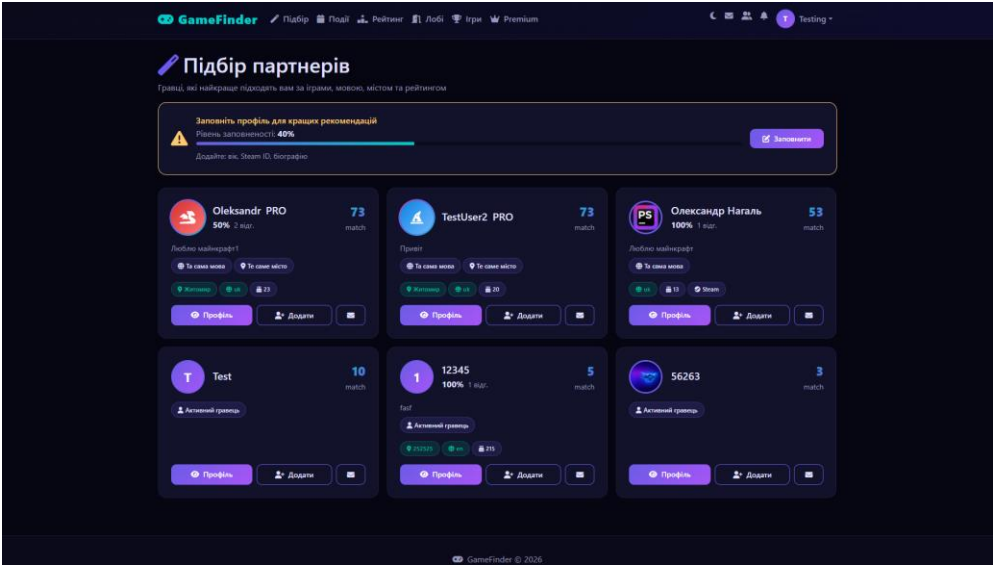


Рисунок 3.6 – Інтерфейс інтелектуального підбору (Matchmaking)

Для планування майбутніх ігрових сесій реалізовано сторінку ігрових подій (рис. 3.7). Вона має вигляд календаря або списку карток, де користувачі можуть фільтрувати розклад майбутніх матчів чи аматорських турнірів за конкретною грою та датою.

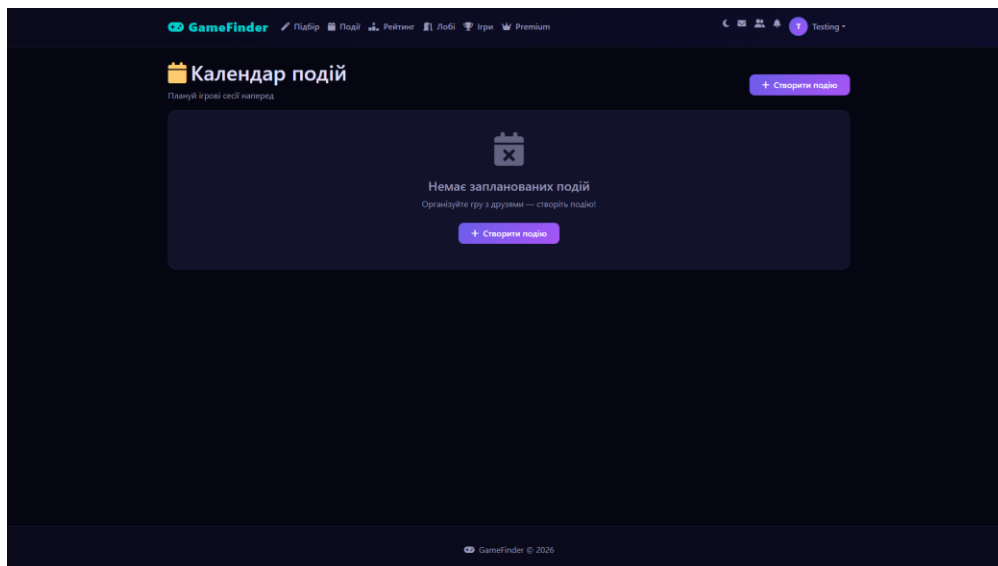


Рисунок 3.7 – Сторінка ігрових подій

За необхідності користувач може ініціювати створення власної події через спеціальне модальне вікно (рис. 3.8). Інтерфейс форми містить вибір дати через вбудований календар (Datepicker), селект для вибору гри з каталогу та поле для визначення максимальної кількості учасників.

Рисунок 3.8 – Створення події

Організатор заходу має право скасувати його. Для уникнення випадкових скасування, видалення події (рис. 3.9) обов'язково супроводжується діалоговим вікном підтвердження (Confirmation modal), після чого подія безповоротно зникає з розкладу.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 66   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

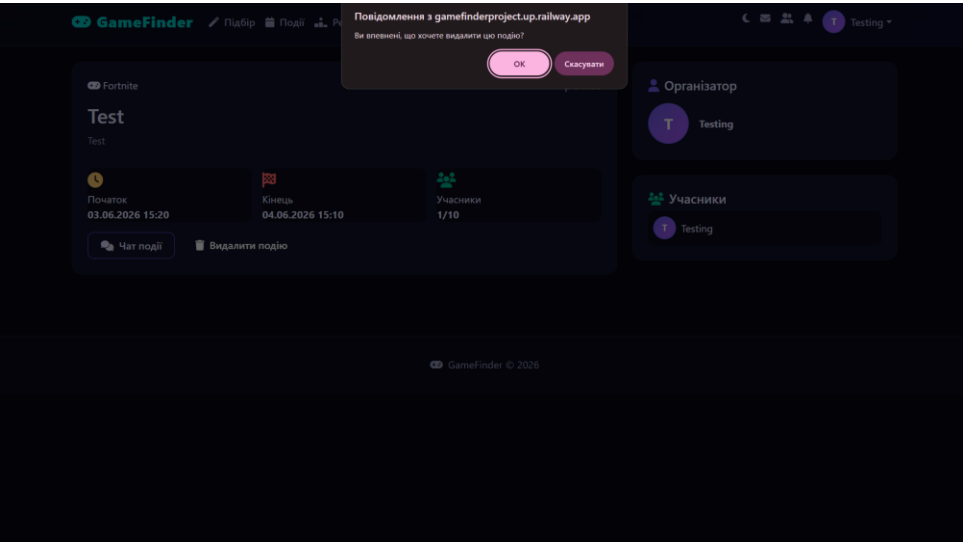


Рисунок 3.9 – Видалення події

Для ручного пошуку користувач переходить на сторінку лобі (рис. 3.10), яка містить сітку відкритих кімнат. Зліва доступна панель фільтрів для миттєвого сортування лобі за грою, рангом, місцями чи мовою без перезавантаження сторінки.

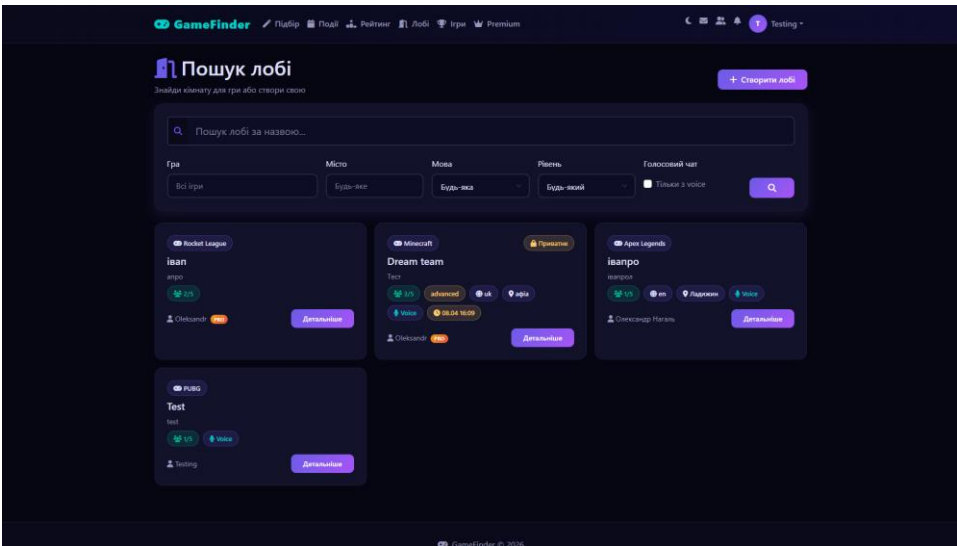


Рисунок 3.10 – Сторінка лобі

Якщо жодна кімната не підходить, гравець може перейти до створення лобі (рис. 3.11). Форма створення лобі містить перемикачі (Radio buttons) для

вибору приватності (відкрите/закрите), Dropdown для вибору гри та велике текстове поле для опису правил кімнати.

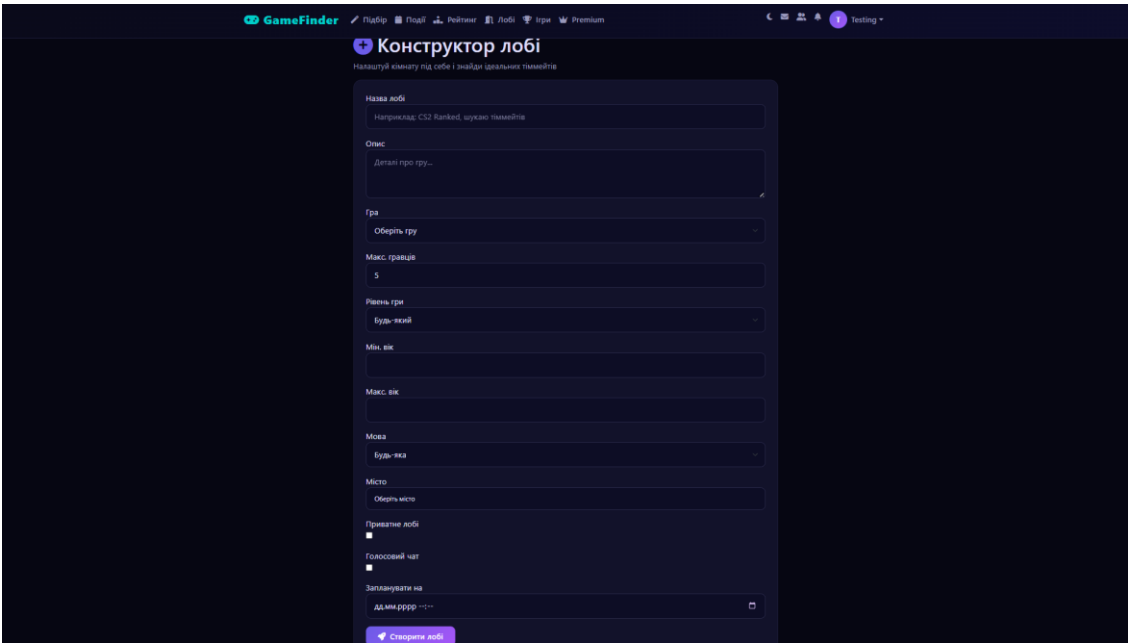


Рисунок 3.11 – Створення лобі

Основним інструментом комунікації гравців до початку гри є сторінка чату лобі (рис. 3.12). Її інтерфейс розділено на дві частини: справа – список поточних учасників кімнати, зліва – історія повідомлень у режимі реального часу (завдяки технології Mercure) [6]. У нижній частині екрана розташована панель вводу тексту зі кнопками для прикріплення файлів (скріншотів). Окрім текстового спілкування, в чаті інтегровано підтримку голосових повідомлень (Voice Messages). Завдяки використанню Web Audio API, користувач може записати аудіо безпосередньо через мікрофон свого пристрою, натиснувши відповідну іконку, після чого запис миттєво відправляється у стрічку повідомлень для прослуховування іншими учасниками.

