ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Інформаційна система управління замовленнями на маркетплейсі фріланс-послуг у середовищі моддингу Garry's Mod»

на здобуття освітнього ступеня бакалавра
зі спеціальності 126 Інформаційні системи та технології
освітньо-професійної програми Інформаційні системи та технології

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

	Данило ТИЩЕНКО
(підпис)	*Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач вищої освіти гр. ІСД-42
Данило Тищенко
Ім'я, ПРІЗВИЩЕ

Керівник:
PhD Валентина ДАНИЛЬЧЕНКО
Ім'я, ПРІЗВИЩЕ

Рецензент:
науковий ступінь, вчене звання
Ім'я, ПРІЗВИЩЕ

Київ 2026

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Інформаційних систем та технологій
Ступінь вищої освіти бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітньо-професійна програма Інформаційні системи та технології

ЗАТВЕРДЖУЮ
Завідувач кафедрою ІСТ
_____Каміла СТОРЧАК
«____» _____2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Тищенко Данило Володимирович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Інформаційна система управління замовленнями на маркетплейсі фріланс-послуг у середовищі моддингу Garry's Mod

керівник кваліфікаційної роботи Валентина ДАНИЛЬЧЕНКО, доктор філософії,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологій
від «20» 02 2026р. № 51

2. Строк подання кваліфікаційної роботи «01» 06 2026р.

3. Вихідні дані до кваліфікаційної роботи: наукова та технічна література з проєктування інформаційних систем, документація з веброзробки, матеріали щодо фриланс-платформ і маркетплейсів цифрових послуг, документація Garry's Mod/Lua, результати аналізу існуючих платформ Fiverr, Upwork, GmodStore.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1. Аналіз предметної області, існуючих фриланс-платформ та формування вимог до системи.
   2. Вибір технологічного стеку, проєктування архітектури веб-платформи та моделювання її функціоналу.
   3. Розробка, інтеграція, тестування та оцінка ефективності інформаційної системи

управління замовленнями.

5. Перелік ілюстративного матеріалу: *презентація*

6. Дата видачі завдання «20» 02 2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та постановка задачі дослідження.	11.03.2026	
2	Дослідження існуючих рішень і визначення вимог до системи.	24.03.2026	
3	Вибір технологічного стеку та проєктування архітектури платформи.	12.04.2026	
4	Розробка основних функціональних модулів веб-платформи.	07.05.2026	
5	Тестування системи та оцінка результатів роботи.	13.05.2026	
6	Розробка демонстраційних матеріалів.	15.05.2026	
7	Попередній захист дипломної роботи.	19.05.2026	
8	Подання роботи в деканат.	01.06.2026	

Здобувач вищої освіти _____ Данило ТИЩЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Валентина ДАНИЛЬЧЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 86 стор., 30 рис., 4 табл., 30 джерел.

Мета роботи – Розробка веб-платформи для автоматизації управління замовленнями та підтримки взаємодії між замовниками й виконавцями у сфері Garry’s Mod-фрилансу.
Об’єкт дослідження – Процес взаємодії між замовниками та виконавцями у сфері фриланс-послуг у середовищі Garry’s Mod.
Предмет дослідження – Методи та засоби проєктування і реалізації інформаційної системи управління замовленнями та автоматизованого підбору виконавців.

У межах дипломної роботи було проведено аналіз предметної області та існуючих платформ, сформовано вимоги до системи та спроєктовано архітектуру веб-платформи. Було реалізовано основні функціональні модулі, розроблено алгоритм автоматизованого підбору виконавців і проведено тестування системи.
Результатом роботи стала спеціалізована інформаційна система, яка може використовуватись для організації фриланс-взаємодії у середовищі моддингу Garry’s Mod.

КЛЮЧОВІ СЛОВА: ВЕБРОЗРОБКА, МАРКЕТПЛЕЙС, ЦИФРОВІ ПОСЛУГИ, ФРИЛАНС-ПЛАТФОРМА, УПРАВЛІННЯ ЗАМОВЛЕННЯМИ.

ЗМІСТ

ВСТУП 8
РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ 11
1.1 Актуальність автоматизації бізнес-процесів у сфері фрилансу 11
1.2 Аналіз існуючих рішень та платформ для взаємодії замовників та виконавців 17
1.3 Дослідження та формування портрету цільової аудиторії 23
1.4 Постановка задачі автоматизації підбору виконавців та управління проєктами і визначення функціональних та нефункціональних вимог до веб-платформи 31
РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЧНОГО СТЕКУ ТА ПРОЄКТУВАННЯ АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ 38
2.1 Вибір та обґрунтування програмних засобів, СКБД та зовнішніх інструментів для реалізації системи 38
2.2 Проєктування архітектури веб-платформи 48
2.3 Моделювання функціоналу системи 56
2.3 Обґрунтування рішень щодо масштабування та безпеки платформи 64
РОЗДІЛ 3. РОЗРОБКА, ІНТЕГРАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ПЛАТФОРМИ МАРКЕТПЛЕЙСУ ФРІЛАНС-ПОСЛУГ 70
3.1. Опис реалізації основних функціональних модулів 70
3.2. Розробка алгоритму автоматизованого підбору виконавців 78
3.3. Налаштування, конфігурація та розгортання веб-платформи 84
3.4. Тестування розробленої системи, оцінка ефективності впровадження та перспектив розвитку 87
ВИСНОВКИ 92
ПЕРЕЛІК ПОСИЛАНЬ 95
ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація) 97

ВСТУП

У сучасних умовах стрімкого розвитку цифрових технологій та онлайн-економіки особливої актуальності набувають інформаційні системи, спрямовані на автоматизацію взаємодії між замовниками та виконавцями у сфері фриланс-послуг. Розвиток інтернет-комунікацій, поширення дистанційної зайнятості та зростання популярності цифрових платформ призвели до формування окремого сегмента ринку, у межах якого співпраця між учасниками здійснюється повністю в онлайн-середовищі. Одним із таких напрямів є індустрія моддингу відеоігор, де користувачі створюють, модифікують та підтримують додатковий контент для існуючих ігрових платформ.

Особливе місце серед моддингових платформ займає Garry’s Mod – популярне sandbox-середовище, яке дозволяє створювати власні ігрові режими, серверні системи, інтерфейси, моделі, текстури та інші елементи контенту. Навколо Garry’s Mod сформувалась велика міжнародна спільнота розробників, дизайнерів і власників серверів, між якими постійно виникає потреба у виконанні фриланс-замовлень. При цьому взаємодія між замовниками та виконавцями часто відбувається через Discord-сервери, форуми або приватні домовленості, що створює низку проблем, пов’язаних із пошуком виконавців, перевіркою репутації, контролем виконання робіт і безпечністю співпраці.

Існуючі універсальні фриланс-платформи, такі як Fiverr або Upwork, не враховують специфіку Garry’s Mod-спільноти та не забезпечують інструментів, орієнтованих на особливості моддингового середовища. Водночас спеціалізовані рішення для Garry’s Mod мають обмежений функціонал і здебільшого орієнтовані на продаж готового контенту, а не на організацію повноцінної взаємодії між замовниками та виконавцями. Унаслідок цього виникає потреба у створенні спеціалізованої інформаційної системи, яка забезпечить централізоване управління замовленнями, автоматизацію пошуку виконавців і підтримку репутаційної взаємодії між користувачами.

Актуальність теми полягає у необхідності автоматизації процесів організації фриланс-замовлень у Garry’s Mod-середовищі та створенні єдиної веб-платформи, здатної забезпечити структуровану взаємодію між учасниками спільноти. Реалізація подібної системи дозволяє спростити процес пошуку виконавців, підвищити рівень довіри між користувачами, оптимізувати управління замовленнями та створити основу для формування професійної цифрової екосистеми всередині моддингової спільноти.

Об’єктом дослідження є процес взаємодії між замовниками та виконавцями у сфері фриланс-послуг у середовищі Garry’s Mod.

Предметом дослідження є методи та засоби проєктування і реалізації інформаційної системи управління замовленнями та автоматизованого підбору виконавців для спеціалізованого фриланс-маркетплейсу.

Метою роботи є розробка веб-платформи для автоматизації управління замовленнями та підтримки взаємодії між замовниками й виконавцями у сфері Garry’s Mod-фрилансу.

Для досягнення поставленої мети у роботі вирішуються такі *задачі*:

- аналіз предметної області та існуючих платформ для організації фриланс-послуг;
- визначення функціональних і нефункціональних вимог до веб-платформи;
- вибір технологій та програмних засобів реалізації системи;
- проєктування архітектури веб-платформи та структури бази даних;
- розробка алгоритму автоматизованого підбору виконавців;
- реалізація основних функціональних модулів системи;
- налаштування та розгортання програмного середовища;
- тестування реалізованої системи та оцінка ефективності її використання.
- тестування реалізованої системи та оцінка ефективності її використання.

Практична цінність роботи полягає у створенні працездатної веб-платформи, яка може використовуватися для організації фриланс-взаємодії в

Garry’s Mod-спільноті. Реалізована система дозволяє створювати та управляти замовленнями, подавати заявки, формувати репутацію користувачів і автоматизувати процес підбору виконавців. Архітектура платформи передбачає можливість подальшого масштабування та розширення функціоналу, включаючи інтеграцію платіжних систем, внутрішнього месенджера, розширеної модерації та рекомендаційних механізмів.

Апробація результатів. Основні положення та результати дослідження пройшли апробацію на:

1. VII Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT» (Київ, ДУІКТ, 20 квітня 2026 р.) Тези доповіді на тему «МОДЕЛЬ ОЦІНЮВАННЯ РЕПУТАЦІЇ КОРИСТУВАЧІВ У ЦИФРОВИХ СЕРВІСАХ ЯК СКЛАДОВА IoT-ЕКОСИСТЕМ» опублікований у збірнику матеріалів конференції (стор. 279)
2. VII Міжнародної наукової конференції «Період трансформаційних процесів в світовій науці: задачі та виклики» (Полтава, 22 червня 2026 р.) Тези доповіді на тему «СИСТЕМА РЕПУТАЦІЇ ЯК ЕЛЕМЕНТ ДОВІРИ В МАРКЕТПЛЕЙСАХ ЦИФРОВИХ ПОСЛУГ» опублікований у збірнику матеріалів конференції (стор. 316)

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1 Актуальність автоматизації бізнес-процесів у сфері фрилансу

Фриланс як форма організації праці набуває дедалі більшого значення в умовах цифровізації економіки, поширення віддаленої зайнятості та зростання попиту на гнучкі моделі взаємодії між замовниками й виконавцями. Якщо раніше фриланс часто розглядався як додаткове джерело доходу або тимчасовий формат зайнятості, то сьогодні він поступово перетворюється на повноцінну складову ринку праці, що охоплює широкий спектр професійних послуг: програмування, дизайн, адміністрування серверів, створення цифрового контенту, технічну підтримку, тестування, написання текстів, моделювання, розробку ігрових модифікацій тощо. Така трансформація безпосередньо пов’язана з розвитком онлайн-платформ, які забезпечують пошук виконавців, публікацію замовлень, комунікацію, оцінювання якості роботи, проведення оплат та формування репутації учасників ринку. У цьому контексті автоматизація бізнес-процесів у сфері фрилансу є не лише технічним вдосконаленням окремих операцій, а необхідною умовою ефективного функціонування цифрового посередницького середовища.

Актуальність дослідження підтверджується загальним зростанням онлайн-ринку праці. У роботі О. Кассі та В. Ледонвірти, присвяченій Online Labour Index, зазначено, що традиційні статистичні інструменти недостатньо пристосовані до вимірювання онлайн-зайнятості, оскільки значна частина таких трудових відносин відбувається через цифрові платформи та не завжди відображається у звичайних показниках ринку праці. Запропонований авторами індекс відстежує кількість проєктів і завдань, опублікованих на великих онлайн-платформах майже в реальному часі. У подальшому дослідженні Online Labour Index 2020 зазначено, що в період з 2016 до 2021 року попит на онлайн-фриланс зріс приблизно на 90%, що відповідає середньорічному темпу зростання близько 10% [1].

Географічний розподіл виконавців і замовників на онлайн-платформах демонструє глобальний характер фрилансової взаємодії: значна частина виконавців зосереджена в країнах Південної Азії та Східної Європи, тоді як основні джерела замовлень припадають на США, Велику Британію, Канаду та Австралію. Відповідний розподіл наведено на рисунку 1.1.

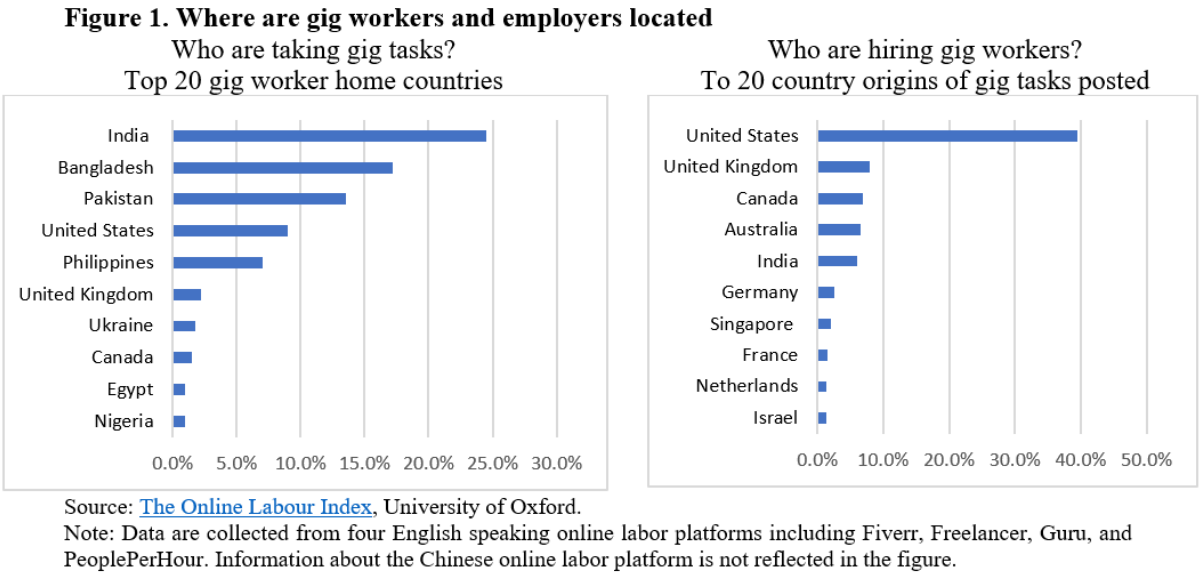


Рис 1.1 Географічний розподіл виконавців і замовників онлайн-gig роботи [2]

Також підкреслюється, що більшість онлайн-завдань зосереджена у сфері технологій, за якою йдуть графічний дизайн і написання текстів, що особливо важливо для цифрових та креативних ніш, близьких до середовища ігрового моддингу [3].

Розвиток онлайн-фрилансу створює передумови для появи спеціалізованих платформ, орієнтованих не лише на масовий ринок послуг, а й на вузькі професійні спільноти. Середовище моддингу Garry's Mod є прикладом такої нішевої цифрової екосистеми, у якій замовниками можуть виступати власники серверів, ігрові спільноти, RP-проєкти або комерційні команди, а виконавцями – розробники Lua/GLua, UI/UX-дизайнери, 3D-моделери, мапери, конфігуратори ігрових систем, технічні адміністратори та інші спеціалісти. Особливістю цього середовища є поєднання творчих, технічних і комунікаційних завдань, які часто виконуються

дистанційно, мають проєктний характер і потребують швидкого узгодження вимог між сторонами. За відсутності спеціалізованої інформаційної системи такі процеси зазвичай відбуваються неструктуровано: через мессенджер Discord, форуми, приватні повідомлення або неформальні домовленості. Це ускладнює контроль виконання замовлень, знижує прозорість взаємодії, підвищує ризик непорозумінь і не дозволяє системно накопичувати інформацію про якість роботи виконавців.

Потреба в автоматизації бізнес-процесів посилюється тим, що фрилансова взаємодія складається з багатьох повторюваних етапів: створення замовлення, опис вимог, пошук відповідного виконавця, оцінювання кандидатів, комунікація сторін, погодження умов, відстеження статусу виконання, приймання результату, формування відгуків і репутаційних показників. Узагальнену послідовність виконання фриланс-замовлення в межах інформаційної системи наведено на рисунку 1.2.

Рис 1.2 Узагальнена схема бізнес-процесу виконання фриланс-замовлення

У ручному або напів ручному форматі ці процеси залежать від суб’єктивної пам’яті учасників, неформальних домовленостей і зовнішніх каналів комунікації.

Автоматизована веб-платформа дає змогу перетворити такі дії на керований бізнес-процес із чіткою структурою даних, фіксацією статусів, збереженням історії взаємодії та використанням алгоритмів для підбору виконавців. Це особливо важливо для нішевих спільнот, де кількість активних фахівців може бути обмеженою, а репутація й попередній досвід мають вирішальне значення для довіри між сторонами.

Важливою характеристикою сучасного онлайн-фрилансу є зростання ролі алгоритмів у регулюванні взаємодії між замовниками та виконавцями. Дослідження показують, що віддалена gig-праця значною мірою формується під впливом платформного алгоритмічного контролю. Логіку взаємозв'язку між даними профілю виконавця, алгоритмічним підбором та репутаційною системою наведено на рисунку 1.3.