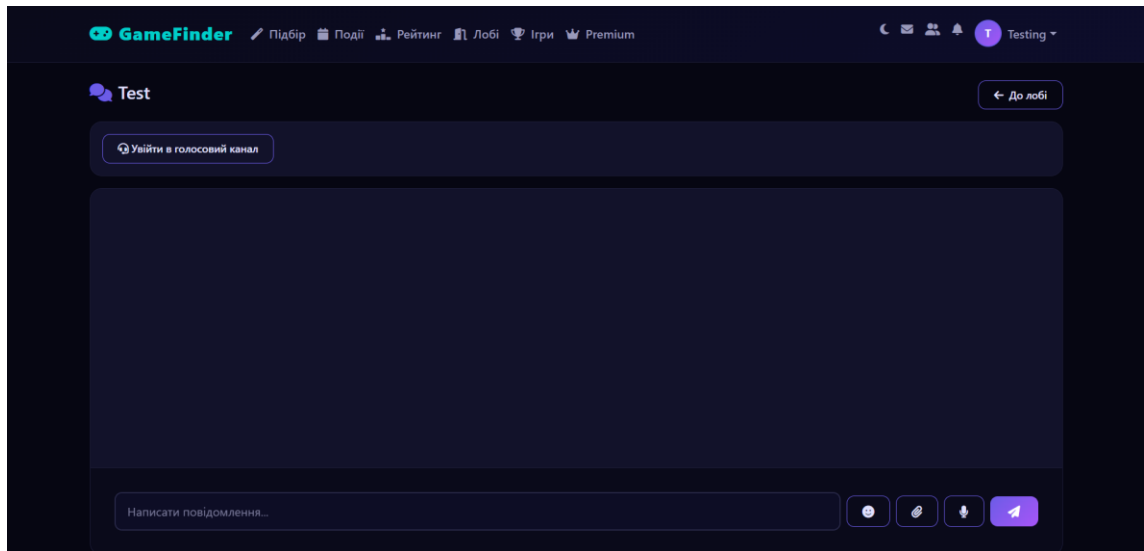


Рисунок 3.12 – Чат лобі

Окрім текстового чату, для повноцінної координації гравців інтегровано сторінку голосового каналу (рис. 3.13). Під час підключення до Voice-кімнати інтерфейс відображає список поточних спікерів з динамічною індикацією розмови (анімація навколо аватарів), а також панель швидкого доступу для вимкнення мікрофона чи навушників. Передача аудіо реалізована за допомогою технології WebRTC, що дозволяє спілкуватися з мінімальною затримкою прямо у браузері, без необхідності запуску сторонніх програм (як Discord чи TeamSpeak) [7].

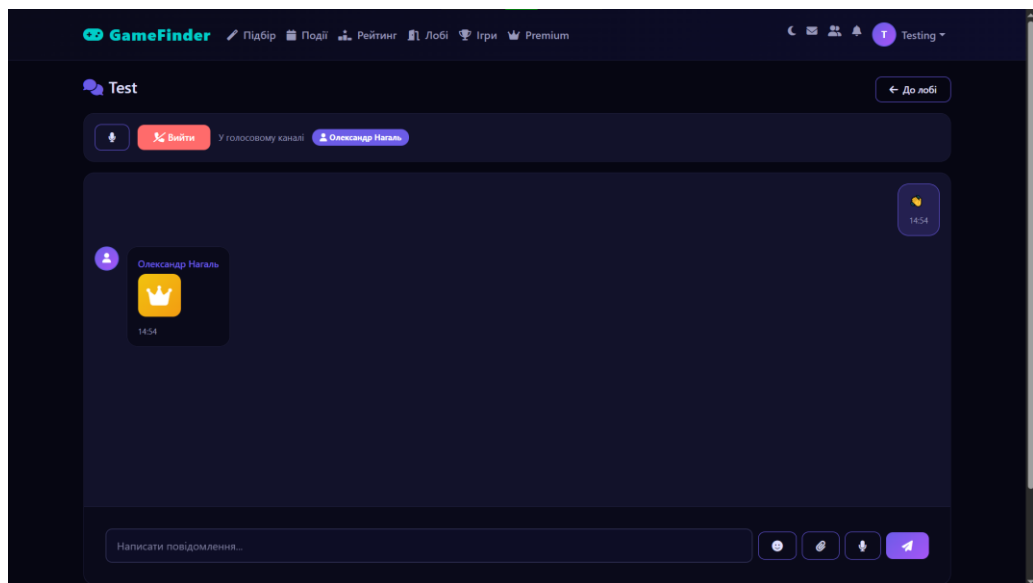


Рисунок 3.13 – Голосовий канал лобі

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 69   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Організатор має можливість достроково закрити або видалити лобі (рис. 3.14). Для цього в інтерфейсі управління кімнатою передбачено спеціальну кнопку, натискання якої викликає модальне вікно підтвердження, щоб уникнути випадкового закриття.

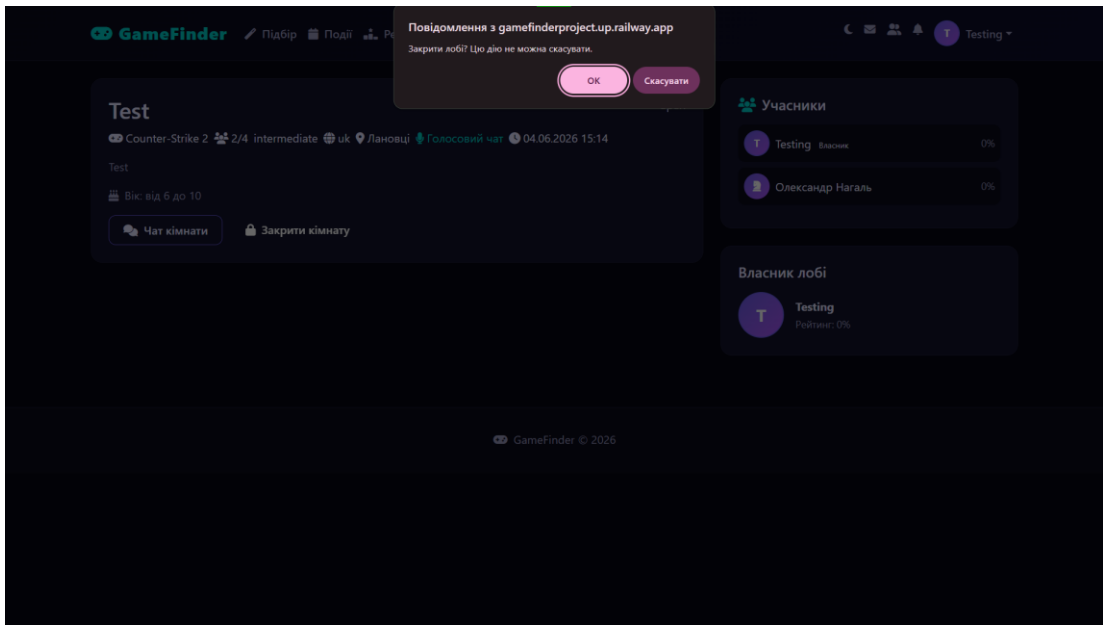


Рисунок 3.14 – Закриття лобі

Після завершення спільної гри (коли лобі переходить у статус closed), користувачам стає доступним інтерфейс оцінювання напарників. Візуально це реалізовано через форму або модальне вікно (рис. 3.15), де гравець може поставити позитивну (Like) або негативну (Dislike) оцінку та залишити короткий текстовий відгук про поведінку чи навички партнера. Ця форма має захист від повторної відправки (механізм upsert).

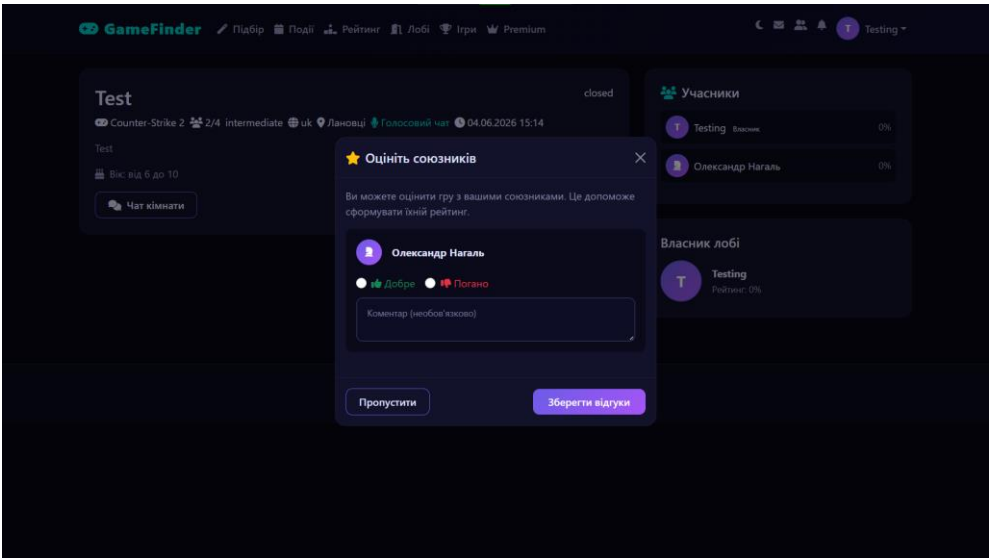


Рисунок 3.15 – Відгук про користувача

На основі залишених оцінок формується загальна сторінка рейтингу (Leaderboard) (рис. 3.16). Цей розділ виступає не лише як інформаційна база, а і як змагальний елемент платформи. Тут гравці можуть переглядати відсортований список найкращих користувачів за рівнем довіри (Trust Factor), який динамічно перераховується системою. Для зручності навігації таблиця підтримує пагінацію, а також швидкий пошук гравця за його нікнеймом. Клік по будь-якому учаснику з лідерборду дозволяє детально переглянути його ігрову статистику, співвідношення позитивних і негативних оцінок та безпосередньо самі текстові відгуки від інших напарників.