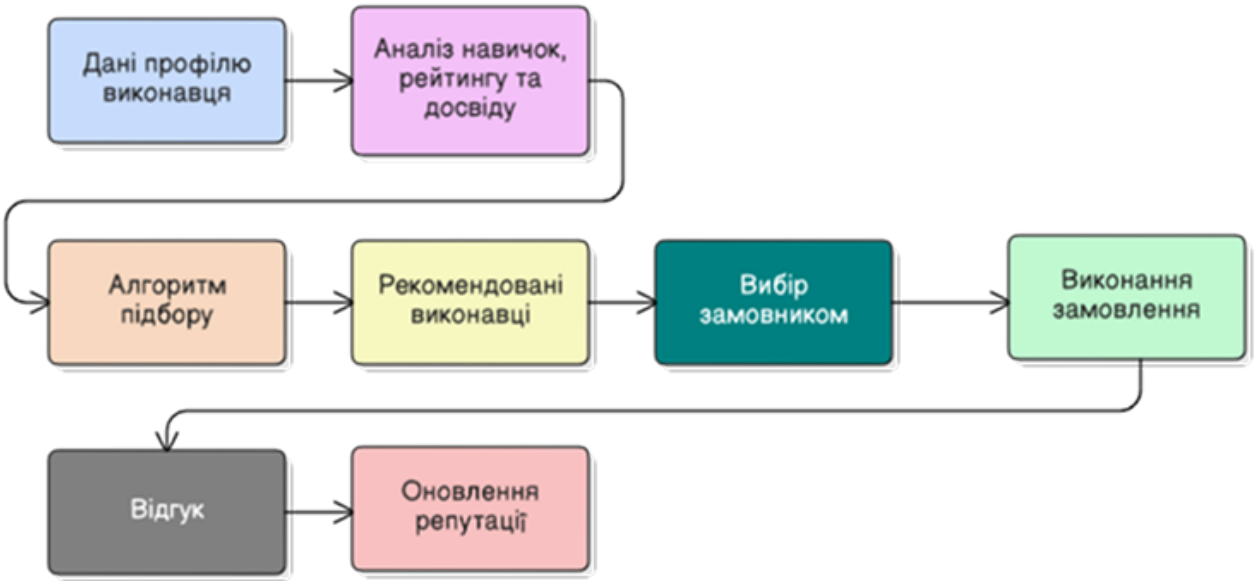


Рис 1.3 Роль алгоритмічного підбору та репутаційної системи у фриланс-платформі

Алгоритмічне управління може забезпечувати виконавцям гнучкість, автономію, різноманітність завдань і складність роботи, однак одночасно воно може призводити до низької оплати, соціальної ізоляції, нерегулярного графіка, перевтоми та виснаження [4]. Для інформаційної системи управління

замовленнями це означає, що алгоритми мають застосовуватися не лише як інструмент ранжування або контролю, а як засіб підвищення прозорості, справедливості й передбачуваності взаємодії. Зокрема, алгоритм автоматизованого підбору виконавців повинен враховувати не тільки рейтинг, а й релевантність навичок, досвід у конкретній категорії, завантаженість, історію виконання подібних замовлень та якість зворотного зв’язку.

Застосування алгоритмічних механізмів у фрилансових платформах безпосередньо пов’язане з проблемою довіри. У традиційній організації праці довіра частково забезпечується трудовими договорами, ієрархією управління, внутрішніми правилами компанії та безпосереднім контролем. У фрилансі, особливо в дистанційному форматі, ці механізми або відсутні, або значно послаблені. Тому цифрова платформа повинна компенсувати брак формальних гарантій через репутаційні системи, структуровані профілі, історію виконаних замовлень, відгуки, статуси верифікації та механізми фільтрації. Для ринку моддингу Garry’s Mod це має особливе значення, оскільки замовник часто не може заздалегідь оцінити реальну компетентність виконавця без перегляду портфоліо, попередніх робіт або відгуків інших учасників спільноти. Автоматизація дозволяє зменшити інформаційну асиметрію між сторонами та створити середовище, у якому рішення про співпрацю ґрунтується не лише на особистих рекомендаціях, а й на систематизованих даних.

Додатковим чинником актуальності є економічне зростання фрилансових платформ. За узагальненими даними Jobbers.io, у 2025 році вартість ринку freelance platform market оцінювалася приблизно у 8,39 млрд доларів США, а прогнозоване значення на 2029 рік становить близько 14-17 млрд доларів США.

Рис 1.4 Прогноз зростання ринку фриланс-платформ у 2025–2029 роках

Хоча такі оцінки необхідно розглядати з урахуванням методологічних відмінностей між джерелами, вони демонструють загальну тенденцію до розширення платформної інфраструктури фрилансу. У тому самому огляді зазначається, що вузька оцінка кількості працівників онлайн-gig платформ, наведена з посиланням на Світовий банк, становить 154-435 млн осіб, тоді як ширші показники самозайнятості включають значно більші групи працівників і не повинні прямо ототожнюватися з цифровим фрилансом [5]. Це підкреслює не лише масштаб явища, а й складність його вимірювання, що своєю чергою вказує на потребу в інформаційних системах, здатних накопичувати, структурувати й аналізувати дані про реальні процеси взаємодії на фрилансових ринках.

Автоматизація бізнес-процесів у сфері фрилансу також має важливе значення для підвищення ефективності управління проєктами. У випадку розробки модифікацій для Garry’s Mod типове замовлення може включати кілька етапів: уточнення технічного завдання, оцінювання складності, погодження вартості, реалізацію функціоналу, тестування на сервері, виправлення помилок і подальшу підтримку. Якщо ці етапи неформалізовані, виникають проблеми з контролем термінів, обсягом робіт і відповідальністю сторін. Інформаційна система може

забезпечити єдине середовище для створення замовлень, категоризації послуг, фіксації вимог, відстеження прогресу та зберігання результатів комунікації. Завдяки цьому зменшується залежність від зовнішніх каналів, підвищується прозорість виконання робіт і створюються передумови для подальшого аналізу ефективності платформи.

Окрему увагу слід приділити нефункціональним аспектам автоматизації, зокрема масштабованості, безпеці та надійності. Фрилансова платформа працює з персональними даними користувачів, історією замовлень, репутаційними оцінками, повідомленнями та потенційно фінансовою інформацією. Тому її розроблення потребує продуманого підходу до автентифікації, авторизації, розмежування ролей, захисту облікових записів і збереження цілісності даних. У нішевому середовищі, де значна частина взаємодії базується на довірі між учасниками спільноти, втрата або спотворення репутаційних даних може мати суттєві наслідки для користувачів. Саме тому автоматизація повинна охоплювати не лише зовнішній інтерфейс взаємодії, а й внутрішню логіку обробки даних, перевірку прав доступу, аудит дій і механізми запобігання зловживанням.

1.2 Аналіз існуючих рішень та платформ для взаємодії замовників та виконавців

Сучасний ринок фриланс-послуг значною мірою сформований навколо цифрових платформ, які виконують роль посередника між замовниками та виконавцями. Такі платформи забезпечують інфраструктуру для розміщення замовлень, пошуку спеціалістів, оцінювання їхнього досвіду, комунікації, укладання домовленостей та, у багатьох випадках, проведення оплати. Їх поява суттєво змінила характер взаємодії на ринку праці, оскільки замість неформального пошуку виконавців через особисті контакти або спільноти користувач отримує централізований інструмент для вибору спеціаліста на основі профілю, портфоліо, рейтингу, відгуків та попереднього досвіду. Водночас більшість великих фриланс-платформ орієнтована на максимально широкий ринок і не враховує специфіку

окремих професійних або ігрових спільнот, зокрема середовища моддингу Garry’s Mod.

Однією з найвідоміших глобальних платформ фрилансу є Fiverr. Вона позиціонується як маркетплейс фриланс-послуг, де замовники можуть знаходити виконавців у різних категоріях, зокрема у сфері програмування й технологій, графічного дизайну, цифрового маркетингу, написання текстів, відео, анімації, музики, бізнес-послуг, консалтингу та роботи з даними. Така структура демонструє універсальний характер платформи: вона не прив’язана до певної галузі, а намагається охопити максимально широкий спектр цифрових і креативних послуг. [6] Для замовника це є перевагою, оскільки він отримує доступ до великої кількості виконавців із різних країн та сфер діяльності. Для виконавця перевагою є можливість створити профіль, представити власні послуги та отримувати замовлення незалежно від географічного розташування.

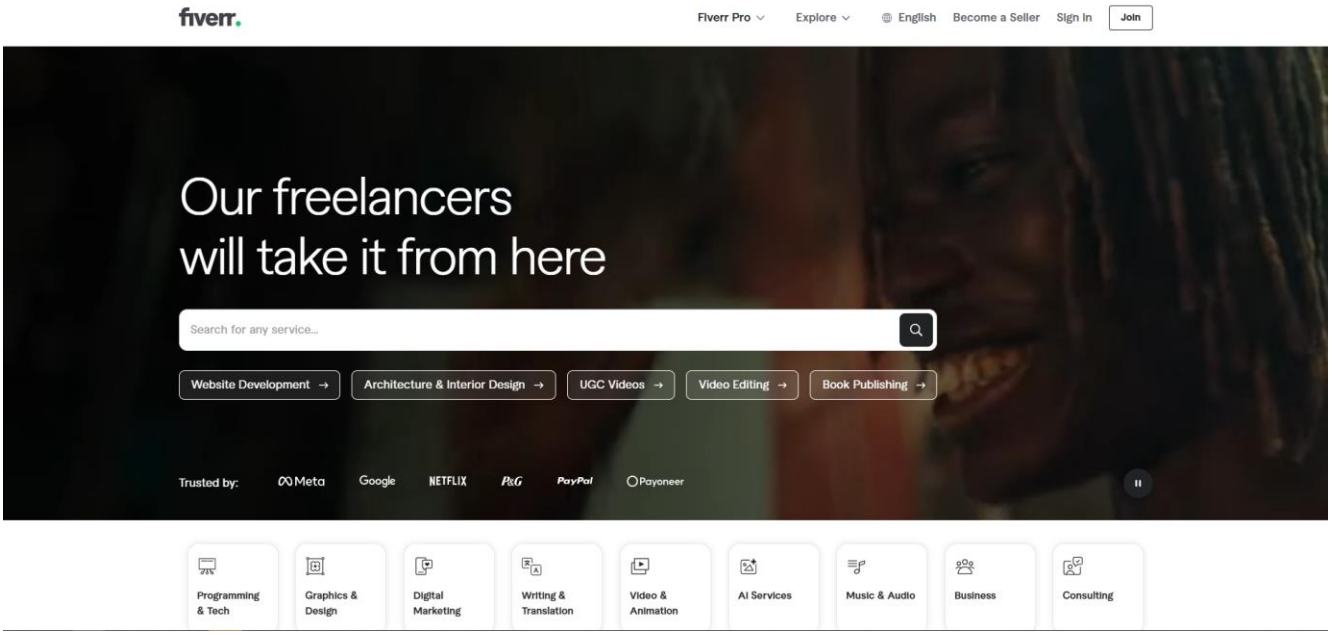


Рис 1.5 Домашня сторінка Fiverr

Однак універсальність Fiverr одночасно є його обмеженням у контексті вузькоспеціалізованих ніш. Платформа не орієнтована на специфічні вимоги Garry’s Mod-розробки, такі як досвід роботи з GLua (діалектом Lua), знання

доступних ігрових режимів, розробки інтерфейсів з використанням вбудованих бібліотек, HUD-систем, інтеграцій із ігровими серверами, оптимізації Lua-коду або особливостей розробки під конкретні ігрові структури. Через це пошук виконавця для вузької технічної задачі може бути ускладнений, а оцінювання компетентності фрилансера часто потребує додаткової перевірки поза межами самої платформи.

Іншим прикладом глобальної фриланс-платформи є Upwork. На відміну від Fiverr, який історично більше асоціюється з каталогом послуг, Upwork значною мірою побудований навколо моделі публікації вакансій або проєктів, подання пропозицій виконавцями та укладання контракту між сторонами. Згідно з описом платформи, виконавець може створити профіль, зазначити навички, портфоліо та бажану ставку, після чого шукати погодинні або фіксовані за вартістю замовлення й надсилати пропозиції клієнтам. [7] Для замовників Upwork пропонує публікацію замовлень, отримання пропозицій від фахівців, перегляд перевіреної історії роботи та відгуків, використання фільтрів пошуку, а також механізми захищених платежів і оплату за етапами або погодинними контрактами. [8] Такий підхід є більш наближеним до класичної моделі управління проєктами, оскільки замовник описує задачу, отримує заявки, порівнює кандидатів і обирає виконавця.

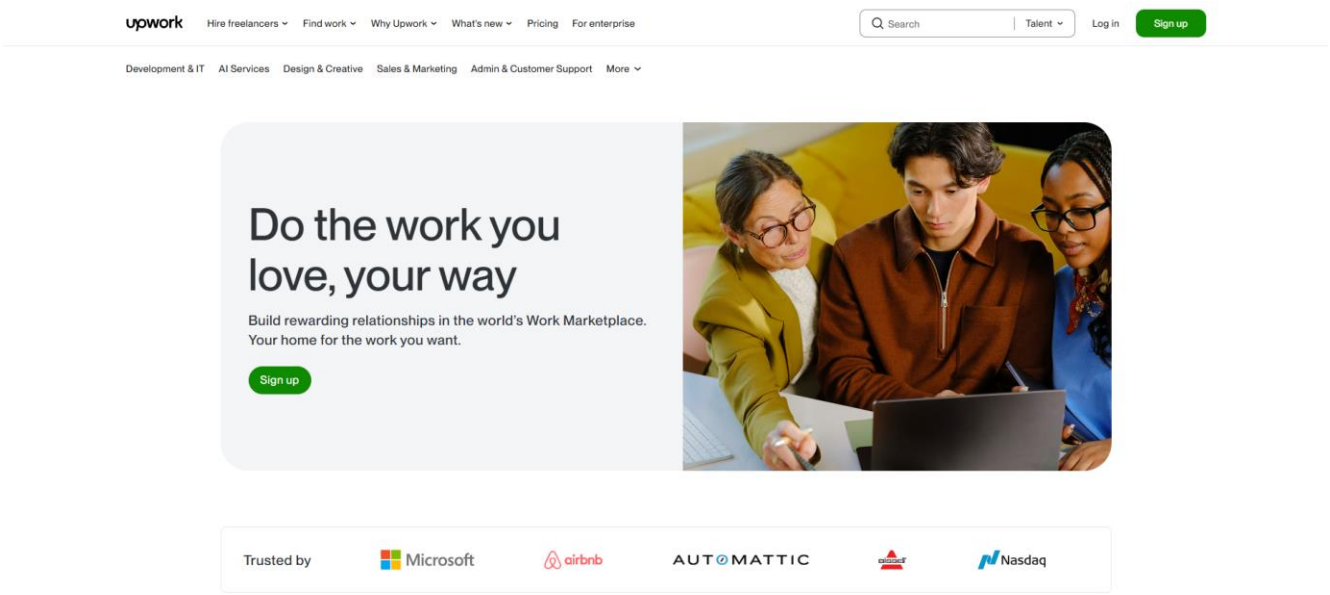


Рис 1.6 Домашня сторінка Upwork

Разом із тим Upwork, як і Fiverr, залишається універсальною платформою. Її перевага полягає у масштабності, розвинених механізмах довіри, великій базі користувачів, системі відгуків і підтримці різних форматів співпраці. Проте саме масштабність створює проблему для вузької ніші. Замовлення, пов’язані з моддингом Garry’s Mod, на таких платформах фактично розчиняються серед значно ширшого масиву завдань із веб-розробки, дизайну, геймдеву, адміністрування, маркетингу та інших напрямів. Через це замовнику складніше знайти виконавця, який має не лише загальні навички програмування або дизайну, а й практичний досвід роботи з конкретною ігровою екосистемою. Крім того, глобальні платформи зазвичай не мають спеціалізованої категоризації, яка була б природною для Garry’s Mod-спільноти. Відсутність такої предметної структури знижує точність пошуку й ускладнює автоматизований підбір виконавців.

У межах середовища Garry’s Mod найбільш релевантним спеціалізованим рішенням є GmodStore. Насамперед ця платформа відома як маркетплейс готових рішень для Garry’s Mod, де автори можуть публікувати та продавати вже створені продукти. На сторінці перегляду товарів GmodStore прямо описується як місце для пошуку великої кількості продуктів від авторів контенту, а користувачам пропонується переглядати маркетплейс або створювати власний продукт [9].

Рис 1.7 Домашня сторінка GmodStore

Така модель орієнтована передусім на продаж готових скриптів, систем, інтерфейсів, addon-рішень та інших цифрових продуктів, які можуть бути багаторазово придбані різними користувачами. Це суттєво відрізняється від моделі індивідуального фриланс-замовлення, де результат створюється під конкретні вимоги замовника, має унікальне технічне завдання та часто потребує прямої комунікації між сторонами.

Водночас GmodStore має окремий розділ Job Market, який дозволяє переглядати й публікувати замовлення. На сторінці перегляду робіт зазначено, що користувачі можуть переглядати замовлення, подаватися на ті, для яких вони мають відповідну кваліфікацію, а також публікувати власну роботу. Станом на момент перегляду сторінка містила понад 6400 замовлень, розподілених за категоріями DarkRP, Administration, Gamemode, Modelling, Mapping, VGUI, HUD, Artwork, Web, TTT, Prophunt, Loading screen, Donation та іншими. Наявність таких категорій є важливою перевагою GmodStore порівняно з глобальними платформами, оскільки вона відображає реальну термінологію й типові напрями робіт у Garry's Mod-спільноті. Для замовника це спрощує формулювання потреби, а для виконавця – пошук релевантних завдань у межах знайомої предметної області.

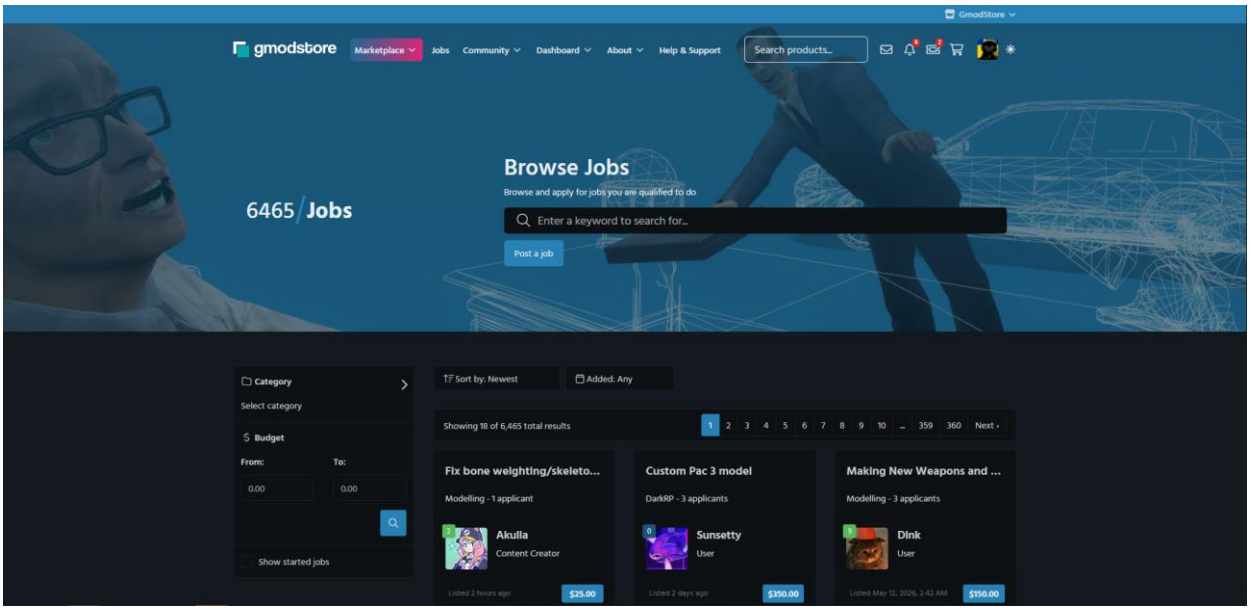


Рис 1.8 Розділ Job Market на GmodStore

Однак спеціалізація GmodStore не повністю вирішує проблему організації фриланс-взаємодії. Оскільки основною функцією платформи є маркетплейс готових продуктів, розділ пошуку виконавців має допоміжний характер. Це обмежує глибину функціоналу, необхідного саме для управління індивідуальними замовленнями. У контексті розроблюваної інформаційної системи важливими є не лише публікація списку робіт і фільтрація за категоріями, а й повноцінна підтримка життєвого циклу замовлення: створення структурованого технічного завдання, зіставлення вимог із навичками виконавців, автоматизований підбір кандидатів, контроль статусів, ведення історії комунікації, формування репутації саме за виконаними індивідуальними замовленнями, а також можливість аналізу попереднього досвіду виконавця в конкретній категорії. Якщо платформа першочергово орієнтована на продаж готових addon-рішень, то фриланс-модуль може не мати достатньої деталізації для таких процесів.