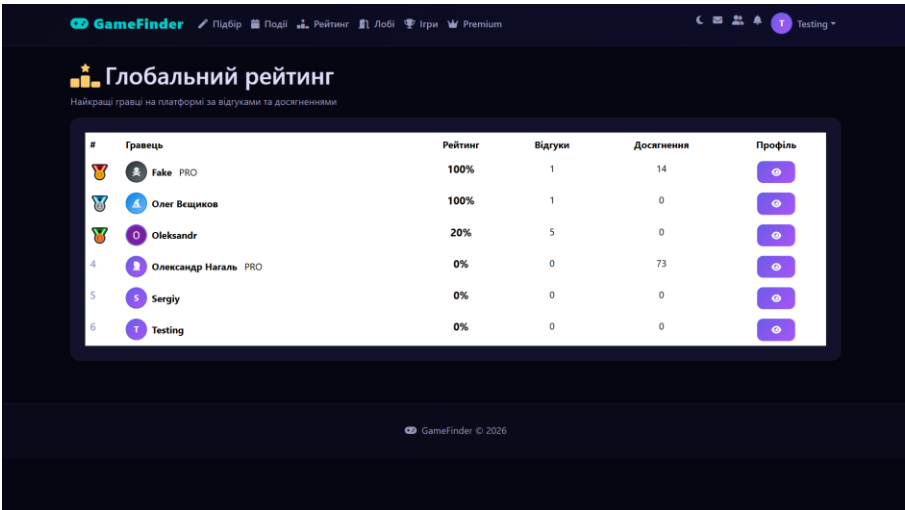


Рисунок 3.16 – Сторінка Рейтингу

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 71   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Сторінка ігор (рис. 3.17) представляє собою візуально привабливий каталог усіх доступних на платформі кіберспортивних ігор. Кожна гра оформлена у вигляді великої картки з офіційною обкладинкою, описом та кнопкою швидкого переходу до пошуку лобі саме для цієї гри.

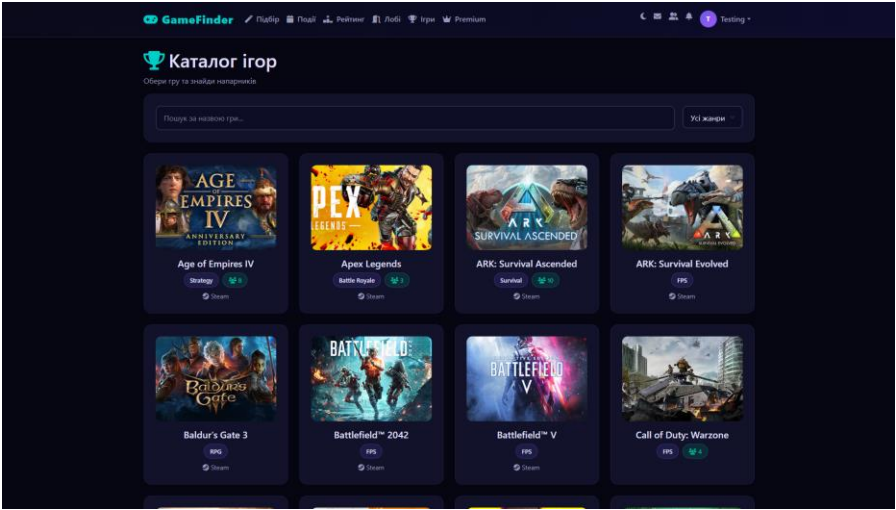


Рисунок 3.17 – Сторінка ігор

Гравці, які бажають підтримати платформу, можуть перейти на сторінку Преміум (рис. 3.18). Це стилізована Landing-сторінка, де за допомогою яскравих банерів та порівняльних таблиць описані всі переваги підписки (кастомну рамку аватара, пріоритет у пошуку).

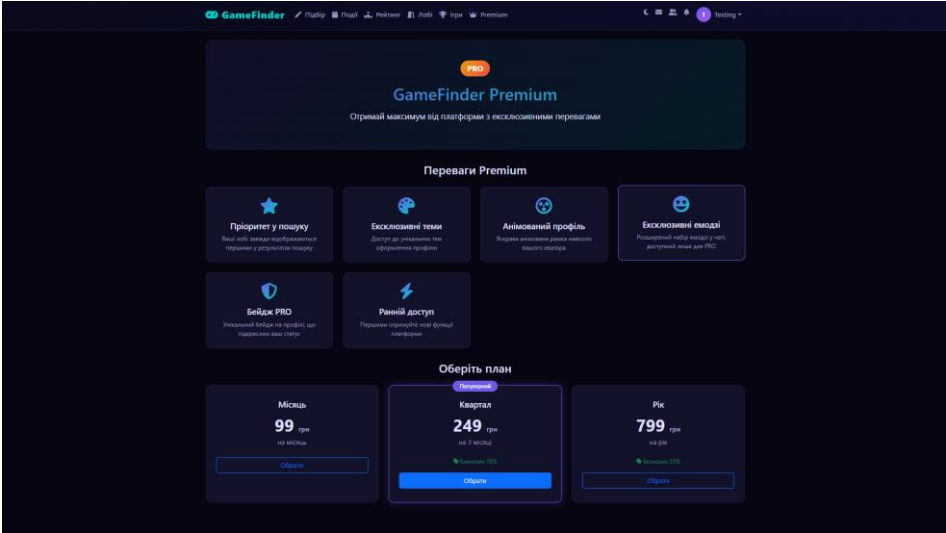


Рисунок 3.18 – Сторінка Преміум

Після вибору тарифу користувач перенаправляється на сторінку оплати (рис. 3.19), яка є безпечним віджетом (Checkout) інтегрованим напряму від платіжного провайдера Stripe. Платформа не збирає та не зберігає дані банківських карток у своїх формах.

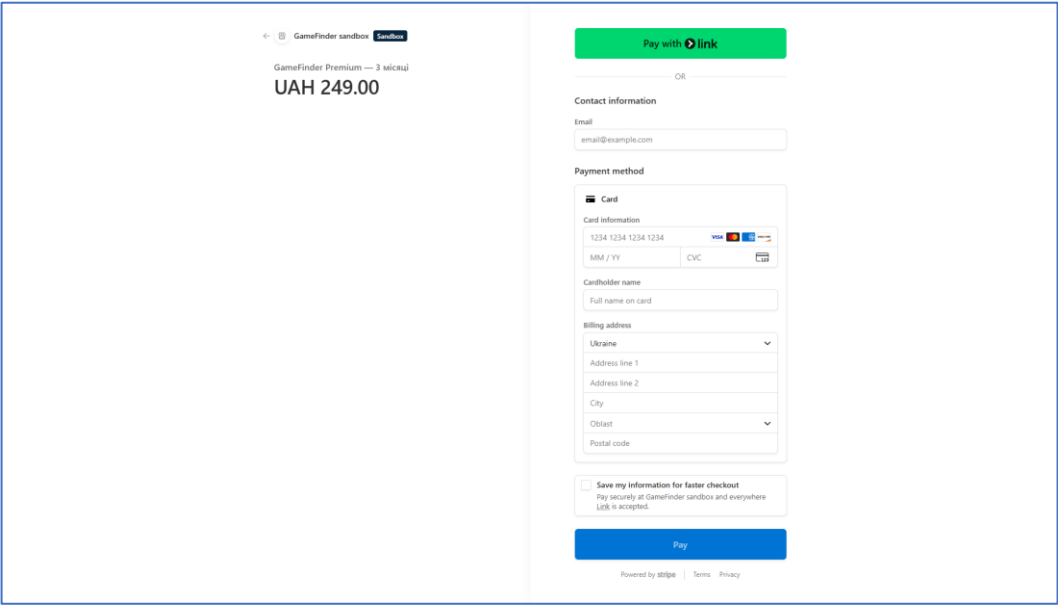


Рисунок 3.19 – Сторінка оплати підписки

Поза межами лобі комунікація забезпечується через сторінку особистих повідомлень (рис. 3.20). Інтерфейс імітує класичні месенджери: ліворуч відображається список діалогів, а праворуч – вікно обраного чату з історією переписки.

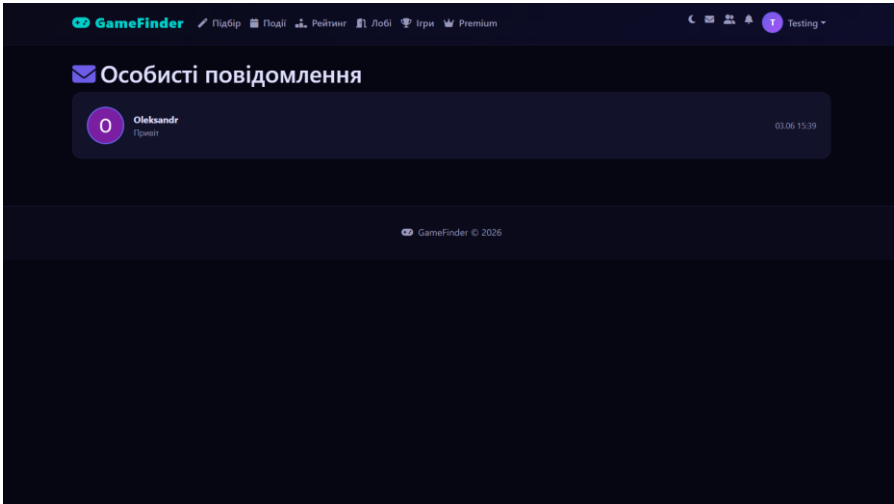


Рисунок 3.20 – Сторінка особистих повідомлень

Управління соціальними зв'язками відбувається на сторінці друзів (рис. 3.21). Вона містить кілька вкладок: "Усі друзі", "Онлайн", "Вхідні заявки". Користувач може прийняти або відхилити заявку натисканням відповідних кнопок прямо зі списку.

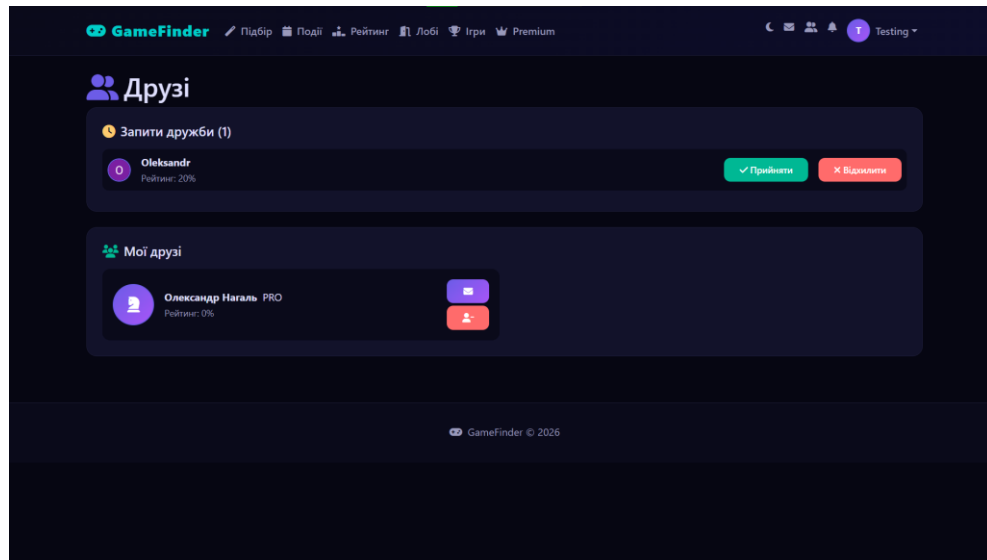


Рисунок 3.21 – Сторінка друзів

Системні сповіщення збираються на сторінці нотифікацій (рис. 3.22). Непрочитані сповіщення виділяються фоном. При кліку на сповіщення (наприклад, "Вас запросили в лобі") користувач автоматично перенаправляється на відповідну сторінку.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 74   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

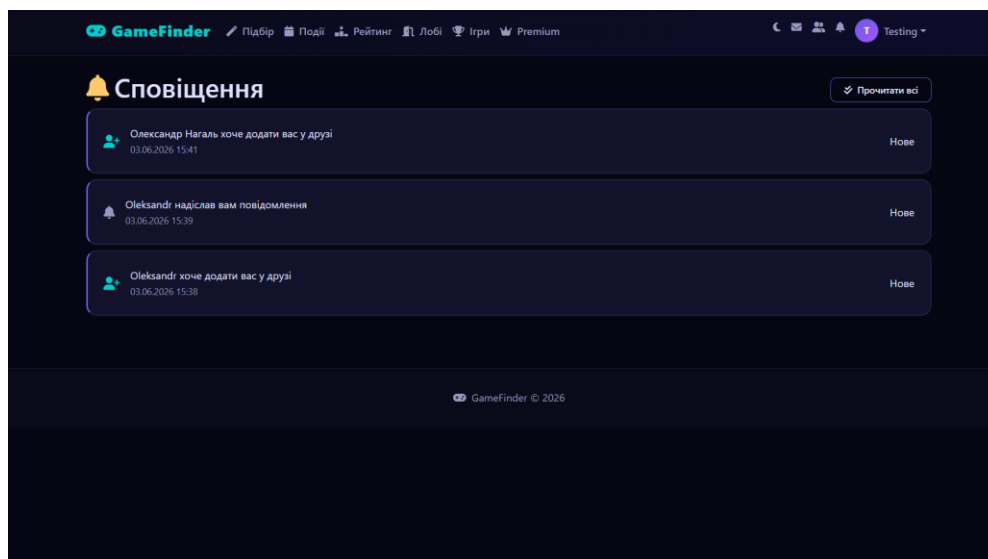


Рисунок 3.22 – Сторінка повідомлень

Кожен гравець має свою унікальну візитну картку – публічну сторінку профілю (рис. 3.23). Інтерфейс профілю гнучко налаштовується та має чіткий візуальний поділ: користувачі з Premium-статусом автоматично отримують виділення нікнейму кольором, анімовані рамки аватара та можливість встановлення кастомного фонового зображення, що робить їх сторінку значно виразнішою. На сторінці компактно та зручно структуровані основні блоки: детальна біографія гравця, загальний рівень довіри (Trust Factor), динамічно підвантажена ігрова статистика зі Steam (наприклад, кількість награних годин або рівень акаунта), а також стрічка останніх текстових відгуків від напарників. Додатково, для зручної взаємодії, під аватаром розташовані кнопки швидких дій: запросити у друзі, написати повідомлення або надіслати скаргу.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 75   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

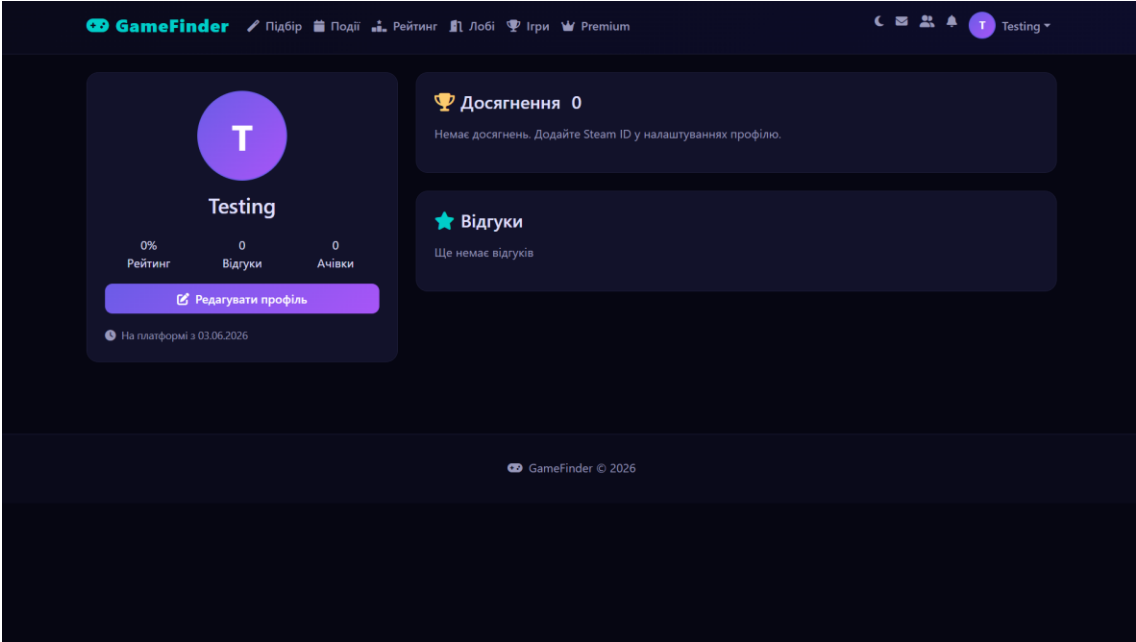


Рисунок 3.23 – Сторінка профіля

Користувач може змінити свої дані через інтерфейс редагування профілю (рис. 3.24). Окрім стандартних текстових полів для зміни опису чи пароля, тут розміщена кнопка "Синхронізувати зі Steam", яка через OAuth оновлює ігрову бібліотеку користувача.

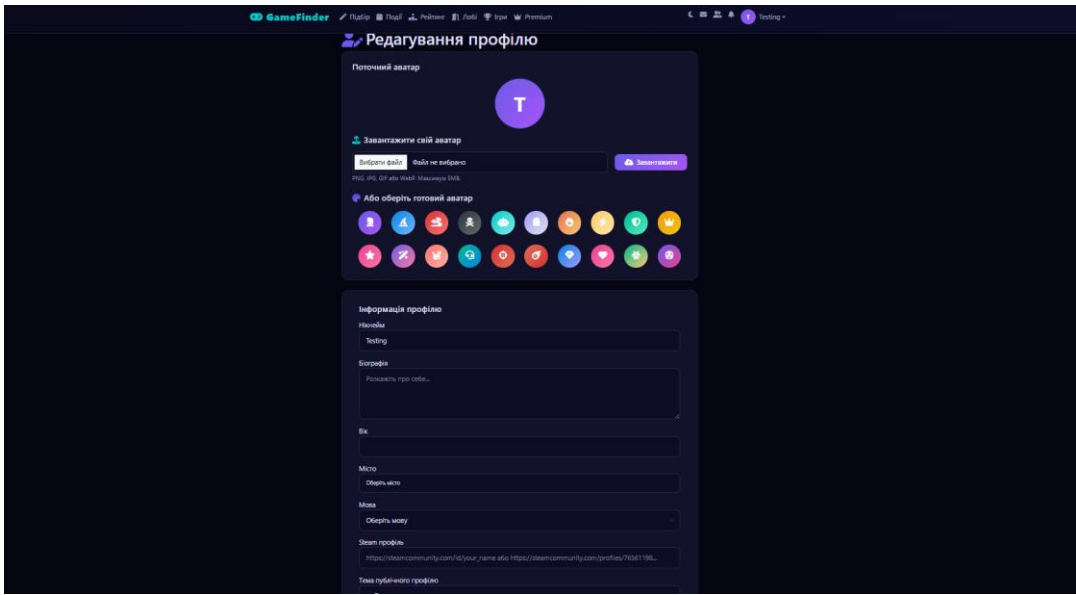


Рисунок 3.24 – Редагування профілю

Для підтримки порядку на платформі передбачена панель модератора (рис. 3.25). Це закритий розділ зі строгою стилістикою та табличним представленням даних.

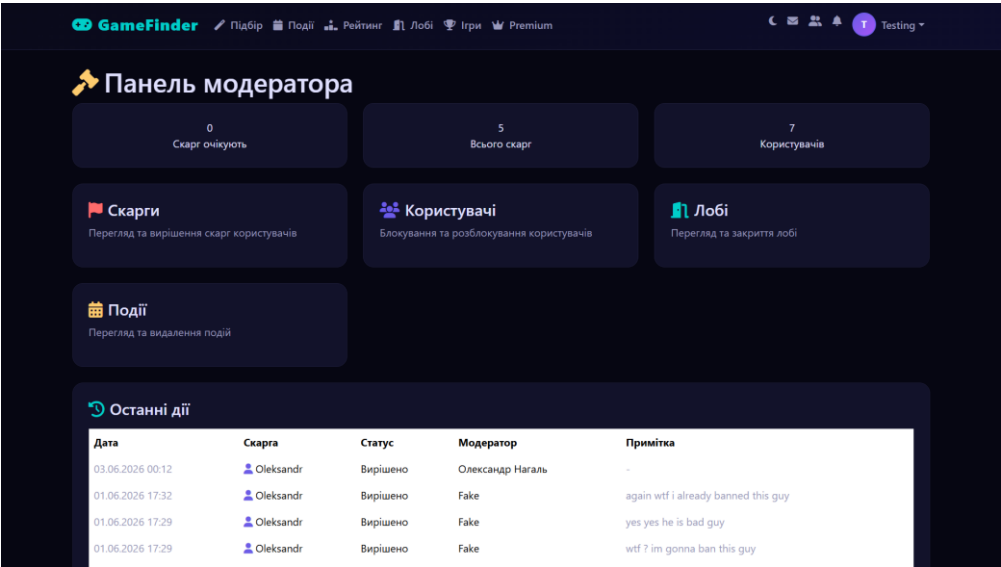


Рисунок 3.25 – Модераторська панель

В рамках цієї панелі модератор може переглядати сторінку скарг (рис. 3.26). Вона виглядає як черга тікетів (Tickets), де кожен запит містить причину скарги, посилання на повідомлення в чаті (доказову базу) та кнопки для швидкої реакції ("Забанити", "Відхилити").

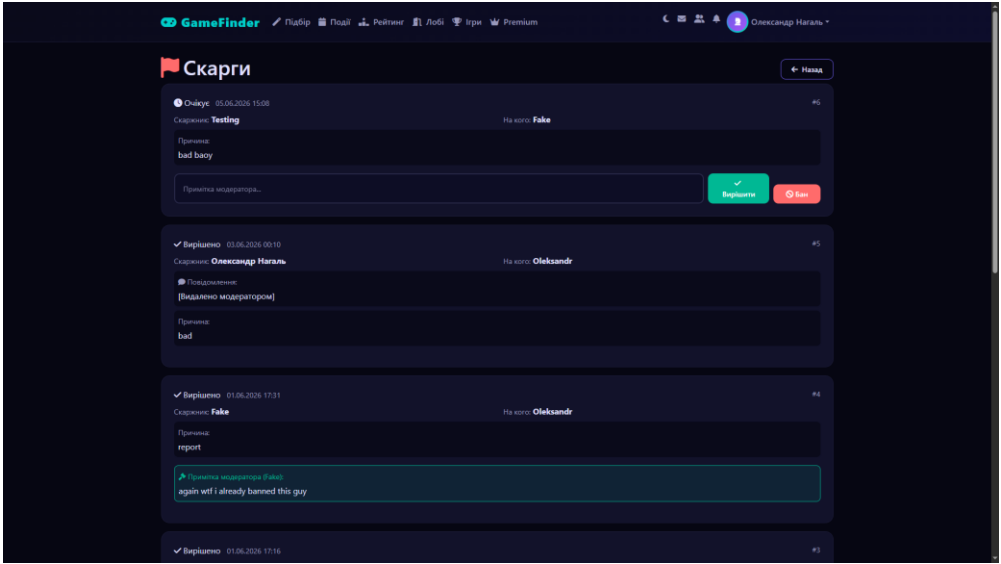


Рисунок 3.26 – Сторінка скарг

Сторінка управління користувачами (рис. 3.27) містить інструменти пошуку по базі даних гравців. Знайшовши порушника, модератор може відкрити модальне вікно для тимчасового блокування (бану), вказавши тривалість покарання.