Порівняння Fiverr, Upwork і GmodStore дозволяє виділити кілька типів існуючих рішень. Перший тип – це глобальні універсальні платформи, які забезпечують розвинену інфраструктуру взаємодії, але не мають глибокої спеціалізації під конкретну ігрову або моддингову нішу. До цього типу належать Fiverr і Upwork. Їхня сила полягає у масштабі, довірі, наявності рейтингових механізмів, великій кількості користувачів і підтримці різних форматів оплати та співпраці. Їхня слабкість у межах цього дослідження полягає у недостатній адаптації до предметної області Garry’s Mod. Другий тип – це спеціалізовані галузеві рішення, представлені GmodStore, які краще розуміють структуру Garry’s Mod-ринку, але історично зосереджені переважно на продажі готових продуктів, а не на комплексному управлінні фриланс-замовленнями. Саме між цими двома типами платформ виникає функціональна прогалина, яку може заповнити спеціалізована інформаційна система для управління замовленнями на маркетплейсі фриланс-послуг у середовищі моддингу Garry’s Mod.

Для розроблюваної системи особливо важливо врахувати переваги й недоліки кожного з розглянутих підходів. Від глобальних платформ доцільно запозичити ідею структурованих профілів виконавців, портфоліо, відгуків,

рейтингу, публікації замовлень, подання заявок і фільтрації кандидатів. Від спеціалізованої моделі GmodStore доцільно використати предметну категоризацію, орієнтовану на реальні напрями Garry’s Mod-розробки: DarkRP, Gamemode, Mapping, VGUI, HUD, Web, Loading screen, Donation та інші. Водночас розроблювана система повинна розвивати ці підходи далі, зосереджуючись не на продажі готових скриптів, а саме на управлінні індивідуальними замовленнями, де ключовими є відповідність виконавця вимогам проєкту, прозорий процес виконання, репутаційна історія та зручна взаємодія між сторонами.

Отже, аналіз існуючих платформ показує, що на ринку вже присутні ефективні універсальні інструменти для організації фриланс-праці, а також окремі спеціалізовані рішення для Garry’s Mod-спільноти. Проте жодне з розглянутих рішень повною мірою не відповідає потребі у вузькоспеціалізований системі управління замовленнями саме для фриланс-послуг у середовищі моддингу Garry’s Mod. Глобальні платформи є занадто загальними й не враховують особливості цієї предметної області, тоді як GmodStore, попри наявність Job Market, залишається насамперед маркетплейсом готових рішень. Це обґрунтовує доцільність створення окремої веб-платформи, яка поєднує спеціалізацію під Garry’s Mod із функціональністю повноцінної системи управління фриланс-замовленнями.

1.3 Дослідження та формування портрету цільової аудиторії

Визначення цільової аудиторії є важливим етапом проєктування інформаційної системи, оскільки саме потреби, поведінкові особливості та проблеми майбутніх користувачів визначають функціональну структуру платформи. У межах теми дипломної роботи цільова аудиторія розглядається не як абстрактна група користувачів фриланс-сервісу, а як сукупність учасників конкретної цифрової екосистеми – спільноти моддингу Garry’s Mod. Така спільнота має власну термінологію, типові види робіт, сталі канали комунікації, очікування щодо якості послуг і специфічні критерії оцінювання компетентності виконавців. Тому для створення ефективної веб-платформи недостатньо орієнтуватися лише на

загальні принципи фриланс-маркетплейсів; необхідно враховувати особливості саме цієї предметної області.

Garry’s Mod є фізичною sandbox-грою, у якій відсутні наперед задані цілі, а користувачам надаються інструменти для створення власного ігрового досвіду, побудови об’єктів, взаємодії з контентом і гри на онлайн-серверах. Така відкрита структура гри сприяла формуванню навколо неї активної спільноти, орієнтованої на створення власних ігрових режимів, серверів, скриптів, інтерфейсів, моделей, мап, систем адміністрування та інших модифікацій. В офіційній документації Garry’s Mod окремо виділено ресурси для роботи з Lua API, що підтверджує технічну природу значної частини активності спільноти та наявність серед користувачів окремої групи розробників, які створюють функціональні розширення для гри [10].

З огляду на це, цільову аудиторію розроблюваної інформаційної системи доцільно поділити на дві основні групи: замовники фриланс-послуг та виконавці. До першої групи належать власники Garry’s Mod-серверів, адміністратори ігрових проєктів, керівники рольових спільнот, команди серверів, контент-менеджери та окремі користувачі, які потребують створення або доопрацювання певного цифрового продукту. Їхні замовлення можуть стосуватися розробки унікального функціоналу для сервера, створення HUD або VGUI-інтерфейсів, налаштування DarkRP-систем, написання SWEP, розробки ентіті, інтеграції донат-систем, створення loading screen, мапінгу, моделювання, оптимізації коду або виправлення помилок у вже наявних addon-рішеннях. Основною потребою цієї групи є пошук надійного виконавця, який не просто володіє загальними навичками програмування чи дизайну, а розуміє технічні та практичні особливості Garry’s Mod.

Замовники в межах такої системи зазвичай стикаються з кількома проблемами. По-перше, їм складно оцінити реальну кваліфікацію виконавця до початку співпраці, оскільки знання Lua або загального геймдеву не завжди означає досвід роботи з GLua, серверною архітектурою Garry’s Mod, існуючими gamemode-структурами або популярними addon-екосистемами. По-друге, значна частина комунікації у спільноті відбувається через Discord, форуми або приватні

повідомлення, що ускладнює фіксацію домовленостей, контроль статусу замовлення та збереження історії взаємодії. По-третє, замовник часто не має зручного інструменту для порівняння виконавців за релевантними критеріями: досвідом у конкретній категорії, попередніми роботами, відгуками, репутацією, завантаженістю та відповідністю бюджету. У результаті вибір виконавця може відбуватися на основі випадкових рекомендацій або суб'єктивного враження, що підвищує ризик неякісного виконання роботи.

Друга основна група цільової аудиторії – це виконавці, тобто фрилансери, які надають послуги у сфері моддингу Garry's Mod. До них можуть належати GLua-розробники, UI-розробники, дизайнери інтерфейсів, 3D-моделери, мапери, художники, веб-розробники, технічні адміністратори серверів та спеціалісти з інтеграцій. Для цієї групи ключовою потребою є доступ до релевантних замовлень, які відповідають їхнім навичкам, досвіду та очікуваному рівню оплати. На відміну від глобальних фриланс-платформ, де виконавець змушений конкурувати з великою кількістю спеціалістів у широких категоріях, спеціалізована система може забезпечити більш точне позиціонування фрилансера саме в межах Garry's Mod-екосистеми. Це дає змогу виконавцю демонструвати не лише загальне портфоліо, а й конкретні приклади робіт: створені HUD, меню персонажа, системи інвентарю, скрипти для DarkRP, карти, моделі, loading screen або інші модифікації.

Проблеми виконавців також мають специфічний характер. Однією з основних є складність системного пошуку замовлень, оскільки потенційні клієнти можуть публікувати запити в різних Discord-серверах, форумах, особистих повідомленнях або на окремих платформах без єдиної структури. Іншою проблемою є відсутність прозорої та накопичувальної репутаційної системи. Навіть якщо виконавець успішно завершив багато замовлень, ці результати можуть залишатися розпорошеними між різними спільнотами й не формувати єдиного підтвердженого профілю. Це особливо важливо для нових виконавців, яким складно отримати перше замовлення без видимої історії роботи, а також для досвідчених спеціалістів, які потребують інструменту для демонстрації власної надійності та професійного рівня. Тому система повинна підтримувати портфоліо,

категоризацію навичок, відгуки після виконання замовлень, показники успішності та механізми, які дозволяють замовникам об’єктивніше оцінювати кандидатів.

Окремо доцільно виділити допоміжну групу користувачів – адміністраторів платформи або модераторів. Їхня роль полягає у забезпеченні коректної роботи сервісу, перевірці спірних ситуацій, контролі порушень правил, обробці скарг, підтримці структури категорій і запобіганні зловживанням репутаційною системою. У фрилансовому середовищі, де значна частина взаємодії будується на довірі, модерація є важливою умовою стабільності платформи. Вона дозволяє зменшити ризики шахрайства, фейкових відгуків, неякісних профілів, дублювання замовлень або недобросовісної поведінки сторін. Для нішевої спільноти Garry’s Mod ця функція має особливе значення, оскільки користувачі часто взаємодіють у межах обмеженого професійного кола, де репутаційні конфлікти можуть мати довготривалі наслідки.

Портрет цільової аудиторії також можна сформувати через аналіз мотивації користувачів. Замовники зацікавлені у зменшенні часу на пошук виконавця, зниженні ризику невиконання роботи, отриманні якісного результату та можливості контролювати процес виконання замовлення. Для них важливими є зрозумілий інтерфейс створення заявки, шаблони опису технічного завдання, можливість прикріплення матеріалів, вибір категорії, вказання бюджету, строків і бажаного рівня досвіду виконавця. Виконавці, зі свого боку, зацікавлені в доступі до постійного потоку релевантних замовлень, справедливій конкуренції, можливості демонструвати власну експертизу та накопичувати репутацію. Для них важливими є зручний профіль, система навичок, портфоліо, фільтрація замовлень, сповіщення про нові проєкти та прозорі правила взаємодії.

За рівнем технічної підготовки аудиторія є неоднорідною. Частина замовників може мати глибоке розуміння Garry’s Mod-розробки, самостійно адмініструвати сервер і точно формулювати технічне завдання. Інша частина може мати лише загальне уявлення про бажаний результат, але не знати, які саме технології, скрипти або обмеження необхідно врахувати. Це створює потребу в інтерфейсі, який допомагає структурувати замовлення навіть для менш технічно

підготовлених користувачів. Наприклад, форма створення замовлення повинна містити категорії, підказки, поля для опису очікуваного результату, бюджету, термінів, наявних матеріалів і прикладів бажаного функціоналу. Водночас виконавці, як правило, мають вищий рівень технічної компетентності, тому для них важливішими є точність опису задачі, наявність технічних деталей, можливість поставити уточнювальні питання та оцінити складність роботи до прийняття замовлення.

Важливим фактором є те, що Garry’s Mod залишається активною грою з тривалою історією існування та стабільною користувацькою базою. За даними SteamDB, гра має десятки тисяч одночасних гравців у пікові періоди та понад мільйон користувацьких відгуків, що свідчить про значний масштаб і стійкість спільноти [11].

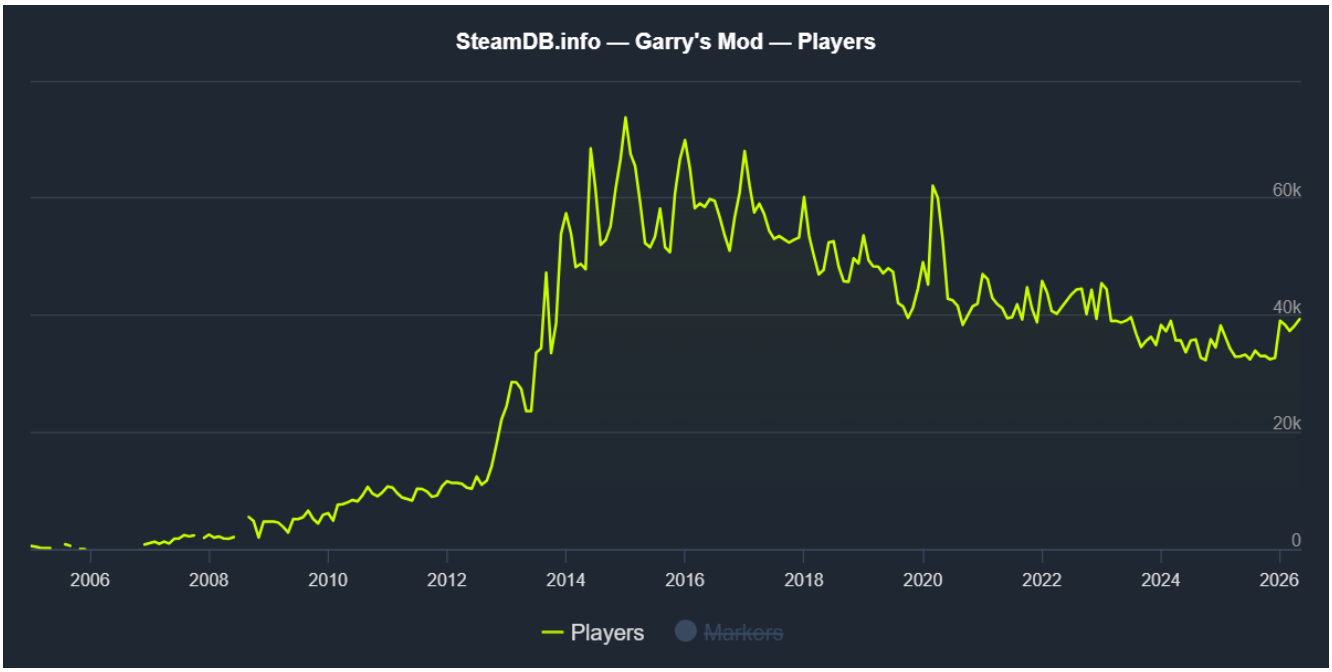


Рис 1.9 Графік одночасно підключених гравців в період 2005-2026 років [11]

Для досліджуваної системи це означає, що потенційна аудиторія не обмежується випадковими одноразовими користувачами, а формується навколо довготривалої екосистеми серверів, спільнот і розробників. Саме сталість цієї екосистеми створює передумови для появи спеціалізованого маркетплейсу

фриланс-послуг, який може обслуговувати повторювані потреби: запуск нових серверів, оновлення існуючих систем, розробку унікального контенту, технічну підтримку та адаптацію готових рішень.

З погляду користувацьких сценаріїв, замовник має отримати можливість швидко створити замовлення, описати проблему або бажаний результат, обрати категорію, зазначити бюджет і терміни, після чого отримати перелік потенційних виконавців або заявки від зацікавлених фрилансерів. Виконавець, у свою чергу, повинен мати можливість переглядати релевантні замовлення, фільтрувати їх за категоріями, подавати заявки, демонструвати портфоліо та отримувати відгуки після завершення роботи. Адміністратор системи повинен мати інструменти для контролю якості контенту, перегляду скарг, управління категоріями та підтримки правил платформи. Таке розділення сценаріїв дозволяє сформувати вимоги до ролей користувачів і визначити основні модулі системи.

Для узагальнення результатів дослідження цільової аудиторії доцільно сформувати її профіль за основними соціально-демографічними, поведінковими та функціональними характеристиками. Оскільки розроблювана система є двосторонньою платформою, її аудиторія не є однорідною: з одного боку виступають замовники, які мають потребу у створенні або доопрацюванні цифрового продукту, з іншого – виконавці, які надають професійні послуги у сфері Garry’s Mod-моддингу. Додатково в системі може бути передбачена роль адміністратора або модератора, однак основна цінність платформи формується саме через взаємодію між замовником і виконавцем.