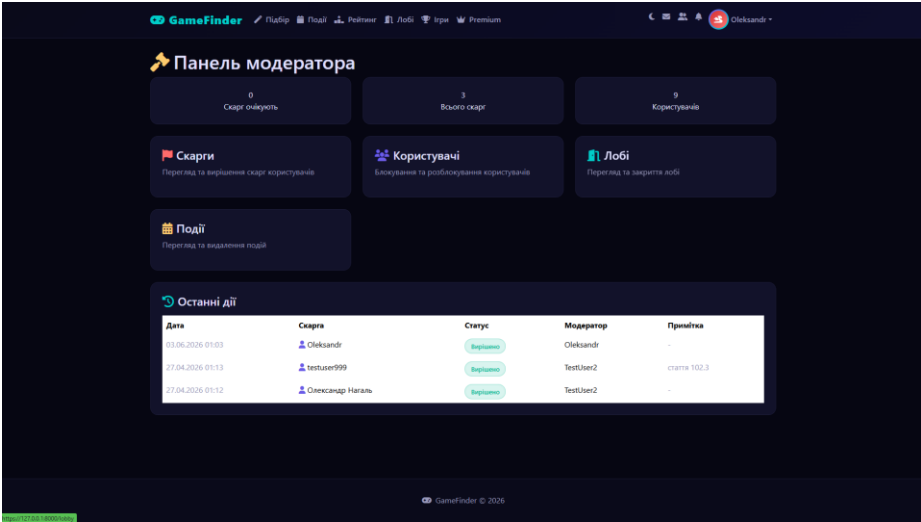


Рисунок 3.27 – Сторінка управління користувачами

Вищий рівень доступу має адміністратор. Адмінська панель (рис. 3.28) є дашбордом з графіками активності, розширеними таблицями користувачів та глобальними налаштуваннями платформи.

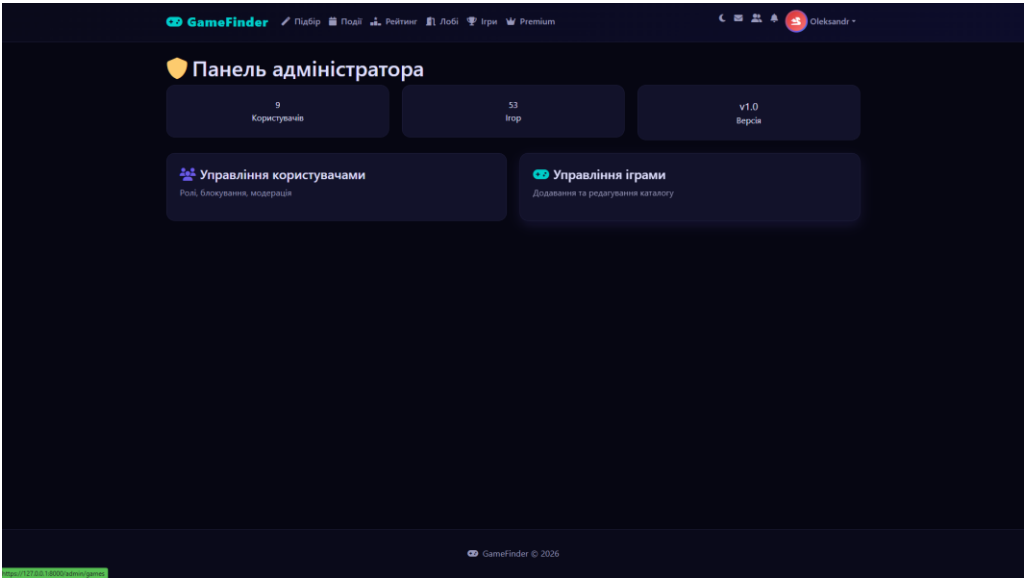


Рисунок 3.28 – Адмінська панель

Одним із головних завдань адміністратора є управління іграми (рис. 3.29). Інтерфейс містить CRUD-таблицю (Create, Read, Update, Delete) з переліком усіх підтримуваних ігор.

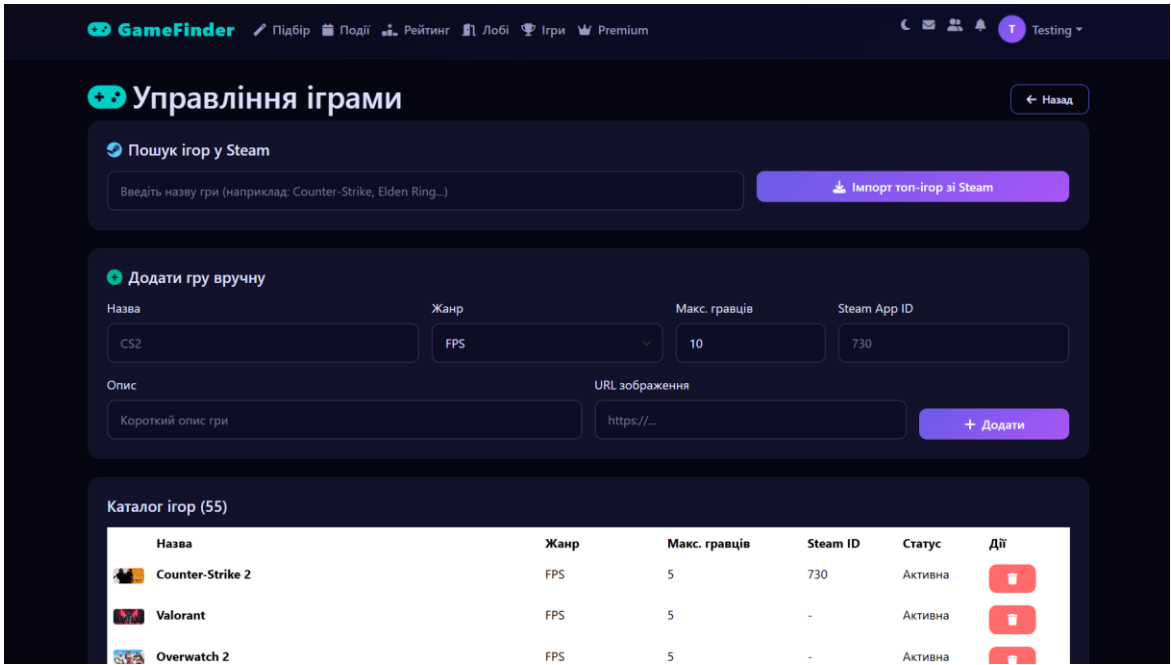


Рисунок 3.29 – Управління іграми

За потреби адміністратор може ініціювати додавання гри через спеціальну форму (рис. 3.30), ввівши її назву, опис, жанр та завантаживши обкладинку через вбудований Drag-and-Drop завантажувач (який відправляє фото на Cloudinary).

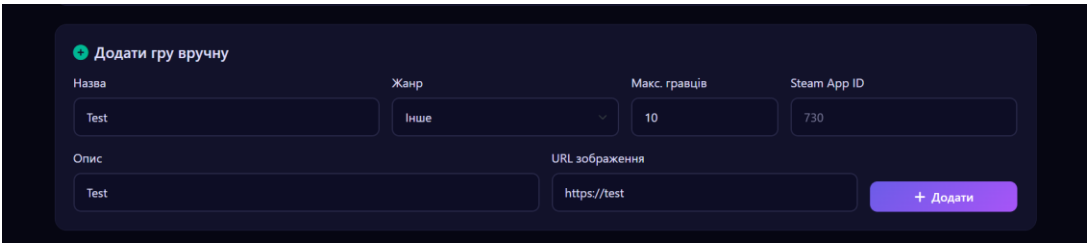


Рисунок 3.30 – Додавання гри

Якщо гра втратила популярність, виконується процедура видалення (рис. 3.31). Інтерфейс видалення вимагає додаткового підтвердження з боку

адміністратора, оскільки це каскадно впливає на існуючі лобі та статистику гравців.

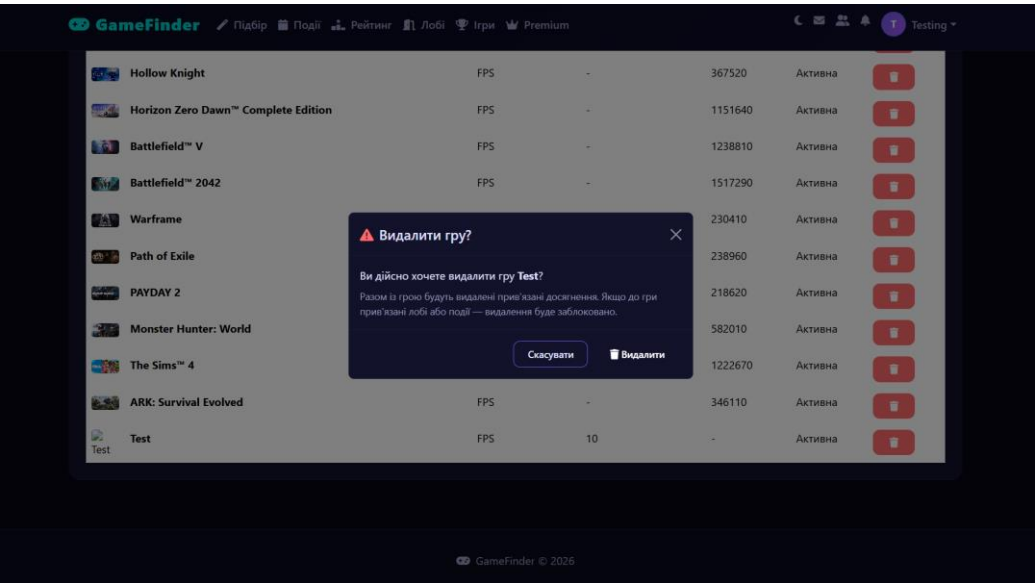


Рисунок 3.31 – Видалення гри

3.3. Тестування роботи програмного продукту

Для забезпечення надійності, стабільності та відповідності вебплатформи GameFinder заданим вимогам, було проведено комплексне тестування, яке охоплює функціональне (модульне та інтеграційне) тестування, а також тестування продуктивності.

Для гарантування надійності та стабільності роботи системи ключовим етапом стало модульне та інтеграційне тестування. Функціональна перевірка складної бізнес-логіки повністю реалізована за допомогою сучасного інструментарію автоматизованого тестування PHPUnit [15]. Модульне тестування (Unit Testing) детально покриває основні сервіси та доменні сутності платформи, дозволяючи ізольовано перевірити кожен компонент системи. Зокрема, критично важливий алгоритм підбору гравців (MatchmakingService) був покритий тестами, що імітують різні сценарії розрахунку балів сумісності. Вони доводять коректність нарахування вагових коефіцієнтів на основі мовного збігу, географічної локації, вікової різниці та глобального рейтингу гравців.