Табл. 1.1

Узагальнений портрет цільової аудиторії платформи

Параметр	Замовники фриланс-послуг	Виконавці фриланс-послуг
Загальна характеристика	Власники або адміністратори Garry's Mod-серверів, керівники RP-проєктів, представники ігрових спільнот, які потребують створення або доопрацювання функціоналу для сервера.	GLua/Lua-розробники, UI/UX-дизайнери, 3D-моделери, мапери, веб-розробники, технічні спеціалісти, які виконують замовлення у сфері Garry's Mod.
Орієнтовний вік	Переважно 16-35 років. Найбільш активна група – користувачі 18–30 років, які мають досвід адміністрування серверів або участі в ігрових спільнотах.	Переважно 16-35 років. Частина виконавців починає з аматорської розробки для власних серверів і поступово переходить до комерційних замовлень.
Стать	Переважно чоловіки, що пов'язано зі специфікою аудиторії Garry's Mod та серверних ігрових спільнот, однак система не має обмежень за статтю.	Переважно чоловіки, особливо серед технічних спеціалістів, однак у напрямах дизайну, 3D-моделювання, ілюстрації та веб-розробки аудиторія може бути різноманітнішою.
Географія	Міжнародна аудиторія; користувачі з країн Європи, Північної Америки та СНД-простору. Взаємодія переважно відбувається дистанційно.	Міжнародна аудиторія; виконавці можуть працювати незалежно від місця проживання, якщо володіють необхідними технічними навичками та можуть комунікувати із замовником онлайн.
Рівень освіти	Середня, професійна або вища освіта. Технічна освіта не є обов'язковою, але частина замовників має базові знання адміністрування серверів, скриптів або конфігурації addon-рішень.	Середня, професійна або вища освіта. Для виконавців важливішими є практичні навички: програмування, робота з Lua/GLua, дизайн інтерфейсів, моделювання, мапінг або адміністрування серверів.
Рівень доходу / бюджет	Нерівномірний: від невеликих приватних серверів з обмеженим бюджетом до комерційних або великих RP-проєктів, готових оплачувати складні індивідуальні рішення.	Нерівномірний: частина виконавців сприймає замовлення як додатковий дохід, інші – як постійне або напівпостійне джерело заробітку.
Цілі використання платформи	Знайти надійного виконавця, швидко опублікувати замовлення, отримати якісний результат, зменшити ризик невиконання	Знайти релевантні замовлення, отримувати стабільний потік клієнтів, демонструвати портфоліо, накопичувати репутацію,

Параметр	Замовники фриланс-послуг	Виконавці фриланс-послуг
	роботи, контролювати процес розробки.	підвищувати вартість власних послуг.
Основні інтереси	Розвиток власного сервера, унікальний ігровий досвід, автоматизація серверних процесів, покращення інтерфейсів, залучення й утримання гравців.	Розробка модифікацій, програмування, дизайн, робота з ігровими системами, участь у спільноті, монетизація власних навичок.
Типові замовлення	HUD, VGUI-інтерфейси, DarkRP-системи, SWEP, ентіті, loading screen, донат-системи, інтеграції з Discord або веб-сервісами, мапи, моделі, оптимізація серверного коду.	Виконання технічних або творчих задач відповідно до спеціалізації: написання GLua-коду, створення інтерфейсів, моделей, карт, веб-частини, виправлення помилок, адаптація готових рішень.
Основні канали комунікації	Discord, Steam, форуми, приватні повідомлення, спільноти серверів, тематичні майданчики Garry’s Mod.	Discord, GitHub, Steam, портфоліо-сайти, форуми, GmodStore, особисті контакти у спільнотах.
Проблеми та болі	Складно знайти перевіреного виконавця; важко оцінити реальний рівень навичок; домовленості часто залишаються в особистих повідомленнях; відсутній зручний контроль статусу замовлення; існує ризик неякісної роботи або зриву термінів.	Замовлення розпорошені між різними Discord-серверами та спільнотами; складно підтвердити репутацію; новим виконавцям важко отримати перші замовлення; відсутня єдина система відгуків і портфоліо саме для Garry’s Mod.
Критерії вибору іншої сторони	Досвід у потрібній категорії, приклади попередніх робіт, відгуки, репутація, вартість, швидкість виконання, зрозуміла комунікація.	Зрозуміле технічне завдання, адекватний бюджет, реалістичні строки, надійність замовника, чіткість вимог, можливість отримати відгук після виконання.
Очікування від системи	Зручне створення замовлення, категорії робіт, фільтри, автоматизований підбір виконавців, перегляд профілів і портфоліо, відгуки, статуси виконання.	Зручний профіль, система навичок, портфоліо, пошук замовлень за категоріями, сповіщення про нові проєкти, можливість подавати заявки та накопичувати репутацію.
Рівень технічної підготовки	Від базового до середнього. Частина замовників може не знати точних технічних деталей, тому потребує підказок під час створення замовлення.	Від середнього до високого. Виконавці зазвичай мають практичні навички у конкретній сфері: GLua, VGUI, 3D-моделювання, мапінг, веб-розробка тощо.

Параметр	Замовники фриланс-послуг	Виконавці фриланс-послуг
Ключова цінність платформи	Зменшення ризиків під час пошуку виконавця та підвищення прозорості виконання замовлення.	Доступ до спеціалізованого ринку замовлень і можливість формувати професійну репутацію в межах Garry’s Mod-спільноти.

На основі сформованого профілю можна зробити висновок, що цільова аудиторія платформи має виражену предметну специфіку. Її потреби не обмежуються звичайним пошуком фрилансера або публікацією оголошення, оскільки для ефективної взаємодії необхідно враховувати категорії робіт, характерні саме для Garry’s Mod, рівень технічної підготовки сторін, значення репутації в межах спільноти та потребу в прозорому управлінні замовленнями. Саме тому розроблювана система повинна поєднувати функції фриланс-маркетплейсу, репутаційної платформи та інструменту управління індивідуальними проєктами.

1.4 Постановка задачі автоматизації підбору виконавців та управління проєктами і визначення функціональних та нефункціональних вимог до веб-платформи

Результати аналізу предметної області, існуючих платформ і цільової аудиторії показують, що середовище моддингу Garry’s Mod потребує спеціалізованого цифрового інструменту для організації взаємодії між замовниками та виконавцями. Основна проблема полягає в тому, що процес пошуку фахівців, погодження умов, виконання замовлень і формування довіри між сторонами часто відбувається фрагментовано: через Discord, форуми, особисті повідомлення або сторонні платформи, які не повністю враховують специфіку Garry’s Mod-спільноти. Унаслідок цього замовнику складно швидко знайти компетентного виконавця, оцінити його попередній досвід і рівень надійності, а виконавцю – отримати стабільний доступ до релевантних замовлень і накопичувати підтверджену репутацію в межах професійної ніші.

Задача автоматизації полягає у створенні веб-платформи, яка забезпечує централізоване середовище для створення, пошуку, виконання та оцінювання замовлень у сфері моддингу Garry’s Mod. Така система має поєднувати функції фриланс-біржі, каталогу профілів виконавців, репутаційного механізму та інструменту базового управління проєктним процесом. Її призначення полягає не лише в публікації замовлень, а й у зменшенні невизначеності під час вибору виконавця, підвищенні прозорості взаємодії, фіксації основних етапів виконання роботи та створенні умов для чесної конкуренції між спеціалістами.

Платформа повинна вирішувати три ключові групи задач. Перша група пов’язана з пошуком і підбором виконавців. Замовник має отримати можливість сформулювати потребу у вигляді структурованого замовлення, визначити категорію роботи, бюджет, строки виконання та опис очікуваного результату. Виконавець, зі свого боку, повинен мати змогу знаходити замовлення, що відповідають його навичкам і спеціалізації, подавати заявку та пропонувати власні умови виконання. Автоматизація цього процесу зменшує залежність від випадкових контактів і дозволяє будувати взаємодію на основі структурованих даних: категорії роботи, вартості, строків, історії виконаних замовлень, відгуків та репутації.

Друга група задач стосується управління життєвим циклом замовлення. У межах фриланс-взаємодії важливо не лише знайти виконавця, а й забезпечити зрозумілий процес переходу замовлення між станами: від публікації та збору заявок до вибору виконавця, виконання роботи, завершення або скасування. Для цього платформа має підтримувати статуси замовлення, які відображають його поточний стан і визначають доступні дії для учасників. Наприклад, відкрите замовлення повинно бути доступним для подання заявок, замовлення в роботі – закріпленим за обраним виконавцем, а завершене або скасоване – збереженим в історії взаємодії. Такий підхід дозволяє формалізувати процес виконання робіт і зменшити кількість спірних або невизначених ситуацій.

Третя група задач пов’язана з формуванням довіри та репутації. Для фриланс-платформи, особливо в нішевій спільноті, репутація є одним із головних механізмів

зменшення ризику. Замовник повинен бачити інформацію про виконавця, його попередній досвід, історію завершених робіт, відгуки та загальний рівень надійності. Виконавець, у свою чергу, також має мати можливість оцінювати досвід взаємодії із замовником, оскільки якість співпраці залежить від обох сторін. У наявній концепції платформи репутація розглядається як основна міра довіри до користувача, що формується через оцінки після виконання замовлень, а профіль користувача має містити історію опублікованих і виконаних робіт, опис, аватар, статуси прив'язаних облікових записів і загальний репутаційний показник.

Відповідно до поставленої задачі, першою функціональною вимогою є підтримка реєстрації та входу користувачів. Система повинна забезпечувати автентифікацію користувача та створення його профілю після входу. Оскільки цільова аудиторія пов'язана з ігровою та комунікаційною екосистемою Garry's Mod, важливою вимогою є можливість прив'язки зовнішніх облікових записів, зокрема Steam і Discord. Наявність таких прив'язок має відображатися у профілі користувача як додатковий маркер довіри, оскільки Steam пов'язаний із ігровою ідентичністю користувача, а Discord часто використовується як основний канал комунікації у спільноті. У матеріалах проєкту зазначено, що після входу для користувача створюється профіль, а статус прив'язки Steam та Discord має бути доступним іншим користувачам.

Другою функціональною вимогою є створення та публікація замовлень. Користувач у ролі замовника повинен мати можливість створити замовлення, вказавши його назву, опис, категорію, орієнтовний бюджет і бажаний термін виконання. Категоризація має відповідати основним напрямам робіт у Garry's Mod-спільноті, зокрема Lua-розробці, створенню моделей, текстур, карт, інтерфейсів та інших типів цифрового контенту. Структурованість замовлення є важливою умовою подальшого пошуку й підбору виконавців, оскільки саме ці параметри дозволяють фільтрувати завдання, зіставляти їх із компетенціями фрілансерів і зменшувати кількість неповних або не чітко сформульованих заявок.

Третьою функціональною вимогою є перегляд і пошук замовлень. Платформа повинна надавати список доступних замовлень із можливістю фільтрації за

категорією, статусом, бюджетом і порядком публікації. Це необхідно для того, щоб виконавці могли швидко знаходити релевантні задачі, а замовники – розуміти загальний стан ринку та конкуренцію за увагу спеціалістів. Повна сторінка замовлення повинна містити детальний опис роботи, бюджет, дедлайн, інформацію про замовника та інші дані, необхідні виконавцю для прийняття рішення щодо подання заявки. Така сторінка виконує роль основного інформаційного центру конкретного замовлення.

Четвертою функціональною вимогою є можливість подання заявок виконавцями. Виконавець повинен мати змогу відгукнутися на відкрите замовлення, вказавши орієнтовну вартість, строк виконання та, за потреби, короткий коментар. Це дозволяє замовнику порівнювати пропозиції не лише за ціною, а й за аргументацією, досвідом і очікуваними строками виконання. У свою чергу, виконавець отримує формалізований спосіб заявити про зацікавленість у роботі, не покладаючись на неструктуровані приватні повідомлення. Після отримання заявок замовник повинен мати можливість обрати одного виконавця, що переводить замовлення у стан виконання.

П’ятою функціональною вимогою є підтримка статусів замовлення. Система повинна розрізняти щонайменше такі стани: відкрите замовлення, замовлення в процесі виконання, завершене замовлення та скасоване замовлення. Статус визначає логіку доступних дій: на відкрите замовлення можна подавати заявки; після вибору виконавця нові заявки мають бути недоступними; завершене замовлення повинно потрапляти до історії користувачів і бути основою для оцінювання; скасоване замовлення має зберігати факт припинення взаємодії. Така модель є необхідною для базового управління проєктним процесом, оскільки дозволяє відстежувати прогрес і зменшує невизначеність між сторонами.

Шостою функціональною вимогою є наявність профілів користувачів. Профіль повинен виконувати функцію цифрової візитної картки замовника або виконавця. Він має містити базову інформацію про користувача, аватар, опис або біографію, статуси прив’язаних облікових записів, загальну репутацію, а також історію опублікованих і виконаних замовлень. Для виконавця профіль є

інструментом демонстрації досвіду, а для замовника – джерелом інформації для прийняття рішення про співпрацю. Для самої платформи профіль є центральним елементом репутаційної системи, оскільки саме навколо нього накопичуються дані про активність і якість взаємодії користувача.

Сьомою функціональною вимогою є реалізація системи рейтингів і відгуків. Після завершення замовлення користувачі повинні мати можливість залишити оцінку та, за потреби, текстовий коментар. Репутаційний показник має формуватися на основі отриманих оцінок і відображатися в профілі. Такий механізм дозволяє створити накопичувальну систему довіри, у якій кожне завершене замовлення впливає на подальшу видимість і привабливість користувача для потенційних партнерів. Водночас репутація не повинна розглядатися лише як числовий рейтинг; важливим є також контекст відгуків, історія виконаних робіт і відповідність досвіду конкретній категорії замовлень.

Окремо слід визначити вимоги до автоматизованого підбору виконавців. У межах базової версії платформи цей підбір може ґрунтуватися на структурованих параметрах замовлення та профілю користувача: категорії роботи, репутації, історії виконаних замовлень, наявності релевантного досвіду, активності виконавця та відповідності заявленим умовам. Основна мета такого механізму полягає не в повній заміні рішення замовника, а в інформаційній підтримці цього рішення. Система повинна допомагати замовнику швидше виявити потенційно придатних кандидатів, а виконавцям – знаходити замовлення, які відповідають їхній спеціалізації. Такий підхід підвищує ефективність взаємодії без надмірного ускладнення платформи на початковому етапі її розроблення.

Нефункціональні вимоги визначають якісні характеристики системи, які впливають на її зручність, надійність, безпеку та можливість подальшого розвитку. Насамперед платформа повинна бути доступною через веб-інтерфейс і забезпечувати коректну роботу для різних груп користувачів: замовників, виконавців і адміністраторів. Інтерфейс має бути зрозумілим, логічно структурованим і адаптованим до основних сценаріїв використання: створення замовлення, перегляд списку робіт, подання заявки, вибір виконавця, перегляд

профілю та оцінювання після завершення співпраці. Оскільки частина користувачів може не мати високої технічної підготовки, особливо серед замовників, система повинна допомагати їм формулювати замовлення через зрозумілі поля, категорії та підказки.

Важливою нефункціональною вимогою є надійність зберігання даних. Платформа повинна забезпечувати цілісність інформації про користувачів, замовлення, заявки, статуси, рейтинги та відгуки. Втрата або некоректна зміна таких даних може безпосередньо вплинути на довіру до сервісу, тому система повинна передбачати стабільну роботу з даними, коректну обробку помилок і збереження історії ключових дій. Особливо критичними є дані репутаційної системи, оскільки вони впливають на подальші можливості користувачів отримувати замовлення або обирати виконавців.

Наступною нефункціональною вимогою є безпека. Платформа повинна захищати облікові записи користувачів, персональні дані, історію взаємодії та репутаційні показники. Необхідним є розмежування доступу до дій залежно від ролі користувача та його відношення до конкретного замовлення. Наприклад, створювати замовлення може авторизований користувач, обирати виконавця – лише автор замовлення, подавати заявку – виконавець на відкрите замовлення, а залишати відгук – учасник завершеної взаємодії. У матеріалах проєкту також визначено потребу в захищених маршрутах, розмежуванні ролей і базових антиспам-механізмах, що є важливими умовами безпечної роботи платформи.

Система повинна бути масштабованою та придатною до подальшого розвитку. На початковому етапі достатньо реалізувати мінімально необхідний набір функцій: автентифікацію, створення й перегляд замовлень, подання заявок, вибір виконавця, завершення замовлення, профілі та рейтинги. Водночас архітектура платформи має дозволяти надалі розширювати функціональність: додавати розширений пошук, повідомлення, модерацію спорів, портфоліо, рекомендаційні механізми, категорії спеціалізацій, систему скарг, інтеграції з іншими сервісами або платіжні інструменти. Такий підхід дозволяє спочатку

перевірити життєздатність основної ідеї, а потім поступово розвивати платформу відповідно до реальних потреб користувачів.

До нефункціональних вимог також належить продуктивність. Платформа повинна забезпечувати швидке завантаження основних сторінок, оперативну фільтрацію замовлень і стабільну роботу профілів користувачів навіть за зростання кількості записів. Для фриланс-маркетплейсу особливо важливо, щоб користувач міг швидко знайти потрібну інформацію: актуальні замовлення, список заявок, репутацію виконавця або історію робіт. Низька швидкість роботи системи знижує зручність використання та може негативно впливати на активність користувачів.

Ще однією важливою вимогою є підтримка прозорості й зрозумілості логіки роботи платформи. Користувачі повинні розуміти, які дії вони можуть виконувати на кожному етапі, що означає певний статус замовлення, як формується рейтинг і чому певні дані відображаються в профілі. Це особливо важливо для репутаційної системи, оскільки непрозорі або незрозумілі правила можуть викликати недовіру й конфлікти між учасниками. Тому платформа повинна мати просту й передбачувану модель взаємодії, у якій ключові процеси легко пояснюються користувачам.

Таким чином, задача розроблюваної веб-платформи полягає у формуванні спеціалізованого середовища для автоматизації пошуку виконавців, організації замовлень, управління їхнім життєвим циклом і накопичення репутації в межах Garry’s Mod-спільноти. Функціональні вимоги охоплюють автентифікацію, профілі користувачів, створення й перегляд замовлень, фільтрацію, подання заявок, вибір виконавця, зміну статусів, завершення роботи та систему рейтингів. Нефункціональні вимоги передбачають зручність використання, безпеку, надійність, цілісність даних, масштабованість, продуктивність і прозорість логіки роботи. Виконання цих вимог дозволить створити платформу, яка не дублює універсальні фриланс-сервіси, а вирішує конкретну задачу нішевого ринку: забезпечує структуровану, безпечну та репутаційно прозору взаємодію між замовниками й виконавцями у сфері моддингу Garry’s Mod.

РОЗДІЛ 2. ВИБІР ТЕХНОЛОГІЧНОГО СТЕКУ ТА ПРОЄКТУВАННЯ АРХІТЕКТУРИ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ЗАМОВЛЕННЯМИ

2.1 Вибір та обґрунтування програмних засобів, СКБД та зовнішніх інструментів для реалізації системи

Для реалізації інформаційної системи управління замовленнями на маркетплейсі фриланс-послуг у середовищі моддингу Garry’s Mod необхідно обрати такі програмні засоби, які забезпечують швидку розробку веб-платформи, зручну роботу з користувацькими профілями, замовленнями, заявками, рейтингами та історією взаємодії. Оскільки система має обслуговувати дві основні групи користувачів – замовників і виконавців, – важливими критеріями вибору технологій є надійність, масштабованість, безпека, підтримка сучасної веб-архітектури, зручність розробки та можливість подальшого розширення функціоналу. У межах проєкту передбачено створення full-stack веб-платформи з використанням Next.js [12], tRPC [13], Prisma ORM [14], PostgreSQL [15], Better Auth [16], додаткового сервісу для інтеграції Steam-автентифікації [17] та Coolify [18] для розгортання. Такий набір технологій відповідає функціональним потребам системи, яка повинна підтримувати автентифікацію, створення та перегляд замовлень, подання заявок виконавцями, вибір виконавця, завершення замовлень, профілі користувачів і рейтингову систему.