Окрім цього, значна увага приділялася тестуванню логіки життєвого циклу ігрових кімнат. За допомогою автоматизованих тестів перевіряються процеси ініціалізації лобі, механізми приєднання нових учасників, коректна обробка прав доступу та алгоритми валідації запитів на приєднання. Паралельно тестується генерація системних сповіщень про критичні події: запрошення до лобі, зміни статусів дружби чи надходження нових відгуків. На рівні базових сутностей перевіряється інкапсульована бізнес-логіка моделі користувача. Сюди входить строга валідація преміум-статусів, механіка накладання тимчасових або перманентних блокувань (банів), алгоритм обчислення залишку часу штрафу, а також коректний перерахунок актуальної кількості прийнятих учасників у межах закритого або відкритого лобі.

Окремо варто виділити процес тестування підсистеми відгуків, яка є фундаментальною частиною для формування прозорості та справедливої репутації гравців на платформі. З метою запобігання штучній накрутці рейтингу та фейковим оцінкам, було впроваджено багатопарову архітектуру захисту, коректність якої доводиться саме за допомогою PHPUnit. Насамперед автоматизовані тести перевіряють правило участі: система має дозволяти залишити відгук лише тому гравцю, з яким ініціатор гарантовано перебував у спільному лобі. Другим рівнем є перевірка стану самої кімнати – оцінювання технічно неможливе, доки лобі не змінить свій статус на успішно завершене. Третім рівнем виступає часовий захист (Anti-Spam), який автоматично відхиляє можливість залишити відгук, якщо лобі проіснувало менше хвилини; ця перевірка повністю блокує скрипти, які намагаються миттєво створювати та закривати кімнати для завищення репутації. Нарешті, механізм тестування підтверджує роботу захисту від дублювання (Upsert-логіка): якщо користувач спробує залишити декілька відгуків одному напарнику за одну гру, система лише оновить існуючий відгук без створення нового запису. Нижче наведено приклад реального лістингу модульного тесту ReviewBusinessLogicTest, написаного за допомогою фреймворку PHPUnit, який програмно доводить дотримання вищезазначених правил [15]:

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 81   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

namespace App\Tests\Service;
use App\Entity\Lobby;
use App\Entity\User;
use App\Service\ReviewService;
use PHPUnit\Framework\TestCase;

class ReviewBusinessLogicTest extends TestCase
{
    public function testCannotReviewIfLobbyNotClosed(): void
    {
        $lobby = new Lobby();
        $lobby->setStatus('active'); // Лобі ще триває

        $author = new User();
        $target = new User();

        $this->expectException(\DomainException::class);
        $this->expectExceptionMessage('Відгук можна залишити лише після
завершення гри (статус closed).');

        $reviewService = new ReviewService();
        $reviewService->validateReviewEligibility($author, $target,
$lobby);
    }

    public function testCannotReviewIfLobbyDurationTooShort(): void
    {
        $lobby = new Lobby();
        $lobby->setStatus('closed');
        // Імітація: лобі було створено і закрито за 10 секунд
        $lobby->setCreatedAt(new \DateTimeImmutable('-10 seconds'));

        $author = new User();
        $target = new User();

        $this->expectException(\DomainException::class);
        $this->expectExceptionMessage('Підозра на накрутку: лобі існувало
менше 60 секунд.');
```

Щодо покриття коду, завдяки впровадженню CI/CD пайплайну на базі GitLab CI код автоматично перевіряється на відсоток покриття тестами [17].

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 82   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Покриття критично важливих сервісів становить 100%, що гарантує відсутність прихованих вразливостей у бізнес-логіці.

Таблиця 3.1.

Звіт про покриття коду

| Компонент / Сервіс | Покриття класів | Покриття методів | Покриття рядків (Lines) |
|--------------------|-----------------|------------------|-------------------------|
| MatchmakingService | 100%            | 75.0%            | 91.8%                   |
| ReviewService      | 100%            | 100%             | 100%                    |
| LobbyService       | 100%            | 100%             | 100%                    |
| SecurityBundle     | 100%            | 90.0%            | 41.9%                   |
| Controllers        | 100%            | 57.1%            | 72.6%                   |

Такий рівень покриття підтверджує, що архітектура стійка до змін (регресійних багів), а всі ключові сценарії використання надійно захищені.

Інтеграційне тестування перевіряє взаємодію різних компонентів системи між собою та з базою даних. Наскрізне тестування життєвого циклу лобі перевіряє процеси створення, приєднання, виходу учасників, а також коректність закриття лобі та запобігання дублюванню запитів. Під час тестування сервісу преміум-підписок перевіряється правильність активації тарифних планів, накопичення днів при повторних покупках та автоматичне завершення протермінованих підписок. Окрім цього, проводиться тестування HTTP-запитів до контролерів, яке гарантує правильну роботу обмежень доступу для неавторизованих користувачів, коректне відображення сторінок управління та сувору валідацію webhook-запитів від платіжної системи Stripe, відхиляючи будь-які виклики без дійсного цифрового підпису.

Використання цих тестів дозволило виявити та виправити логічні помилки ще на етапі розробки, забезпечивши стабільність ключового функціоналу.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 83   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Для перевірки оптимізації було проведено аналіз продуктивності інтерфейсу. Інструмент Google PageSpeed Insights було використано для аналізу швидкості завантаження клієнтської частини додатку [21]. Після завершення оцінювання ключових вебметрик було отримано звіт (рис. 3.32), який продемонстрував відмінні результати: Ефективність – 99, Доступність – 95, Оптимальні методи – 100, SEO – 100.

За результатами перевірки ключові показники продуктивності виявилися на високому рівні. Час до відображення першого контенту (FCP) та найбільшого елемента на сторінці (LCP) склав усього 0,7 секунди. Загальний час блокування (TBT), протягом якого сторінка могла б не реагувати на дії користувача, дорівнює нулю мілісекунд. Крім того, показник кумулятивного зсуву макета (CLS) становить 0.001, що свідчить про ідеальну візуальну стабільність інтерфейсу без будь-яких неочікуваних стрибків чи зміщень блоків під час завантаження.

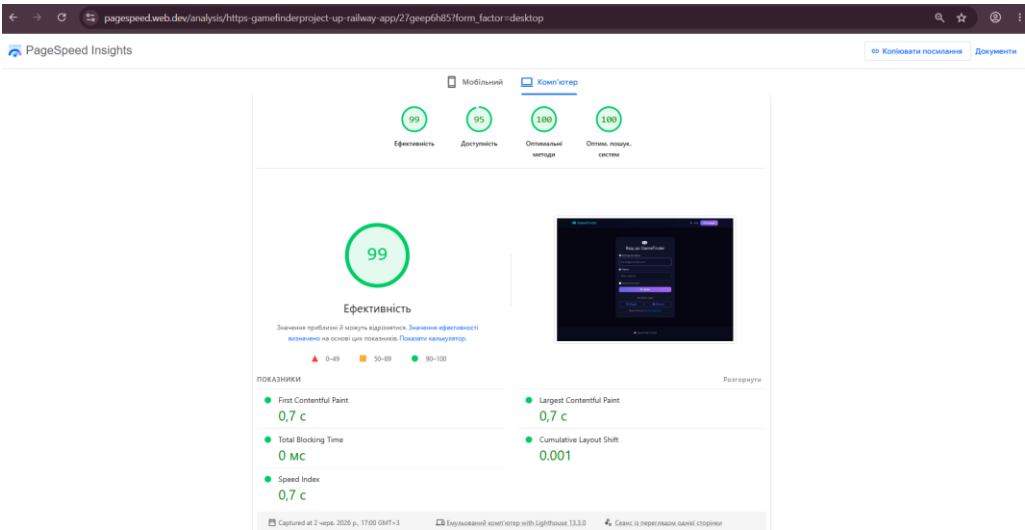


Рисунок 3.32 – Звіт продуктивності інтерфейсу в PageSpeed Insights

На основі рекомендацій PageSpeed було проведено оптимізацію: налаштовано кешування, мініфіковано CSS/JS файли за допомогою Symfony Webpack Encore та оптимізовано завантаження зображень (використання WebP форматів через Cloudinary), що суттєво підвищило швидкість роботи інтерфейсу.

Важливим етапом життєвого циклу розробки є налаштування автоматизованого процесу безперервної інтеграції (Continuous Integration). У системі реалізовано пайплайн на базі GitLab CI, який автоматично запускається завдяки дзеркалюванню репозиторію з GitHub при кожному оновленні коду [17]. Головною задачею цього пайплайну є автоматизована перевірка коду, що відбувається у три послідовні етапи. На етапі збирання система встановлює всі необхідні PHP-залежності через Composer, а також завантажує пакети Node.js для подальшої компіляції фронтенд-ресурсів [1]. Далі виконується статичний аналіз коду, під час якого перевіряється синтаксис PHP-файлів, валідуються шаблони Twig та конфігурація сервіс-контейнера Symfony [13]. На фінальному етапі відбувається паралельний запуск модульних та інтеграційних тестів через PHPUnit з генерацією детальних звітів про результати перевірки.

Це гарантує, що нові зміни відповідають стандартам якості та не ламають існуючий функціонал платформи. Після успішного проходження пайплайну, стабільна версія коду розгортається на хмарному хостингу Railway безпосередньо з основного репозиторію GitHub [16].

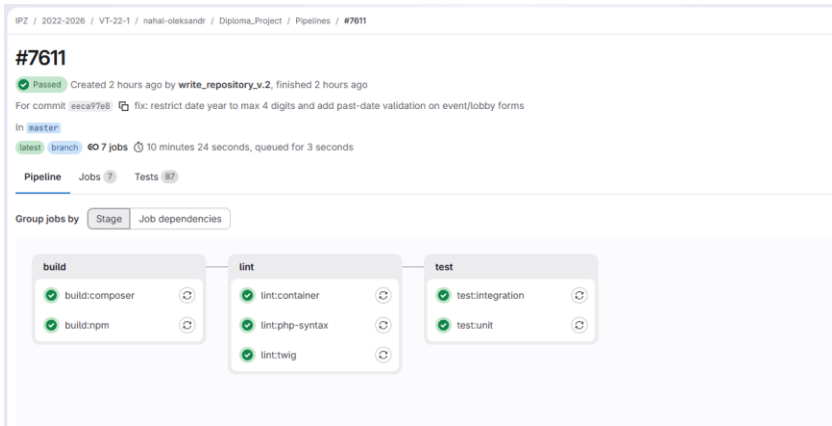


Рисунок 3.33 – Автоматизований процес тестування у пайплайні GitLab CI

### Висновки до третього розділу

У даному розділі наведено схему розгортання вебплатформи GameFinder. Описано ключові компоненти (сервер Railway, БД MySQL, Mercure

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    | 85   |

Hub) та їхню взаємодію із зовнішніми сервісами (Steam API, Stripe, Cloudinary), що розкриває фізичну архітектуру системи.

Надано детальний опис графічного інтерфейсу розробленого програмного продукту. Розглянуто ключові функціональні можливості додатку згідно з визначеними ролями користувачів: від процесів авторизації, роботи інтелектуального алгоритму підбору гравців та взаємодії в лобі, закінчуючи панелями модерації та адміністрування платформи. Усі кроки користувача підкріплені відповідними знімками екранів розробленої системи.

Насамкінець описано процес комплексного тестування платформи. Для перевірки надійності бізнес-логіки було розроблено набір модульних (Unit) та інтеграційних (Integration) тестів за допомогою фреймворку PHPUnit. Окрім цього, за допомогою інструменту Google PageSpeed Insights було проаналізовано та оптимізовано швидкість завантаження клієнтського інтерфейсу, що підтверджується відмінними показниками продуктивності (99/100). Для автоматизації процесів перевірки якості коду успішно впроваджено пайплайн безперервної інтеграції (CI) на базі GitLab CI.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 86   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

## ВИСНОВКИ

У ході виконання випускної кваліфікаційної роботи було успішно вирішено актуальну проблему пошуку напарників для багатокористувацьких ігор шляхом проектування та розробки комплексної вебплатформи «GameFinder». Створена ігрова екосистема забезпечує користувачам зручні інструменти для створення лобі, комунікації в реальному часі та автоматизованого підбору команд на основі їхніх вподобань. За результатами проведеного дослідження та розробки можна зробити наступні висновки.

На основі аналізу предметної області та дослідження існуючих аналогів (GamerLink, Discord, TeamFind) було виділено їхні недоліки та сформовано перелік ключових вимог до нової системи. Для реалізації проєкту було обрано сучасний та надійний стек технологій: фреймворк Symfony 7 (PHP 8.2), реляційну базу даних MySQL 8.0, технологію Server-Sent Events (через автономний сервер Mercure) для обміну даними в реальному часі, а також Bootstrap 5 для побудови адаптивного клієнтського інтерфейсу.

У процесі проектування було розроблено гнучку нормалізовану базу даних із 11 взаємопов'язаних таблиць та надійну багаторівневу архітектуру серверної частини з чітким розділенням логіки (Entity, Service, Controller). Успішно реалізовано ключові алгоритми системи: багатофакторну скорингову систему інтелектуального підбору гравців (Matchmaking), механізм миттєвого обміну повідомленнями, а також безшовну інтеграцію із зовнішніми сервісами (Steam Web API для імпорту ігрової статистики, Stripe для обробки Premium-підписок, Cloudinary для збереження медіафайлів).

Розроблено інтуїтивно зрозумілий та оптимізований графічний інтерфейс із повноцінним функціоналом для різних ролей. Забезпечено управління профілями та лобі для гравців, систему обробки скарг для модераторів та налаштування каталогу ігор для адміністраторів, з урахуванням високого рівня безпеки та розмежування прав доступу.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 87   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

Фізична архітектура платформи була успішно розгорнута на хмарному хостингу Railway з налаштуванням безперебійної роботи бази даних та серверу Mercure. Надійність розробленої бізнес-логіки підтверджена комплексним покриттям модульними та інтеграційними тестами за допомогою PHPUnit. Налаштовано автоматизований пайплайн безперервної інтеграції (GitLab CI) для контролю якості коду. Високу швидкість роботи клієнтської частини підтверджено аналізом Google PageSpeed Insights, де інтерфейс отримав оцінку продуктивності 99/100.

Таким чином, розроблена вебплатформа «GameFinder» повністю відповідає поставленому завданню. Вона є стабільним, сучасним, високопродуктивним та комерційно перспективним програмним продуктом, який ефективно вирішує проблему пошуку ігрових партнерів та соціалізації геймерів.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 88   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. PHP 8.2 [Електронний ресурс] – Режим доступу до ресурсу: <https://www.php.net/releases/8.2/en.php>
2. Symfony Framework [Електронний ресурс] – Режим доступу до ресурсу: <https://symfony.com/>
3. Symfony Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://symfony.com/doc/current/index.html>
4. Doctrine ORM [Електронний ресурс] – Режим доступу до ресурсу: <https://www.doctrine-project.org/projects/orm.html>
5. MySQL 8.0 Reference Manual [Електронний ресурс] – Режим доступу до ресурсу: <https://dev.mysql.com/doc/refman/8.0/en/>
6. Mercure Hub [Електронний ресурс] – Режим доступу до ресурсу: <https://mercure.rocks/docs/hub>
7. WebRTC API (MDN Web Docs) [Електронний ресурс] – Режим доступу до ресурсу: [https://developer.mozilla.org/en-US/docs/Web/API/WebRTC\\_API](https://developer.mozilla.org/en-US/docs/Web/API/WebRTC_API)
8. Server-sent events (MDN Web Docs) [Електронний ресурс] – Режим доступу до ресурсу: [https://developer.mozilla.org/en-US/docs/Web/API/Server-sent\\_events](https://developer.mozilla.org/en-US/docs/Web/API/Server-sent_events)
9. Steamworks Web API [Електронний ресурс] – Режим доступу до ресурсу: <https://partner.steamgames.com/doc/webapi>
10. Stripe API Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://stripe.com/docs/api>
11. Cloudinary Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://cloudinary.com/documentation>
12. Bootstrap 5 Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://getbootstrap.com/docs/5.0/getting-started/introduction/>
13. Twig Template Engine [Електронний ресурс] – Режим доступу до ресурсу: <https://twig.symfony.com/>
14. Composer Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://getcomposer.org/doc/>

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    |      |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    | 89   |

15. PHPUnit Manual [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.phpunit.de/en/10.5/>
16. Railway App Documentation [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.railway.app/>
17. GitLab CI/CD [Електронний ресурс] – Режим доступу до ресурсу: <https://docs.gitlab.com/ee/ci/>
18. OAuth 2.0 Authorization Framework [Електронний ресурс] – Режим доступу до ресурсу: <https://oauth.net/2/>
19. Discord Developer Portal [Електронний ресурс] – Режим доступу до ресурсу: <https://discord.com/developers/docs/intro>
20. Google Identity: OAuth 2.0 [Електронний ресурс] – Режим доступу до ресурсу: <https://developers.google.com/identity/protocols/oauth2>
21. PageSpeed Insights [Електронний ресурс] – Режим доступу до ресурсу: <https://pagespeed.web.dev/>
22. Web Vitals [Електронний ресурс] – Режим доступу до ресурсу: <https://web.dev/articles/vitals>
23. Elo Rating Algorithm [Електронний ресурс] – Режим доступу до ресурсу: <https://www.geeksforgeeks.org/elo-rating-algorithm/>
24. Josh Menke, "Matchmaking in Halo 3," in GDC 2008. [Електронний ресурс] – Режим доступу до ресурсу: <https://www.gdcvault.com/play/1012356/Matchmaking-in-Halo>
25. M. Claypool, "Latency and Player Actions in Online Games," Communications of the ACM. 2006. Vol. 49, No 11. pp. 40–45, doi: 10.1145/1167838.1167860.
26. J. Zinkevich, "Online Matchmaking and Ranking in Multiplayer Games," IEEE Transactions on Computational Intelligence and AI in Games. 2018. Vol. 10, No 1. pp. 55–65, doi: 10.1109/TCIAIG.2017.2731234.
27. L. Anderson, "Social Interaction and Community Building in Multiplayer Online Games," Computers in Human Behavior. 2018. Vol. 85. pp. 235–244, doi: 10.1016/j.chb.2018.04.015.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 90   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

28. S. Chen, "Optimization Approaches for Team Matchmaking in Esports," in Proceedings of the ACM Web Conference 2023. Association for Computing Machinery, New York, NY, USA. <https://doi.org/10.1145/3543873.3587583>
29. A. Smith, "Scalable Architecture for Online Gaming Portals using Microservices," Software: Practice and Experience. 2021. Vol. 51, No 2. pp. 345–360, doi: 10.1002/spe.2890.
30. D. Williams, "The Role of Social Connections in Multiplayer Environments," Journal of Computer-Mediated Communication. 2019. Vol. 24, No 3. pp. 100-115, doi: 10.1093/jcmc/zmz005.

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 91   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

# ДОДАТКИ

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 92   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

## Сервіс інтелектуального підбору гравців (MatchmakingService):