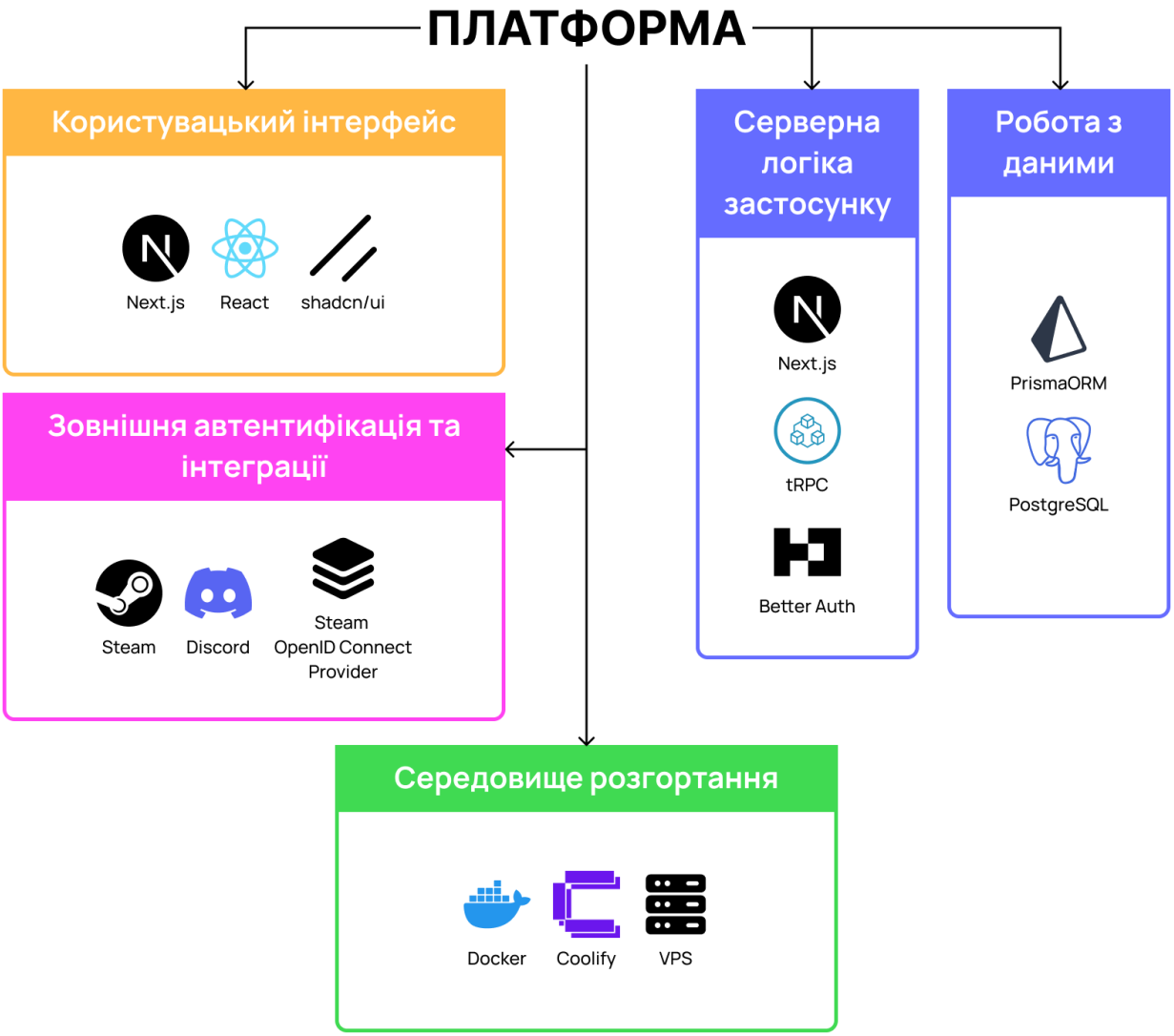


Рис 2.1 Технологічний стек платформи

Основним програмним засобом для побудови веб-платформи обрано Next.js з використанням App Router. Доцільність цього вибору пояснюється тим, що Next.js поєднує можливості frontend- і backend-розробки в межах одного застосунку, що є особливо зручним для створення MVP-версії платформи. У межах однієї кодової бази можна реалізувати сторінки користувацького інтерфейсу, серверну логіку, роботу з маршрутами, обробку захищених сторінок і взаємодію з API. App Router у Next.js орієнтований на побудову застосунків із використанням сучасної файлової маршрутизації, layout-структури, серверних та клієнтських компонентів, що дозволяє логічно організувати різні частини платформи, наприклад: сторінки

замовлень, профілі користувачів, форми створення заявок, списки доступних робіт і сторінки авторизації.

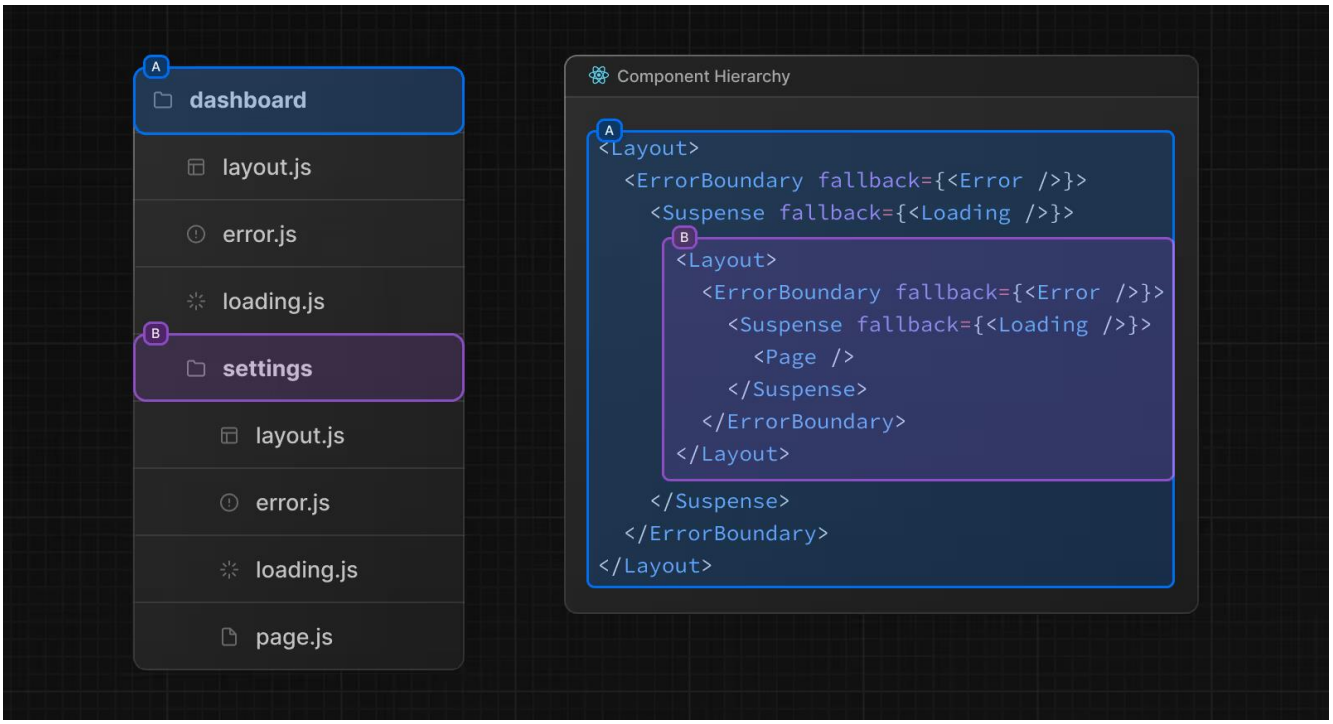


Рис 2.2 Приклад файлової маршутизації App Router в Next.js [19]

Офіційна документація Next.js описує App Router як підхід до побудови застосунків із підтримкою макетів, навігації, серверних та клієнтських компонентів, що робить його придатним для складних інтерактивних веб-сервісів з необхідністю реактивності [20].

Для досліджуваної системи Next.js є доречним також через те, що платформа потребує як публічних сторінок, так і захищених. Публічними можуть бути сторінки перегляду замовлень, профілі користувачів або загальна інформація про платформу. Захищеними мають бути дії, пов’язані зі створенням замовлення, поданням заявки, вибором виконавця, завершенням роботи та залишенням оцінки. Використання Next.js дозволяє побудувати таку структуру без необхідності розділяти frontend і backend на повністю окремі застосунки, що зменшує складність розробки та спрощує підтримку проєкту на початковому етапі.

Для реалізації взаємодії між клієнтською частиною та серверною логікою обрано tRPC. Цей інструмент доцільний у випадку full-stack застосунку на TypeScript, оскільки дозволяє створювати типобезпечний API без необхідності окремого опису схем у форматі REST або GraphQL. Офіційна документація tRPC визначає його як інструмент для побудови та використання повністю типобезпечних API без додаткової генерації коду або окремих схем [21]. Для платформи фриланс-замовлень це має практичне значення, оскільки більшість операцій працює зі даними з завчасно відомою структурою. Помилки у форматі таких даних можуть призводити до некоректної роботи бізнес-логіки, тому типобезпечна взаємодія між клієнтом і сервером підвищує надійність системи ще на етапі розробки.

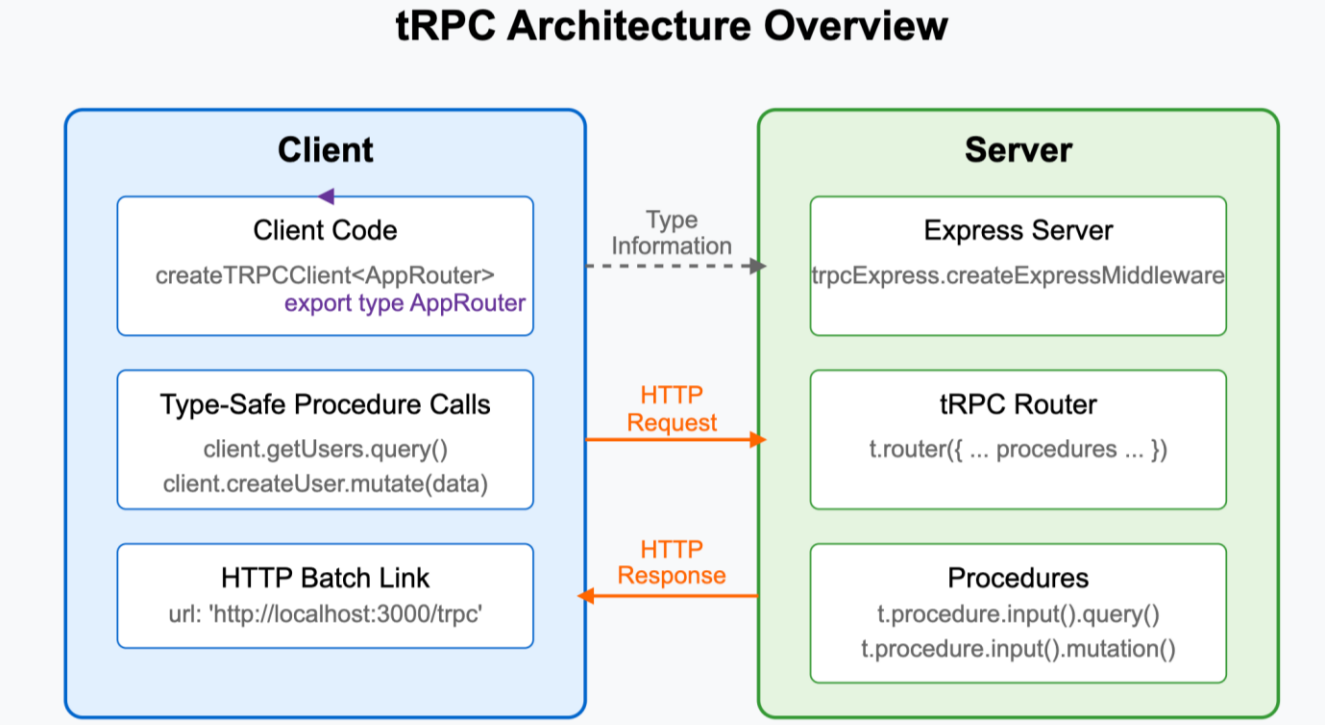


Рис 2.3 Загальна схема архітектури та роботи tRPC [22]

Використання tRPC також добре відповідає потребам MVP-реалізації. На початковому етапі система не потребує публічного API для сторонніх розробників, а основна взаємодія відбувається між власним frontend-інтерфейсом і серверною частиною платформи. У такому випадку tRPC дозволяє швидше реалізовувати

функції створення замовлень, фільтрації, подання заявок, вибору виконавця та оновлення статусів, оскільки типи даних можуть спільно використовуватися між різними частинами застосунку. Це зменшує дублювання коду та спрощує супровід системи під час подальшого розширення функціоналу.

Для роботи з базою даних обрано Prisma ORM. Його використання обґрунтоване потребою в надійному, зрозумілому та типобезпечному доступі до даних. Система управління замовленнями має зберігати взаємопов'язані сутності: користувачів, облікові записи, профілі, замовлення, заявки виконавців, статуси, рейтинги, відгуки та історію активності. Робота з такими даними потребує чіткої моделі, підтримки зв'язків і контрольованого внесення змін до структури бази. В офіційній документації Prisma ORM описується як інструмент для Node.js і TypeScript, що забезпечує типобезпечний доступ до бази даних, міграції та засоби перегляду даних [23].

Prisma є доцільним вибором для цього проєкту через поєднання кількох переваг. По-перше, вона дозволяє описати структуру даних у декларативному вигляді, що полегшує проєктування сутностей платформи. По-друге, Prisma Client забезпечує зручну роботу з даними на рівні TypeScript, що знижує ймовірність помилок під час звернення до полів моделей або побудови запитів. По-третє, система міграцій дозволяє поступово змінювати структуру бази даних у процесі розвитку платформи. Це особливо важливо, оскільки MVP може надалі розширюватися новими функціями: портфоліо, системою скарг, повідомленнями, розширеними фільтрами, модерацією або платіжними механізмами.

Як основну систему керування базами даних обрано PostgreSQL. Такий вибір є обґрунтованим, оскільки предметна область платформи має чітко виражену реляційну структуру. Користувач може мати профіль, зовнішні прив'язки, опубліковані замовлення, виконані замовлення, отримані або залишені відгуки. Замовлення, у свою чергу, пов'язане із замовником, категорією, заявками, обраним виконавцем, статусом і результатом оцінювання. Для такої моделі важливими є зв'язки між таблицями, обмеження цілісності, транзакційність і можливість виконання складних запитів. PostgreSQL підтримує обмеження цілісності даних,

які дозволяють контролювати коректність значень у таблицях і запобігати збереженню даних, що порушують визначені правила [24].

Для фриланс-платформи особливе значення має надійність операцій, пов’язаних зі зміною статусу замовлення, вибором виконавця та формуванням рейтингу. Наприклад, після вибору виконавця замовлення не повинно одночасно залишатися відкритим для нових заявок, а після завершення роботи оцінка має бути пов’язана з конкретним замовленням і конкретними користувачами. Такі операції потребують узгодженості даних. PostgreSQL як транзакційна реляційна СКБД добре підходить для таких сценаріїв, оскільки транзакції гарантують збереження результатів змін у постійному сховищі після успішного завершення операції [25].

Для реалізації аутентифікації обрано Better Auth. Вибір цього інструменту пояснюється необхідністю підтримки сучасної системи входу користувачів, інтеграції з зовнішніми провайдерами, роботи із сесіями та можливістю подальшого розширення механізмів авторизації. Офіційна документація Better Auth описує його як framework-agnostic authentication and authorization framework для TypeScript, який надає готовий набір функцій і плагіну екосистему [26]. Для розроблюваної платформи це важливо, оскільки аутентифікація не є другорядною функцією: вона безпосередньо пов’язана з довірою, репутацією, профілями користувачів і можливістю прив’язки Steam та Discord.

У контексті Garry’s Mod-спільноти особливо важливо підтримати зовнішні облікові записи, пов’язані з реальною активністю користувача. Steam виступає ключовою платформою для ігрової ідентичності, тоді як Discord є одним із головних каналів комунікації в ігрових спільнотах. Тому прив’язка цих облікових записів дозволяє підвищити довіру до профілю користувача та зменшити ризик анонімних або недобросовісних взаємодій. У вимогах до платформи передбачено авторизацію через OAuth/OIDC, можливість прив’язки Steam і Discord, а також відображення статусу таких прив’язок у профілі користувача. На рисунку 2.4 зображено послідовність автентифікації користувача за протоколом OAuth/OIDC.

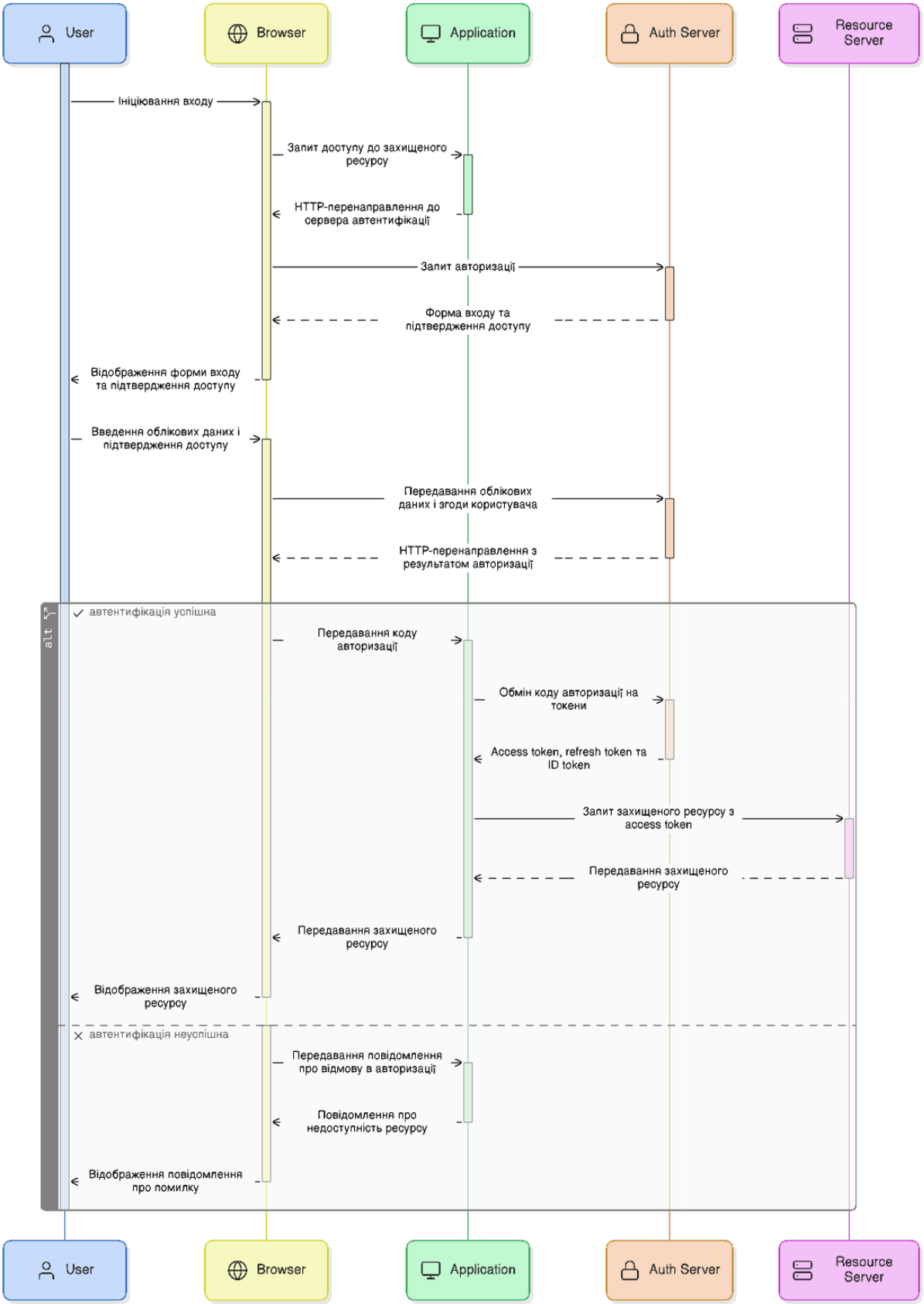


Рис 2.4 Послідовність взаємодії користувача, веб застосунку, сервера аутентифікації та сервера ресурсів під час OAuth/OIDC-аутентифікації

Користувач ініціює вхід через браузер, після чого браузер звертається до веб застосунку із запитом доступу до захищеного ресурсу. Веб Застосунок перенаправляє браузер на сервер аутентифікації, де користувач вводить облікові дані та підтверджує доступ. У разі успішної аутентифікації сервер аутентифікації повертає код авторизації через браузер до веб застосунку. Далі веб застосунок обмінює цей код на access token, refresh token та ID token, після чого використовує access token для отримання захищеного ресурсу із сервера ресурсів. Якщо автентифікація є успішною, захищений ресурс передається до браузера та відображається користувачу. У разі неуспішної аутентифікації веб застосунок отримує повідомлення про відмову в авторизації та відображає користувачу повідомлення про помилку.

Для інтеграції Steam-аутентифікації передбачено використання додаткового сервісу steam-openid-connect-provider. Необхідність такого інструменту пояснюється тим, що Steam традиційно використовує OpenID 2.0, тоді як сучасні системи автентифікації частіше орієнтуються на OAuth/OIDC-підхід. Репозиторій byo-software/steam-openid-connect-provider описує цей сервер як проксі, що дозволяє використовувати Steam як OpenID Connect Identity Provider [17]. На рисунку 2.5 зображено процес автентифікації користувача з використанням проміжного OpenID Proxy.

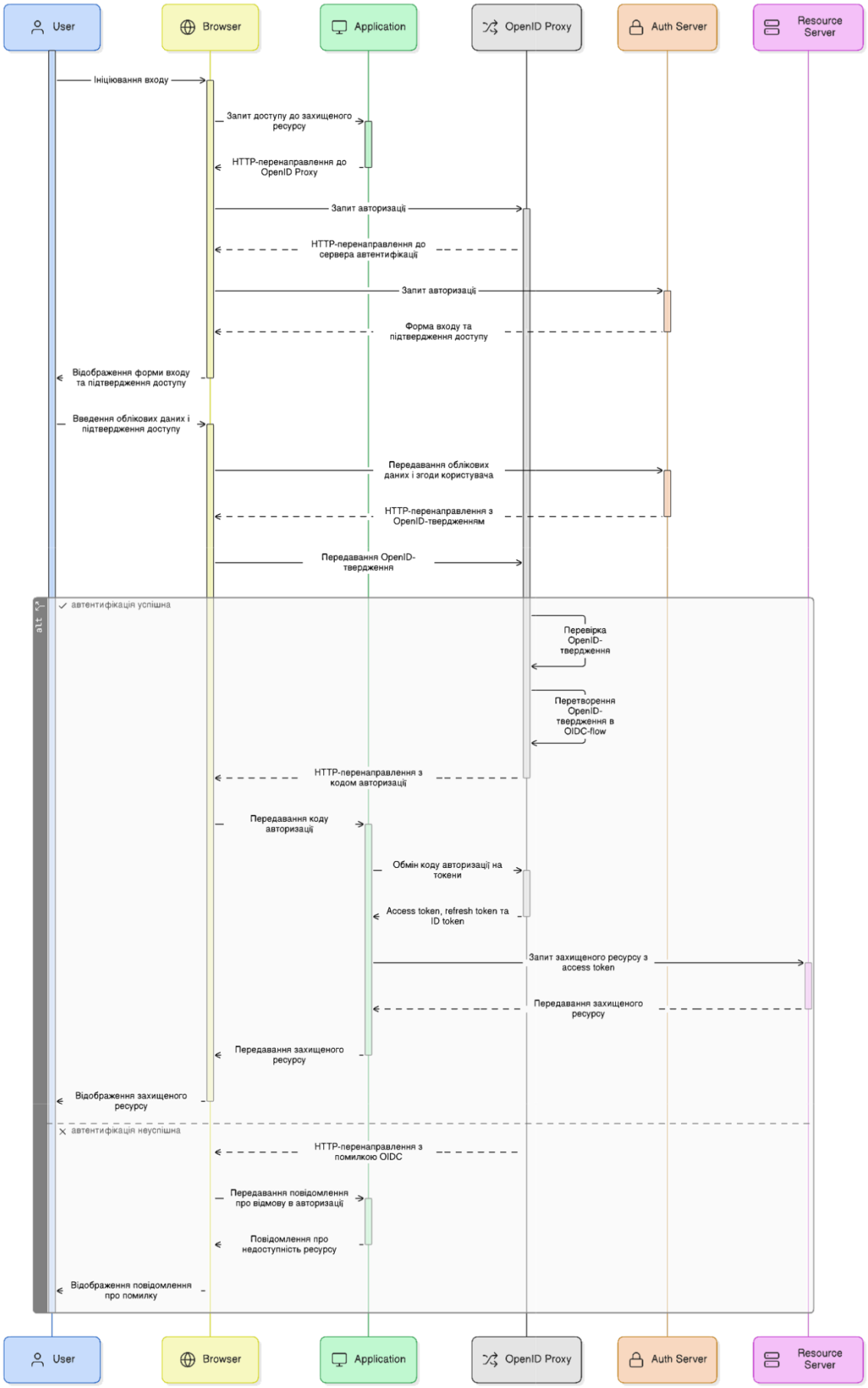


Рис 2.5 Послідовність OAuth/OIDC-аутентифікації з використанням OpenID Proxy

Такий підхід застосовується у випадку, коли зовнішній сервер аутентифікації не підтримує повноцінний OAuth/OIDC-flow напряму, але може працювати через OpenID-механізм. Користувач ініціює вхід через браузер, після чого веб застосунок перенаправляє запит не безпосередньо до сервера аутентифікації, а до OpenID Proxy. Проксі виконує взаємодію із сервером автентифікації, отримує OpenID-твердження, перевіряє його та перетворює результат у OIDC-сумісний потік. У разі успішної аутентифікації OpenID Proxy повертає браузеру код авторизації, який передається до веб застосунку. Далі веб застосунок обмінює цей код на access token, refresh token та ID token через OpenID Proxy і використовує access token для запиту захищеного ресурсу із сервера ресурсів. Якщо автентифікація завершується помилкою, OpenID Proxy повертає OIDC-помилку, після чого веб застосунок повідомляє користувача про недоступність ресурсу. Таким чином, OpenID Proxy виконує роль проміжного адаптера між несумісним OpenID-механізмом і OAuth/OIDC-архітектурою вебплатформи. Для платформи це дає змогу узгодити Steam-вхід із загальною моделлю автентифікації, не ускладнюючи основну логіку застосунку та зберігаючи можливість працювати з Steam-ідентичністю користувача як із частиною профілю.

Discord-інтеграція також є важливою для обраної предметної області, оскільки більшість Garry’s Mod-спільнот використовує Discord для організації команд, спілкування, публікації новин, підтримки користувачів і пошуку виконавців. Прив’язка Discord-акаунта до профілю на платформі дозволяє користувачам легше підтверджувати свою присутність у спільноті та забезпечує додатковий канал ідентифікації. Водночас така інтеграція повинна використовуватися саме як елемент довіри та профілю, а не як повна заміна внутрішніх механізмів платформи. Основна взаємодія щодо замовлень, заявок, статусів і рейтингів має зберігатися всередині системи, щоб уникнути фрагментації даних.

Для розгортання платформи обрано Coolify. Цей інструмент є доцільним для MVP і подальшої експлуатації, оскільки дозволяє розгортати застосунки, бази даних і сервіси на власному сервері без необхідності вручну налаштовувати всю

інфраструктуру з нуля. Офіційна документація Coolify визначає його як open-source PaaS для self-hosting баз даних, сервісів і застосунків із підтримкою Git-інтеграції, SSL, резервного копіювання та розгортання веб-застосунків [27]. У межах проєкту це відповідає вимозі розгортання базової версії на VPS із CI/CD-підходом.

Вибір Coolify також обґрунтований з позиції контролю над інфраструктурою. Для невеликої спеціалізованої платформи використання власного VPS може бути економічно доцільнішим, ніж повністю керовані хмарні сервіси. Крім того, self-hosted підхід дозволяє гнучко керувати середовищем, базою даних, додатковими сервісами й оновленнями. Coolify підтримує сценарії безперервного розгортання з Git-репозиторіїв, коли застосунок може автоматично оновлюватися після внесення змін у кодову базу [28].

Загалом обраний стек технологій є узгодженим і логічним для поставленої задачі. Next.js забезпечує основу для побудови сучасного веб-інтерфейсу та серверної логіки, tRPC спрощує типобезпечну взаємодію між клієнтом і сервером, Prisma ORM забезпечує зручну роботу з реляційною моделлю даних, PostgreSQL відповідає вимогам до надійного зберігання пов’язаних сутностей, Better Auth реалізує автентифікацію й авторизацію, steam-openid-connect-provider дозволяє інтегрувати Steam-ідентичність у сучасну модель входу, а Coolify забезпечує практичне розгортання та підтримку платформи на власній інфраструктурі. Такий набір засобів дозволяє реалізувати мінімально життєздатну версію платформи, що включає автентифікацію, створення й перегляд замовлень, заявки виконавців, вибір виконавця, завершення замовлення, профілі та рейтинги, а також залишає можливість для подальшого розвитку системи.

2.2 Проєктування архітектури веб-платформи

Проєктування архітектури веб-платформи є одним із ключових етапів розроблення інформаційної системи, оскільки саме архітектура визначає загальну структуру застосунку, взаємодію між його компонентами, принципи обробки даних, розмежування відповідальності між модулями та можливість подальшого

масштабування. Для платформи управління замовленнями у сфері моддингу Garry’s Mod архітектура повинна забезпечувати стабільну роботу основних бізнес-процесів: аутентифікацію користувачів, створення та перегляд замовлень, подання заявок виконавцями, вибір виконавця, зміну статусів замовлення, формування профілів і накопичення репутації. У межах базової версії системи передбачено реалізацію саме цих функціональних модулів, що дозволяє побудувати мінімально життєздатний продукт і надалі розширювати його відповідно до потреб користувачів.

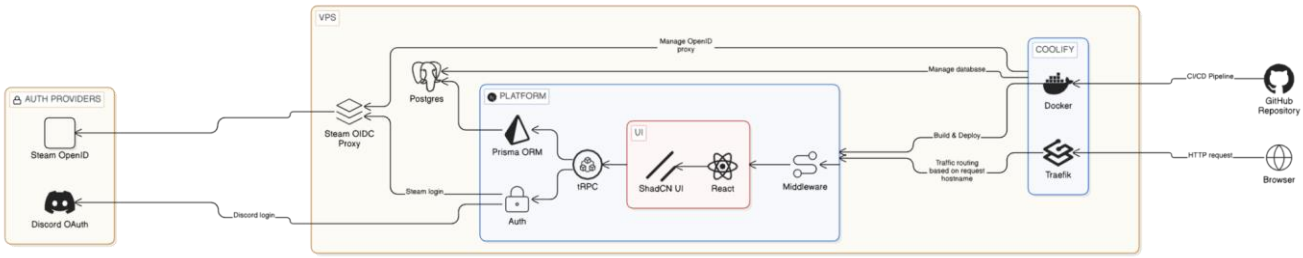


Рис 2.6 Архітектура веб-платформи

Архітектура веб-платформи проєктується як full-stack система, у якій клієнтська частина, серверна логіка, API-рівень і доступ до бази даних поєднані в межах єдиного веб-застосунку. Такий підхід є доцільним для MVP, оскільки зменшує складність розроблення, спрощує узгодження між інтерфейсом і бізнес-логікою та дозволяє швидко реалізувати основні сценарії використання. Водночас система не повинна бути монолітною в логічному сенсі: навіть за умови розміщення в одній кодовій базі її функціональні частини мають бути розділені на окремі модулі, кожен із яких відповідає за певну групу задач. Це дозволяє підтримувати зрозумілу структуру застосунку й полегшує подальше розширення.

Загальну архітектуру платформи можна представити як багаторівневу структуру, що включає рівень користувацького інтерфейсу, рівень серверної логіки, рівень доступу до даних, базу даних і зовнішні інтеграції. Розділення архітектури на рівні представлено в вигляді таблиці 2.1.

Табл. 2.1

Загальні рівні архітектури платформи

Рівень архітектури	Призначення в систем
Користувацький інтерфейс	Відображення сторінок замовлень, профілів, форм створення замовлень і заявок
Серверна логіка	Обробка дій користувачів, перевірка доступу, виконання бізнес-правил
API-рівень	Типобезпечна взаємодія між клієнтською частиною та сервером
Рівень доступу до даних	Робота з моделями бази даних, запитами та зв’язками між сутностями
База даних	Збереження користувачів, замовлень, заявок, статусів, рейтингів і відгуків
Зовнішні сервіси	аутентифікація, прив’язка зовнішніх облікових записів, розгортання та підтримка застосунку

Основним компонентом архітектури є веб-застосунок, побудований на Next.js. Він виконує роль центральної частини системи, через яку користувачі отримують доступ до всіх функцій платформи. У межах цього застосунку реалізуються як публічні сторінки, так і захищені розділи. До публічних сторінок можуть належати головна сторінка платформи, список відкритих замовлень, сторінки перегляду окремих замовлень і публічні профілі користувачів. Захищені сторінки доступні лише авторизованим користувачам і використовуються для створення замовлень, подання заявок, вибору виконавця, завершення роботи та залишення оцінок. Такий поділ дозволяє поєднати відкритість платформи для перегляду з контрольованим доступом до дій, які змінюють дані системи.

Користувацький інтерфейс платформи повинен бути спроєктований навколо основних ролей користувачів. Замовник взаємодіє із системою через створення замовлення, перегляд заявок, вибір виконавця та завершення роботи. Виконавець

використовує платформу для пошуку замовлень, фільтрації за категоріями, подання заявок і накопичення репутації. Адміністратор або модератор може бути залучений до контролю якості контенту, реагування на скарги та підтримки коректної роботи платформи. Такий рольовий підхід важливий для архітектури, оскільки різні групи користувачів мають різні права доступу та різні сценарії використання.

Серверна частина платформи відповідає за виконання бізнес-логіки. Саме на цьому рівні перевіряється, чи може користувач виконати певну дію. Наприклад, лише авторизований користувач може створити замовлення; лише виконавець може подати заявку на відкрите замовлення; лише автор замовлення може обрати виконавця; лише учасники завершеної взаємодії можуть залишити оцінку. Така перевірка не повинна покладатися лише на інтерфейс, оскільки користувацька частина може бути обійдена. Тому контроль доступу має бути реалізований на серверному рівні, де обробляються всі значущі зміни стану системи.

API-рівень доцільно організувати через набір процедур, які відповідають основним діям користувачів. До таких процедур належать створення замовлення, отримання списку замовлень, перегляд конкретного замовлення, подання заявки, вибір виконавця, зміна статусу замовлення, отримання профілю користувача, оновлення профілю та створення відгуку. Використання типобезпечного API дозволяє зменшити кількість помилок під час передавання даних між інтерфейсом і серверною логікою. Для системи, де багато операцій пов’язані зі статусами, ролями й пов’язаними сутностями, це має важливе значення, оскільки некоректні дані можуть порушити логіку виконання замовлення.

Рівень доступу до даних забезпечується через ORM, яка виступає посередником між серверною логікою та реляційною базою даних. Такий підхід дозволяє працювати з даними на рівні моделей, а не вручну формувати всі запити до бази. У межах платформи це особливо корисно, оскільки більшість сутностей має зв’язки між собою. Користувач пов’язаний із профілем, замовленнями, заявками й оцінками. Замовлення пов’язане із замовником, категорією, бюджетом, терміном виконання, статусом, заявками виконавців і, після вибору, конкретним виконавцем. Відгук пов’язаний із завершеним замовленням і користувачами, які

брали участь у взаємодії. Така структура потребує цілісної моделі даних і контрольованої роботи зі зв’язками.

База даних виступає центральним сховищем інформації. Для цієї системи доцільною є реляційна модель, оскільки предметна область має чітко визначені сутності та зв’язки між ними. Основними сутностями є користувач, профіль користувача, замовлення, заявка на замовлення, рейтинг, відгук, зовнішній обліковий запис і статус замовлення. Кожна із цих сутностей виконує окрему роль у бізнес-логіці платформи. Користувач є базовою одиницею системи. Профіль містить публічну інформацію та репутаційні показники. Замовлення описує потребу замовника. Заявка фіксує пропозицію виконавця. Статус відображає поточний етап життєвого циклу замовлення. Рейтинг і відгук забезпечують формування довіри між сторонами.