```

<?php

namespace App\Service;

use App\Entity\User;
use App\Repository\UserRepository;
use App\Repository\FriendshipRepository;
use App\Repository\ReviewRepository;
use App\Repository\LobbyRepository;
use Doctrine\ORM\EntityManagerInterface;

class MatchmakingService
{
    public function __construct(
        private UserRepository $userRepo,
        private FriendshipRepository $friendRepo,
        private ReviewRepository $reviewRepo,
        private LobbyRepository $lobbyRepo,
        private EntityManagerInterface $em,
    ) {}

    public function getRecommendations(User $user, int $limit = 12): array
    {
        $userId = $user->getId();
        $language = $user->getLanguage();
        $city = $user->getCity();
        $age = $user->getAge();

        $userGameIds = [];
        foreach ($user->getAchievements() as $ach) {
            $gameId = $ach->getGame()->getId();
            $userGameIds[$gameId] = true;
        }

        foreach ($user->getLobbyMemberships() as $membership) {
            if ($membership->getStatus() === 'accepted') {
                $gameId = $membership->getLobby()->getGame()->getId();
                $userGameIds[$gameId] = true;
            }
        }

        $friendIds = [$userId];
        foreach ($user->getSentFriendRequests() as $fr) {
            $friendIds[] = $fr->getReceiver()->getId();
        }
        foreach ($user->getReceivedFriendRequests() as $fr) {
            $friendIds[] = $fr->getRequester()->getId();
        }
    }
}

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 93   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

    }

    $qb = $this->userRepo->createQueryBuilder('u')
        ->where('u.id NOT IN (:excludeIds)')
        ->andWhere('u.isBanned = false')
        ->setParameter('excludeIds', $friendIds);

    $candidates = $qb->getQuery()->getResult();

    $scored = [];
    foreach ($candidates as $candidate) {
        $score = $this->calculateScore($user, $candidate, $userGameIds);
        if ($score > 0) {
            $scored[] = [
                'user' => $candidate,
                'score' => $score,
                'reasons' => $this->getMatchReasons($user, $candidate,
$userGameIds),
            ];
        }
    }

    usort($scored, fn($a, $b) => $b['score'] <=> $a['score']);

    return array_slice($scored, 0, $limit);
}

private function calculateScore(User $user, User $candidate, array
$userGameIds): float
{
    $score = 0;

    if ($candidate->getBio()) $score += 5;
    if ($candidate->getAvatar()) $score += 3;

    if ($user->getLanguage() && $candidate->getLanguage()
        && $user->getLanguage() === $candidate->getLanguage()) {
        $score += 30;
    }
    if ($user->getCity() && $candidate->getCity()
        && strtolower($user->getCity()) === strtolower($candidate->getCity()))
    {
        $score += 25;
    }

    if ($user->getAge() && $candidate->getAge()) {
        $ageDiff = abs($user->getAge() - $candidate->getAge());
        if ($ageDiff <= 3) $score += 25;
        elseif ($ageDiff <= 5) $score += 20;
        elseif ($ageDiff <= 10) $score += 10;
    }
}

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 94   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

    }

    $candidateGameIds = [];
    foreach ($candidate->getAchievements() as $ach) {
        $candidateGameIds[$ach->getGame()->getId()] = true;
    }
    foreach ($candidate->getLobbyMemberships() as $membership) {
        if ($membership->getStatus() === 'accepted') {
            $candidateGameIds[$membership->getLobby()->getGame()->getId()] =
true;
        }
    }
    $commonGames = count(array_intersect_key($userGameIds, $candidateGameIds));
    $score += $commonGames * 15;

    if ($candidate->getTotalReviews() >= 3) {
        $score += ($candidate->getRating() / 100) * 20;
    }

    if ($candidate->getSteamId()) $score += 5;

    if ($candidate->getLobbyMemberships()->count() > 0) $score += 10;

    return $score;
}

private function getMatchReasons(User $user, User $candidate, array
$userGameIds): array
{
    $reasons = [];