Життєвий цикл замовлення є одним із центральних процесів архітектури системи. Після створення замовлення воно отримує статус відкритого й стає доступним для перегляду та подання заявок. Виконавці можуть залишати заявки з орієнтовною ціною, строком виконання та коментарем. Замовник переглядає отримані заявки та обирає виконавця, після чого замовлення переходить у стан виконання. На цьому етапі подання нових заявок має бути недоступним, оскільки робота вже закріплена за конкретним виконавцем. Після завершення роботи замовлення переводиться в завершений стан, а учасники отримують можливість залишити оцінки. Якщо робота з певних причин не відбулася, замовлення може бути скасоване. Саме така модель статусів дозволяє системі формалізувати проєктну взаємодію й уникати невизначеності.

Архітектура повинна також враховувати автоматизацію підбору виконавців. На базовому рівні вона може реалізовуватися через структуровані дані, які вже присутні в системі: категорії замовлень, історію виконаних робіт, рейтинг користувача, кількість позитивних і негативних відгуків, наявність релевантного досвіду та активність виконавця. У межах архітектури це означає, що дані мають зберігатися у такому вигляді, щоб їх можна було використовувати не лише для відображення, а й для фільтрації, сортування та рекомендацій. Тобто система

повинна бути спроєктована не як проста дошка оголошень, а як платформа, у якій накопичені дані поступово підвищують якість підбору виконавців.

Окреме місце в архітектурі займає модуль аутентифікації та ідентифікації користувачів. Оскільки платформа працює з репутацією, історією замовлень і взаємодією між сторонами, користувач повинен мати стабільний обліковий запис. Система має підтримувати вхід користувача, створення профілю після першого входу та можливість прив’язки зовнішніх облікових записів Steam і Discord. У вимогах до платформи передбачено, що статус таких прив’язок має бути доступним іншим користувачам, оскільки він виконує роль додаткового індикатора довіри. В архітектурному плані це означає, що інформація про зовнішні акаунти повинна бути пов’язана з профілем користувача й використовуватися при відображенні публічної інформації.

Модуль профілів користувачів повинен бути тісно пов’язаний із модулями замовлень, заявок і рейтингів. Профіль не є ізольованою сторінкою з базовою інформацією; він виконує роль агрегованого представлення активності користувача на платформі. Для замовника профіль виконавця дозволяє оцінити досвід, репутацію та історію виконаних робіт. Для виконавця профіль замовника дозволяє зрозуміти, чи має він позитивну історію взаємодії з іншими фрилансерами. Таким чином, архітектура повинна забезпечувати можливість відображення в профілі не лише статичних даних, а й динамічно сформованої інформації: кількості завершених замовлень, отриманих оцінок, історії робіт і загального репутаційного показника.

Модуль рейтингу та відгуків повинен бути пов’язаний із фактом завершення замовлення. Це важливо для запобігання зловживанням, оскільки оцінка має бути результатом реальної взаємодії, а не довільною дією будь-якого користувача. Тому архітектура повинна передбачати, що залишити відгук можна лише після завершення замовлення та лише користувачам, які брали участь у ньому. Такий підхід підвищує достовірність репутаційної системи й робить рейтинг більш корисним для прийняття рішень. У базовій моделі рейтинг може формуватися через позитивні та негативні оцінки, доповнені необов’язковим текстовим коментарем.

Для кращого розуміння взаємодії компонентів архітектуру системи можна описати через основні модулі наведені в таблиці 2.2.

Табл. 2.2

Призначення модулей системи

Модуль системи	Основне призначення
Модуль аутентифікації	Вхід користувачів, створення облікового запису, прив’язка зовнішніх акаунтів
Модуль профілів	Відображення інформації про користувача, історії замовлень і репутації
Модуль замовлень	Створення, перегляд, фільтрація та зміна статусів замовлень
Модуль заявок	Подання пропозицій виконавцями та вибір виконавця замовником
Модуль рейтингів	Формування оцінок і відгуків після завершення замовлення
Модуль доступу	Перевірка прав користувачів на виконання дій
Модуль адміністрування	Контроль порушень, підтримка правил і базова модерація платформи

З погляду безпеки архітектура повинна передбачати захист як на рівні сторінок, так і на рівні серверних процедур. Захищені сторінки не повинні бути доступні неавторизованим користувачам, а серверні операції повинні додатково перевіряти права доступу. Це особливо важливо для дій, які впливають на стан замовлення або репутацію користувача. Наприклад, користувач не повинен мати можливості завершити чуже замовлення, змінити чужу заявку, залишити відгук без факту співпраці або штучно вплинути на рейтинг. У матеріалах проєкту окремо зазначено потребу в захищених маршрутах, розмежуванні ролей і антиспам-функціональності, що має бути враховано в архітектурі системи.

Архітектура також повинна враховувати можливість подальшого масштабування. На початковому етапі платформа може працювати як єдиний full-

stack застосунок із підключеною базою даних і допоміжними сервісами. Проте логічне розділення модулів дозволяє в майбутньому розширювати систему без повного перепроєктування. Наприклад, модуль повідомлень, система скарг, розширений пошук, платіжний модуль, портфоліо або рекомендаційний алгоритм можуть бути додані як окремі функціональні частини. Також у майбутньому окремі інфраструктурні компоненти можуть бути винесені в самостійні сервіси, якщо навантаження або вимоги до системи зростатимуть.

Розгортання платформи доцільно організувати через VPS із використанням середовища, яке підтримує автоматизоване оновлення застосунку. Такий підхід відповідає потребам MVP, оскільки дозволяє швидко розгорнути систему, тестувати її у реальних умовах і поступово вносити зміни. У межах архітектури розгортання можна виділити веб-застосунок, базу даних, сервіс автентифікації для Steam-інтеграції та механізм автоматичного оновлення після змін у репозиторії. Для початкової версії цього достатньо, оскільки головною метою є перевірка працездатності основних бізнес-процесів: створення замовлення, подання заявки, вибір виконавця, завершення роботи та формування рейтингу.

Важливою вимогою до архітектури є підтримка цілісності даних. Зміна статусу замовлення, вибір виконавця та створення відгуку повинні відбуватися так, щоб система не переходила в суперечливий стан. Наприклад, замовлення не може одночасно бути відкритим і виконуваним; завершене замовлення не повинно приймати нові заявки; відгук не повинен існувати без зв’язку з конкретним замовленням. Такі правила мають бути закладені не лише в інтерфейсі, а й у серверній логіці та структурі даних. Це підвищує надійність системи та знижує ризик помилок у процесі експлуатації.

Таким чином, архітектура веб-платформи повинна забезпечувати поєднання простоти MVP-реалізації з можливістю подальшого розвитку. Її основу становить full-stack веб-застосунок, який взаємодіє з реляційною базою даних, модулем автентифікації, зовнішніми ідентифікаційними сервісами та інфраструктурою розгортання. Логічно система поділяється на модулі автентифікації, профілів, замовлень, заявок, рейтингів, контролю доступу та адміністрування. Така

архітектура дозволяє реалізувати основні функції платформи, забезпечити контроль життєвого циклу замовлення, створити основу для автоматизованого підбору виконавців і сформувати репутаційне середовище, необхідне для безпечної та прозорої взаємодії в межах Garry’s Mod-спільноти.

2.3 Моделювання функціоналу системи

Моделювання функціоналу системи є необхідним етапом проєктування веб-платформи, оскільки дозволяє формалізувати основні сценарії взаємодії користувачів із системою, визначити межі відповідальності окремих модулів і описати логіку переходу даних між станами. Для інформаційної системи управління замовленнями на маркетплейсі фриланс-послуг функціональна модель повинна відображати не лише окремі дії користувачів, а й повний цикл роботи із замовленням: від його створення до вибору виконавця, завершення роботи та формування репутаційної оцінки. Такий підхід дозволяє розглядати платформу не як просту дошку оголошень, а як систему підтримки керованої взаємодії між замовником і виконавцем.

У межах системи доцільно виділити три основні ролі користувачів: замовник, виконавець і адміністратор. Замовник створює замовлення, переглядає заявки, обирає виконавця, контролює стан виконання та залишає відгук після завершення роботи. Виконавець переглядає доступні замовлення, фільтрує їх за параметрами, подає заявку, бере участь у виконанні роботи та також формує відгук після завершення взаємодії. Адміністратор забезпечує підтримку стабільної роботи платформи, контроль порушень, модерацію контенту та реагування на потенційні зловживання. Основна бізнес-логіка платформи зосереджується навколо взаємодії перших двох ролей, тоді як адміністративна роль виконує допоміжну функцію підтримки довіри й безпеки.

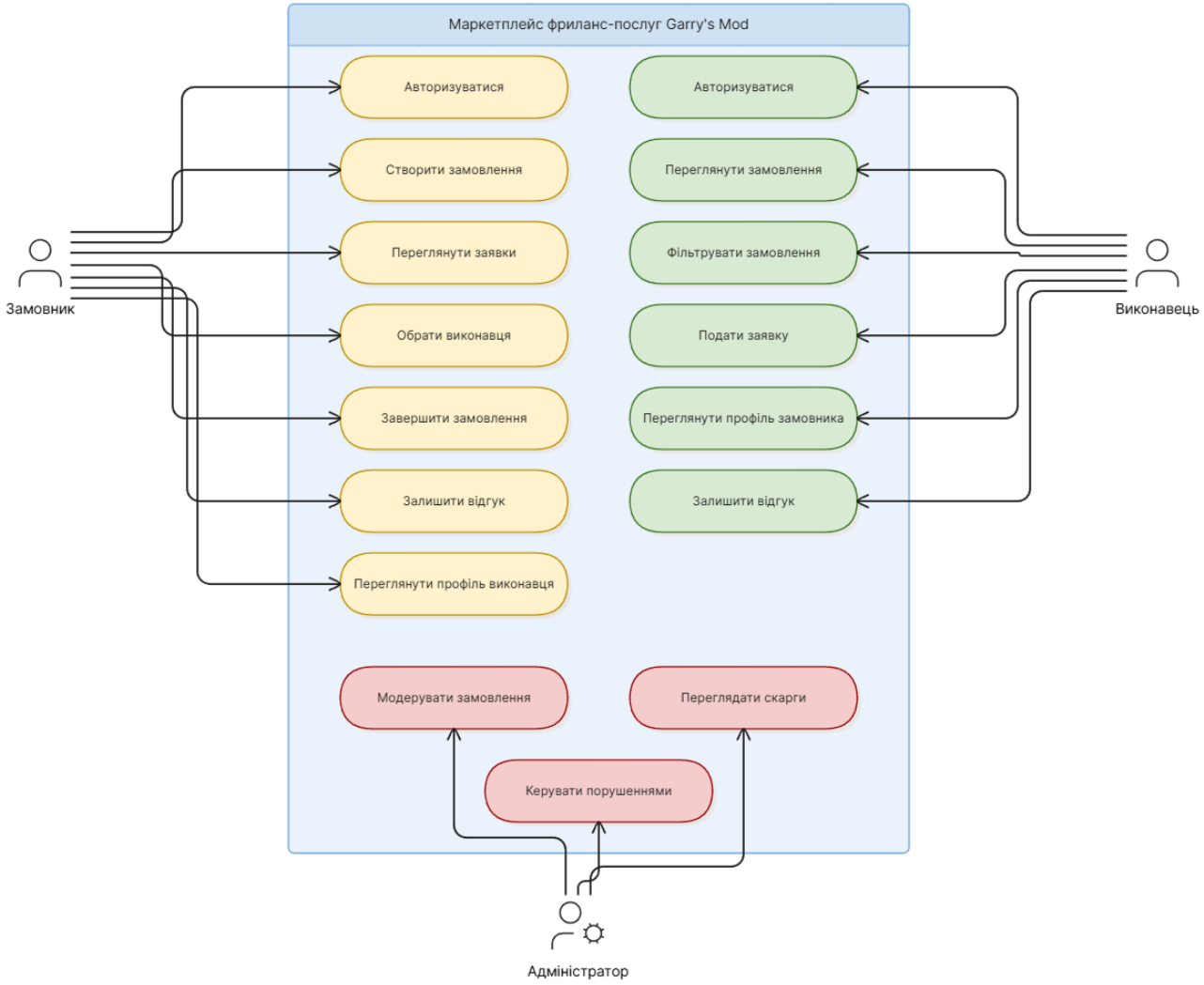


Рис 2.7 UML-діаграма варіантів використання системи

Першим функціональним процесом системи є аутентифікація користувача та створення профілю. Оскільки платформа передбачає взаємодію між реальними учасниками спільноти, користувач повинен мати обліковий запис, пов’язаний із його ідентичністю в зовнішніх сервісах. У функціональній моделі цей процес починається з входу користувача через підтримуваний механізм аутентифікації. Після успішного входу система перевіряє, чи існує профіль користувача. Якщо профілю ще немає, він створюється автоматично. Надалі користувач може доповнити профіль описом, аватаром, прив’язками Steam або Discord, а також використовувати його для участі в замовленнях. У вимогах до системи передбачено

авторизацію через OAuth/OIDC, можливість прив’язки Steam і Discord, а також створення профілю після входу користувача.

Другим ключовим процесом є створення замовлення. Замовник формує замовлення шляхом заповнення структурованої форми, у якій зазначає заголовок, опис, категорію, орієнтовний бюджет і термін виконання. Ці дані є основою для подальшого пошуку виконавців і фільтрації замовлень. Категорія дозволяє віднести роботу до певного напряму, наприклад Lua-розробки, моделей, текстур, карт, UI або іншого виду робіт. Бюджет і термін виконання допомагають виконавцю оцінити доцільність участі, а опис визначає зміст очікуваного результату. Після створення замовлення система присвоює йому початковий статус “відкрите”, що означає можливість перегляду й подання заявок іншими користувачами. Структура замовлення та вимоги до його відображення передбачають наявність заголовка, опису, категорії, бюджету, терміну виконання й фільтрації за основними параметрами.

Наступним процесом є пошук і перегляд замовлень виконавцями. Виконавець отримує доступ до списку відкритих замовлень і може використовувати фільтри за категорією, бюджетом, статусом або порядком публікації. Така модель дозволяє уникнути ситуації, коли всі замовлення відображаються як не структурований потік повідомлень. З погляду функціонального моделювання, цей процес включає отримання списку замовлень, застосування фільтрів, перегляд короткої інформації та перехід на повну сторінку замовлення. Повна сторінка повинна надавати виконавцю достатньо інформації для прийняття рішення щодо подання заявки: опис, бюджет, дедлайн, категорію, інформацію про замовника та його репутаційну історію.

Подання заявки виконавцем є окремим функціональним процесом, який пов’язує конкретного виконавця з конкретним замовленням. Виконавець може залишити заявку лише на замовлення зі статусом “відкрите”. У заявці він зазначає орієнтовну вартість, очікуваний термін виконання та короткий коментар. Після створення заявки замовник бачить її в переліку пропозицій до свого замовлення. На цьому етапі система виконує роль посередника, який структурує пропозиції

виконавців і дозволяє замовнику порівнювати їх за кількома параметрами. У наявній моделі передбачено, що виконавець може залишити заявку з приблизною ціною, терміном виконання й коментарем, а замовник отримує можливість переглянути всі заявки та обрати виконавця.

Після отримання заявок замовник переходить до процесу вибору виконавця. Функціонально цей процес складається з перегляду списку заявок, аналізу профілів виконавців, порівняння репутації, історії виконаних замовлень, запропонованої ціни та строків. Після вибору конкретного виконавця система змінює статус замовлення з “відкрите” на “в процесі виконання”. З цього моменту подання нових заявок повинно бути заблоковане, оскільки замовлення вже закріплене за конкретним виконавцем. Такий механізм забезпечує цілісність життєвого циклу замовлення та запобігає одночасному вибору кількох виконавців для одного й того самого проєкту.

Центральним елементом функціональної моделі є життєвий цикл замовлення зображений на рисунку 2.8.

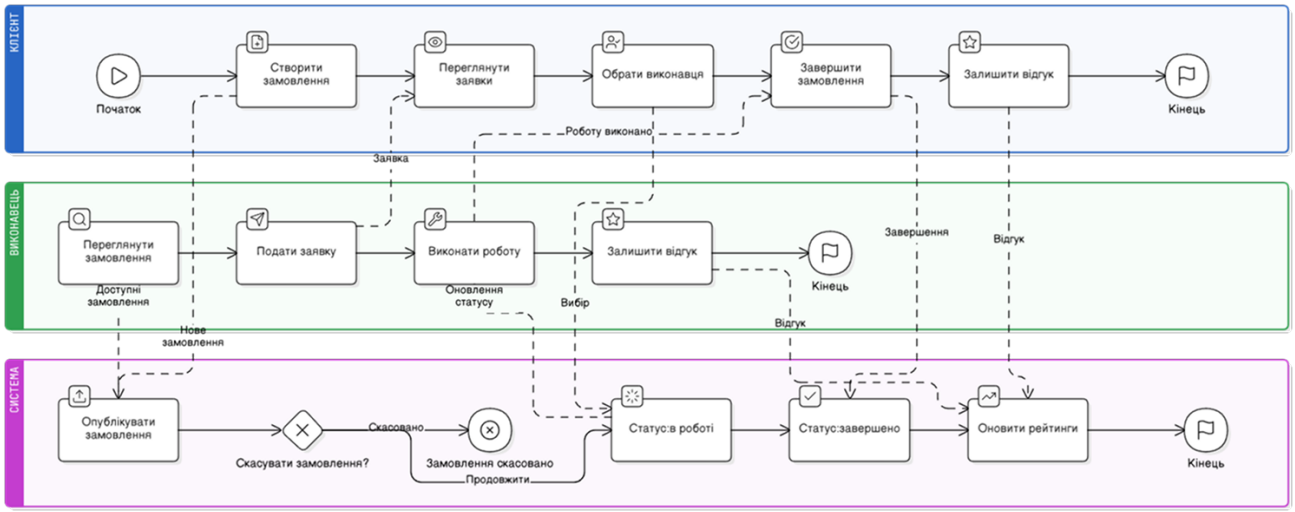


Рис 2.2 BPMN-діаграма процесу виконання замовлення

Замовлення може перебувати в одному з чотирьох основних станів: відкрите, у процесі виконання, завершене або скасоване. Відкритий стан означає, що замовлення доступне для перегляду та прийому заявок. Стан виконання означає,

що замовник уже обрав виконавця, тому нові заявки не приймаються. Завершений стан фіксує факт виконання роботи й відкриває можливість залишити відгук. Скасований стан використовується у випадку, коли робота не відбулася або замовлення втратило актуальність. У вимогах до системи визначено саме такі статуси: open, in_progress, completed і cancelled, причому для відкритого замовлення доступне подання заявок, а після переходу в процес виконання така можливість має бути недоступною.