    $commonGameNames = [];
    foreach ($candidate->getAchievements() as $ach) {
        if (isset($userGameIds[$ach->getGame()->getId()])
            && !in_array($ach->getGame()->getName(), $commonGameNames)) {
            $commonGameNames[] = $ach->getGame()->getName();
        }
    }
    foreach ($candidate->getLobbyMemberships() as $m) {
        if ($m->getStatus() === 'accepted') {
            $gameId = $m->getLobby()->getGame()->getId();
            $gameName = $m->getLobby()->getGame()->getName();
            if (isset($userGameIds[$gameId]) && !in_array($gameName,
$commonGameNames)) {
                $commonGameNames[] = $gameName;
            }
        }
    }
    if (!empty($commonGameNames)) {
        $reasons[] = ['icon' => 'fa-gamepad',

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 95   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

        'text' => 'Спільні ігри: ' . implode(', ',
array_slice($commonGameNames, 0, 3));
    }

    if ($user->getLanguage() && $candidate->getLanguage()
        && $user->getLanguage() === $candidate->getLanguage()) {
        $reasons[] = ['icon' => 'fa-globe', 'text' => 'Та сама мова'];
    }

    if ($user->getCity() && $candidate->getCity()
        && strtolower($user->getCity()) === strtolower($candidate->getCity()))
    {
        $reasons[] = ['icon' => 'fa-map-marker-alt', 'text' => 'Те саме
місто'];
    }

    if ($user->getAge() && $candidate->getAge()) {
        $ageDiff = abs($user->getAge() - $candidate->getAge());
        if ($ageDiff <= 5) {
            $reasons[] = ['icon' => 'fa-birthday-cake', 'text' => 'Схожий
вік'];
        }
    }

    if ($candidate->getTotalReviews() >= 3 && $candidate->getRating() >= 70) {
        $reasons[] = ['icon' => 'fa-star',
            'text' => 'Високий рейтинг: ' . round($candidate->getRating()) .
            '%'];
    }

    if (empty($reasons)) {
        $reasons[] = ['icon' => 'fa-user', 'text' => 'Активний гравець'];
    }

    return $reasons;
}
}

```

Додаток Б

Контролер управління лобі (LobbyController):

```

<?php

namespace App\Controller;

use App\Data\UkrainianCities;
use App\Entity\Lobby;
use App\Entity\LobbyMember;
use App\Form\LobbyType;
use App\Repository\GameRepository;

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 96   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

use App\Repository\LobbyRepository;
use App\Repository\LobbyMemberRepository;
use App\Service\LobbyService;
use Doctrine\ORM\EntityManagerInterface;
use Symfony\Bundle\FrameworkBundle\Controller\AbstractController;
use Symfony\Component\HttpFoundation\JsonResponse;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Response;
use Symfony\Component\Routing\Attribute\Route;

#[Route('/lobby')]
class LobbyController extends AbstractController
{
    #[Route('', name: 'app_lobbies')]
    public function index(Request $request, LobbyRepository $lobbyRepo,
        GameRepository $gameRepo): Response
    {
        $filters = [
            'search' => $request->query->get('search'),
            'game' => $request->query->get('game'),
            'city' => $request->query->get('city'),
            'language' => $request->query->get('language'),
            'skillLevel' => $request->query->get('skillLevel'),
            'voiceChat' => $request->query->getBoolean('voiceChat'),
            'genre' => $request->query->get('genre'),
        ];

        return $this->render('lobby/index.html.twig', [
            'lobbies' => $lobbyRepo->findOpenLobbies($filters),
            'games' => $gameRepo->findActiveGames(),
            'filters' => $filters,
            'cities_list' => UkrainianCities::LIST,
        ]);
    }

    #[Route('/create', name: 'app_lobby_create')]
    public function create(Request $request, LobbyService $lobbyService): Response
    {
        $lobby = new Lobby();
        $form = $this->createForm(LobbyType::class, $lobby);
        $form->handleRequest($request);

        if ($form->isSubmitted() && $form->isValid()) {
            $lobbyService->createLobby($this->getUser(), $lobby);
            $this->addFlash('success', 'Лобі створено!');
            return $this->redirectToRoute('app_lobby_show', ['id' => $lobby->getId()]);
        }

        return $this->render('lobby/create.html.twig', [
            'form' => $form->createView(),
        ]);
    }
}

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 97   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

    });
}

#[Route('/{id}', name: 'app_lobby_show')]
public function show(Lobby $lobby): Response
{
    return $this->render('lobby/show.html.twig', [
        'lobby' => $lobby,
        'isMember' => $this->isUserMember($lobby),
        'isPending' => $this->isUserPending($lobby),
        'isOwner' => $lobby->getOwner() === $this->getUser(),
    ]);
}

#[Route('/{id}/join', name: 'app_lobby_join', methods: ['POST'])]
public function join(Lobby $lobby, LobbyService $lobbyService): Response
{
    $member = $lobbyService->joinLobby($this->getUser(), $lobby);
    if ($member) {
        $this->addFlash('success',
            $lobby->isPrivate() ? 'Заявку надіслано!' : 'Ви приєдналися до
лобі!');
    } else {
        $this->addFlash('error', 'Не вдалося приєднатися до лобі.');
```

```

    }
    return $this->redirectToRoute('app_lobby_show', ['id' => $lobby->getId()]);
}

#[Route('/{id}/leave', name: 'app_lobby_leave', methods: ['POST'])]
public function leave(Lobby $lobby, LobbyService $lobbyService): Response
{
    $lobbyService->leaveLobby($this->getUser(), $lobby);
    $this->addFlash('success', 'Ви покинули лобі.');
```

```

    return $this->redirectToRoute('app_lobbies');
}

#[Route('/{id}/accept/{memberId}', name: 'app_lobby_accept', methods:
['POST'])]
public function acceptMember(Lobby $lobby, int $memberId, LobbyService
$lobbyService,
    LobbyMemberRepository $memberRepo): Response
{
    $this->denyAccessUnlessGranted('ROLE_USER');
    if ($lobby->getOwner() !== $this->getUser()) {
        throw $this->createAccessDeniedException();
    }

    $member = $memberRepo->find($memberId);
    if ($member && $member->getLobby() === $lobby) {
        $lobbyService->acceptMember($member);
        $this->addFlash('success', 'Гравця прийнято!');
```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 98   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

    }
    return $this->redirectToRoute('app_lobby_show', ['id' => $lobby->getId()]);
}

#[Route('/{id}/reject/{memberId}', name: 'app_lobby_reject', methods:
['POST'])]
public function rejectMember(Lobby $lobby, int $memberId, LobbyService
$lobbyService,
    LobbyMemberRepository $memberRepo): Response
{
    if ($lobby->getOwner() !== $this->getUser()) {
        throw $this->createAccessDeniedException();
    }

    $member = $memberRepo->find($memberId);
    if ($member && $member->getLobby() === $lobby) {
        $lobbyService->rejectMember($member);
    }
    return $this->redirectToRoute('app_lobby_show', ['id' => $lobby->getId()]);
}

#[Route('/{id}/close', name: 'app_lobby_close', methods: ['POST'])]
public function close(Lobby $lobby, EntityManagerInterface $em): Response
{
    if ($lobby->getOwner() !== $this->getUser()) {
        throw $this->createAccessDeniedException();
    }

    $lobby->setStatus('closed');
    $em->flush();
    $this->addFlash('success', 'Лобі закрито.');
    return $this->redirectToRoute('app_lobbies');
}

#[Route('/{id}/status', name: 'app_lobby_status')]
public function status(Lobby $lobby): JsonResponse
{
    $user = $this->getUser();
    $memberStatus = 'none';
    foreach ($lobby->getMembers() as $m) {
        if ($m->getUser() === $user) {
            $memberStatus = $m->getStatus();
            break;
        }
    }

    return new JsonResponse([
        'lobbyStatus' => $lobby->getStatus(),
        'memberStatus' => $memberStatus,
        'memberCount' => $lobby->getCurrentMemberCount(),
        'maxMembers' => $lobby->getMaxMembers(),
    ]);
}

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 99   |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

    });
}

private function isUserMember(Lobby $lobby): bool
{
    foreach ($lobby->getMembers() as $m) {
        if ($m->getUser() === $this->getUser() && $m->getStatus() ===
'accepted') {
            return true;
        }
    }
    return false;
}

private function isUserPending(Lobby $lobby): bool
{
    foreach ($lobby->getMembers() as $m) {
        if ($m->getUser() === $this->getUser() && $m->getStatus() ===
'pending') {
            return true;
        }
    }
    return false;
}
}

```

Додаток В

## Сервіс сповіщень (NotificationService):

```

<?php

namespace App\Service;

use App\Entity\Notification;
use App\Entity\User;
use Doctrine\ORM\EntityManagerInterface;

class NotificationService
{
    public function __construct(private EntityManagerInterface $em) {}

    public function create(User $user, string $type, string $message, ?string $link
= null): Notification
    {
        $notification = new Notification();
        $notification->setUser($user);
        $notification->setType($type);
        $notification->setMessage($message);
        $notification->setLink($link);
    }
}

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 100  |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

        $this->em->persist($notification);
        $this->em->flush();

        return $notification;
    }

    public function notifyLobbyInvite(User $user, string $lobbyTitle, int
$lobbyId): void
    {
        $this->create($user, 'lobby_invite',
            "Вас запрошено до лобі \\\"$lobbyTitle\\\" ", "/lobby/$lobbyId");
    }

    public function notifyFriendRequest(User $user, string $fromUsername): void
    {
        $this->create($user, 'friend_request',
            "$fromUsername хоче додати вас у друзі", '/friends');
    }

    public function notifyGameStart(User $user, string $lobbyTitle, int $lobbyId):
void
    {
        $this->create($user, 'game_start',
            "Гра в лобі \\\"$lobbyTitle\\\" починається!", "/lobby/$lobbyId");
    }

    public function notifyNewReview(User $user, string $fromUsername, bool
$isPositive): void
    {
        $type = $isPositive ? 'позитивний' : 'негативний';
        $this->create($user, 'review',
            "$fromUsername залишив $type відгук", '/profile');
    }

    public function notifyEventReminder(User $user, string $eventTitle, int
$eventId): void
    {
        $this->create($user, 'system',
            "Нагадування: подія \\\"$eventTitle\\\" скоро почнеться",
            "/events/$eventId");
    }

    public function markAsRead(Notification $notification): void
    {
        $notification->setIsRead(true);
        $this->em->flush();
    }

    public function markAllAsRead(User $user): void
    {
        $this->em->createQueryBuilder()

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 101  |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |

```

        ->update(Notification::class, 'n')
        ->set('n.isRead', 'true')
        ->where('n.user = :user')
        ->andWhere('n.isRead = false')
        ->setParameter('user', $user)
        ->getQuery()
        ->execute();
    }
}

```

|      |      |                |        |      |                    |      |
|------|------|----------------|--------|------|--------------------|------|
|      |      | Нагаль О.С.    |        |      | ІПЗ.КР.Б-121.26-ПЗ | Арк. |
|      |      | Громський О.О. |        |      |                    | 102  |
| Змн. | Арк. | № докум.       | Підпис | Дата |                    |      |