Моделювання профілю користувача є важливим, оскільки профіль виконує функцію накопичення інформації про активність, надійність і досвід учасника платформи. Профіль повинен містити базову інформацію про користувача, аватар, опис або біографію, статуси прив’язки Steam і Discord, загальний показник репутації, а також історію опублікованих і виконаних замовлень. Для замовника профіль виконавця є джерелом інформації для прийняття рішення про співпрацю. Для виконавця профіль замовника дозволяє оцінити надійність потенційного клієнта. У функціональній моделі профіль пов’язаний із більшістю основних процесів: автентифікацією, створенням замовлень, поданням заявок, вибором виконавця та формуванням відгуків. Вимоги до профілю передбачають наявність аватара, бейджів прив’язки Steam і Discord, опису, загальної репутації та історії опублікованих і виконаних замовлень.

Окремим функціональним блоком є рейтинг і відгуки. Після завершення замовлення сторони отримують можливість оцінити одна одну. Така модель дозволяє формувати репутацію не довільно, а на основі завершених взаємодій. Рейтинг може підвищуватися або знижуватися залежно від оцінки, а текстовий коментар дає додатковий контекст. Для фриланс-платформи це особливо важливо, оскільки числовий показник сам по собі не завжди пояснює причину довіри або недовіри. Наприклад, користувач може мати загалом позитивну репутацію, але окремі відгуки можуть свідчити про проблеми з термінами, комунікацією або якістю виконання. У функціональній моделі відгук повинен бути пов’язаний із конкретним завершеним замовленням, щоб запобігти довільному впливу на репутацію.

Підбір виконавців у системі моделюється як процес інформаційної підтримки рішення замовника. На базовому рівні система не повинна автоматично призначати виконавця без участі замовника, оскільки остаточний вибір залежить від багатьох суб’єктивних і проєктних факторів. Натомість платформа повинна надати замовнику структуровані дані для порівняння кандидатів: запропоновану ціну, строк виконання, коментар до заявки, рейтинг виконавця, історію виконаних замовлень, наявність прив’язаних зовнішніх облікових записів і відповідність категорії робіт. У перспективі ці параметри можуть бути використані для побудови рекомендаційного механізму, який ранжуватиме заявки за релевантністю, але в межах базової функціональної моделі головним залишається прозоре представлення інформації.

Для формалізації функціональності системи доцільно використовувати декілька типів моделей, кожна з яких описує окремий аспект роботи платформи. Діаграма варіантів використання (рис. 2.1) дозволяє визначити, які дії виконують основні учасники системи. Діаграма життєвого циклу замовлення (рис 2.2) відображає допустимі переходи між станами замовлення. Концептуальна модель даних описує основні сутності предметної області та зв’язки між ними. Сукупність цих моделей дає змогу перейти від загального опису платформи до більш формалізованого представлення її поведінки.

У межах моделювання особливу увагу слід приділити зв’язку між діями користувачів і зміною стану замовлення. Наприклад, створення замовлення формує новий об’єкт зі статусом open, подання заявки не змінює статус замовлення, але створює пов’язаний із ним запис пропозиції виконавця. Вибір виконавця змінює статус замовлення на in_progress. Завершення роботи переводить його в completed. Скасування – у cancelled. Така модель дозволяє чітко визначити, які операції є допустимими на кожному етапі та які дані мають бути створені або змінені в результаті дії користувача.

Табл. 2.3

Зв’язок між діями користувача і зміною стану замовлення

Дія користувача	Початковий стан	Результат виконнання
Створення замовлення	Замовлення відсутнє	Створено нове замовлення зі статусом “open”
Подання заявки	Замовлення в статусі “open”	Створено нову заявку, статус замовлення не змінюється
Вибір виконавця	Замовлення в статусі “open”	Визначено виконавця, статус змінено на “in_progress”
Завершення замовлення	Замовлення в статусі “in_progress”	Замовлення переведено в статус “completed”
Скасування замовлення	Замовлення в статусі “in_progress” або “open”	Замовлення переведено в “cancelled”
Залишення відгуку	Замовлення в статусі “completed”	Створено відгук і оновлено репутаційні показники

Концептуальна модель даних використовується для відображення об’єктів, які виникають у процесі роботи системи. У центрі моделі знаходиться користувач, оскільки саме він може виступати як замовником, так і виконавцем. Профіль містить публічну інформацію про користувача та його репутацію. Замовлення пов’язується із замовником, категорією, заявками виконавців і, після вибору, з обраною заявкою. Заявка фіксує пропозицію конкретного виконавця щодо конкретного замовлення. Відгук пов’язується із завершеним замовленням, автором оцінки та користувачем, якому ця оцінка адресована. Така структура дозволяє забезпечити простежуваність взаємодії між сторонами та уникнути ситуацій, коли репутаційні оцінки існують без факту виконаної роботи.

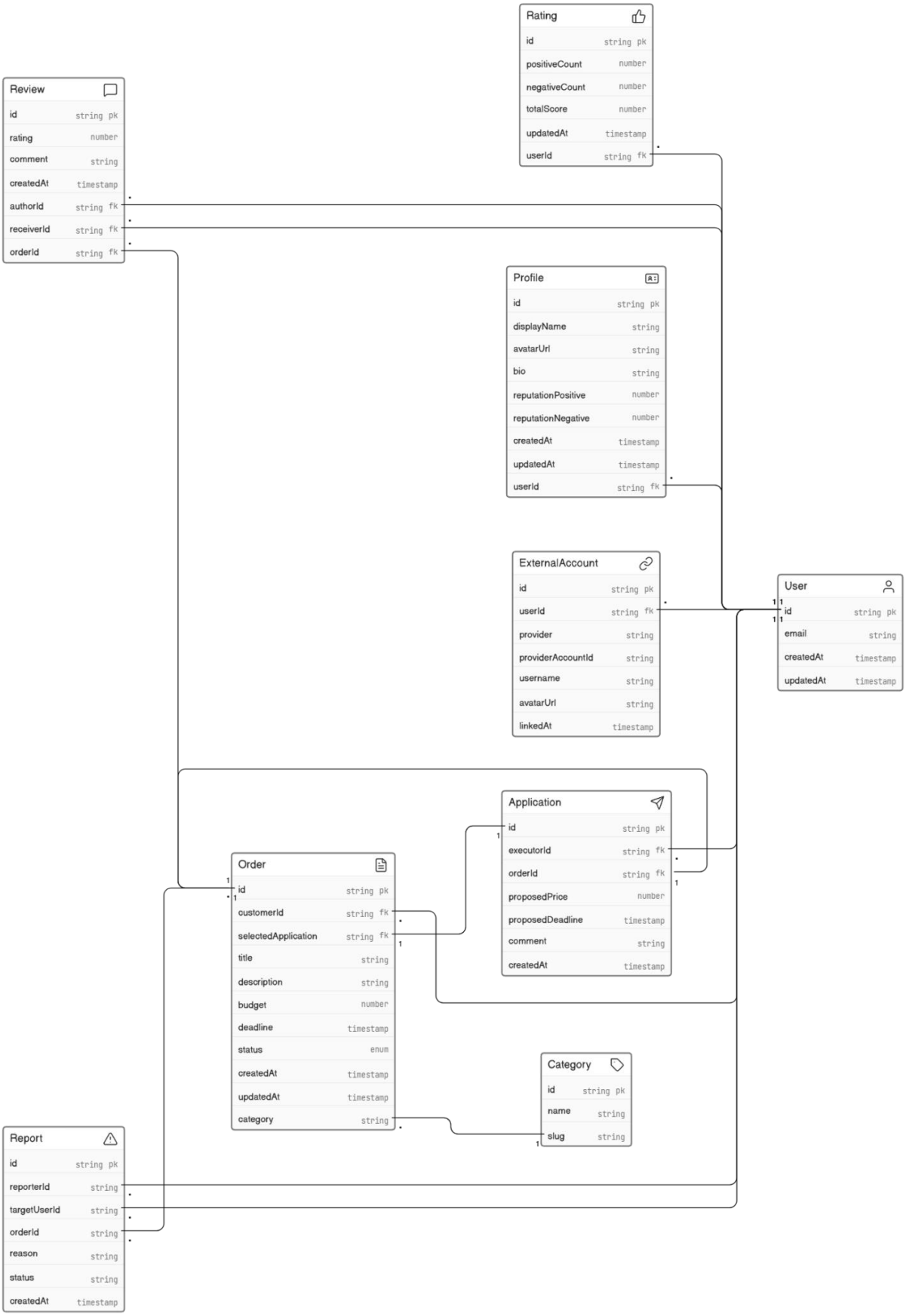


Рис 2.9 Концептуальна (ER-діаграмма) модель даних

Таке моделювання дозволяє не повторювати архітектурний опис системи, а показати логіку її роботи через формальні моделі: хто виконує дії, у яких станах може перебувати замовлення, які дані створюються під час взаємодії та як ці дані пов’язані між собою. Саме ці моделі є основою для подальшої реалізації функціоналу, оскільки вони визначають правила переходів, обмеження на дії користувачів і структуру інформації, яка повинна зберігатися в системі.

2.3 Обґрунтування рішень щодо масштабування та безпеки платформи

Питання масштабування та безпеки є важливими для платформи, оскільки система працює з користувацькими профілями, історією замовлень, заявками виконавців, репутаційними показниками та зовнішніми обліковими записами. Навіть на етапі мінімально життєздатної версії платформа повинна бути спроєктована таким чином, щоб її можна було поступово розширювати без повної перебудови архітектури. Водночас вона має забезпечувати базовий рівень захисту даних, контроль доступу до дій користувачів і стійкість до типових зловживань, характерних для онлайн-маркетплейсів і фриланс-платформ.

Масштабування платформи доцільно розглядати у двох напрямах: функціональному та технічному. Функціональне масштабування означає можливість поступового додавання нових можливостей після реалізації базового набору функцій. Для MVP передбачено автентифікацію, створення та перегляд замовлень, заявки виконавців, вибір виконавця, завершення замовлення, профілі та рейтинги. Саме ці функції формують основний цикл взаємодії між замовником і виконавцем. Після перевірки працездатності цього циклу система може бути розширена додатковими модулями: повідомленнями між користувачами, розширеним портфоліо, системою скарг, модерацією спорів, розширеними фільтрами пошуку, рекомендаційним алгоритмом, платіжними інструментами або інтеграціями з Discord.

Технічне масштабування пов’язане зі здатністю системи обробляти збільшення кількості користувачів, замовлень, заявок, переглядів профілів і

репутаційних записів. На початковому етапі для платформи достатнім є розгортання на VPS із використанням Coolify, що відповідає вимогам базової MVP-версії. Такий підхід є доцільним, оскільки дозволяє швидко розгорнути застосунок, базу даних і допоміжні сервіси без надмірних витрат на інфраструктуру. Водночас обрана модель не обмежує подальший розвиток: за необхідності веб-застосунок, база даних і додаткові сервіси можуть бути винесені на окремі сервери або масштабовані незалежно один від одного.

Важливим рішенням щодо масштабування є використання реляційної бази даних PostgreSQL. Предметна область платформи має чітко структуровану модель даних: користувачі пов’язані з профілями, зовнішніми акаунтами, замовленнями, заявками, рейтингами та відгуками. Така структура добре відповідає реляційному підходу, оскільки дозволяє підтримувати цілісність зв’язків між сутностями. Для масштабування це важливо тому, що зростання кількості записів не повинно призводити до втрати логічної узгодженості даних. Наприклад, заявка має бути пов’язана з конкретним замовленням і виконавцем, відгук – із завершеним замовленням і його учасниками, а рейтинг – із конкретним користувачем. Збереження таких зв’язків є основою коректної роботи платформи.

Окрему роль у масштабуванні відіграє модульний принцип побудови системи. Навіть якщо платформа реалізується як єдиний full-stack застосунок, її логіку доцільно поділяти на окремі функціональні частини: автентифікацію, профілі, замовлення, заявки, рейтинги, відгуки та адміністрування. Такий підхід дозволяє змінювати або розширювати один функціональний блок без суттєвого впливу на інші. Наприклад, система рейтингів може спочатку працювати за простою моделлю позитивних і негативних оцінок, а в майбутньому бути доповнена ваговими коефіцієнтами, текстовим аналізом відгуків або окремими показниками для різних категорій робіт. Аналогічно, модуль замовлень може бути розширений підтримкою етапів виконання, вкладень або проміжного погодження результатів.

З погляду продуктивності важливо передбачити оптимізацію операцій, які потенційно виконуватимуться найчастіше. До таких операцій належать перегляд

списку замовлень, фільтрація за категоріями, бюджетом і статусом, відкриття профілів користувачів, перегляд заявок і підрахунок репутаційних показників. Для цього дані повинні бути організовані таким чином, щоб система могла швидко отримувати найбільш затребувану інформацію. Наприклад, список замовлень не повинен щоразу завантажувати надмірну кількість супровідних даних, а профіль користувача має відображати агреговану репутацію без необхідності щоразу повністю перераховувати всі оцінки. Такий підхід дозволяє зберігати швидкодію навіть за поступового збільшення кількості користувачів і замовлень.

Для подальшого масштабування також важливим є розділення публічних і захищених операцій. Публічні сторінки, такі як перегляд відкритих замовлень або профілів, можуть отримувати значно більше запитів, ніж дії, доступні лише авторизованим користувачам. Натомість створення замовлення, подання заявки, вибір виконавця або залишення відгуку є критичними операціями, які повинні проходити через перевірку прав доступу. Такий поділ дозволяє в майбутньому по-різному оптимізувати різні частини системи: публічні сторінки – для швидкого перегляду, а захищені операції – для надійності, перевірки та цілісності даних.

Безпека платформи є не менш важливою, ніж масштабованість, оскільки система працює з довірою між користувачами. Основними об’єктами захисту є облікові записи, профілі, зовнішні прив’язки Steam і Discord, замовлення, заявки, статуси виконання, відгуки та репутаційні показники. У вимогах до системи передбачено використання захищених маршрутів, захисту на рівні API-процедур, розмежування ролей і антиспам-функціональності. Це відповідає характеру платформи, оскільки кожна важлива дія повинна виконуватися лише тим користувачем, який має на неї право.

Першим рівнем безпеки є аутентифікація користувачів. Користувач повинен увійти до системи перед виконанням дій, які змінюють дані: створенням замовлення, поданням заявки, вибором виконавця, завершенням замовлення або залишенням відгуку. При цьому публічний перегляд окремих даних може залишатися доступним без входу, якщо це не створює ризику для приватності або безпеки. Для Garry’s Mod-спільноти важливою є підтримка прив’язки Steam і

Discord, оскільки ці облікові записи виступають додатковими маркерами ідентичності користувача. Статус прив’язки таких акаунтів може підвищувати довіру до профілю, однак система не повинна розкривати зайві персональні дані без потреби.

Другим рівнем безпеки є авторизація, тобто перевірка прав користувача на виконання конкретної дії. Авторизація має бути реалізована не лише на рівні інтерфейсу, а й на серверному рівні. Наприклад, інтерфейс може приховувати кнопку вибору виконавця від сторонніх користувачів, але серверна логіка все одно повинна перевіряти, чи справді запит на вибір виконавця надійшов від автора відповідного замовлення. Це необхідно для запобігання ситуаціям, коли користувач намагається виконати заборонену дію шляхом прямого звернення до API.

Третім важливим напрямом безпеки є захист репутаційної системи. Рейтинг є одним із головних механізмів довіри на платформі, тому він не повинен бути доступним для довільного маніпулювання. Відгуки мають створюватися лише після завершення замовлення й лише між користувачами, які брали участь у конкретній взаємодії. Це дозволяє уникнути ситуацій, коли користувачі штучно підвищують рейтинг один одного або залишають негативні оцінки без факту співпраці. Крім того, важливо зберігати зв’язок відгуку з конкретним замовленням, щоб у разі спору можна було перевірити контекст оцінки.

Четвертим напрямом є захист від спаму та недобросовісної активності. Для фриланс-платформи потенційними ризиками є масове створення неякісних замовлень, надмірна кількість заявок від одного користувача, дублювання оголошень, створення фейкових акаунтів, маніпуляції рейтингом і публікація некоректного контенту. На початковому етапі антиспам-механізми можуть бути базовими: обмеження частоти створення замовлень або заявок, перевірка авторизації, рольова модерація, можливість скарги на користувача або замовлення. У майбутньому ці механізми можуть бути розширені автоматичним виявленням підозрілої активності та більш гнучкими правилами модерації.

П’ятий напрям безпеки пов’язаний із цілісністю станів замовлення. Система повинна запобігати некоректним переходам між статусами. Наприклад, завершити можна лише те замовлення, яке перебуває в процесі виконання; подати заявку можна лише на відкрите замовлення; після вибору виконавця нові заявки не повинні прийматися; відгук можна залишити лише після завершення роботи. Такі правила є не тільки бізнес-логікою, а й елементом захисту від помилок і зловживань. Якщо система дозволятиме довільно змінювати статуси, це може призвести до конфліктів, втрати довіри й некоректного формування репутації.

Окремо слід враховувати безпеку персональних і публічних даних. Профіль користувача має містити лише ту інформацію, яка необхідна для взаємодії на платформі: аватар, опис, статуси прив’язки акаунтів, репутацію та історію замовлень. Дані, що використовуються для автентифікації або внутрішньої роботи системи, не повинні розкриватися іншим користувачам. Особливо це стосується ідентифікаторів зовнішніх облікових записів, службової інформації, сесійних даних і внутрішніх технічних параметрів. Публічний профіль має бути інформативним для прийняття рішення про співпрацю, але не повинен створювати зайвих ризиків для приватності.

Також важливо передбачити адміністративний контроль. Навіть добре спроєктована система не може повністю виключити конфлікти між замовниками й виконавцями, тому платформа повинна мати можливість реагувати на порушення. Адміністратор або модератор може перевіряти скарги, обмежувати недобросовісну активність, приховувати некоректний контент або втручатися у випадках очевидного зловживання. Такий механізм є важливим для підтримки довіри до платформи, особливо на початковому етапі, коли репутаційна система ще не має великого обсягу накопичених даних.

З погляду інфраструктури безпека повинна включати захищене розгортання, контроль доступу до середовища, регулярне оновлення залежностей і базові засоби резервного копіювання. Оскільки платформа зберігає важливі для користувачів дані, резервне копіювання бази даних є необхідною умовою надійної експлуатації. Втрата історії замовлень або репутаційних записів може суттєво знизити довіру до

сервісу. Тому навіть для MVP доцільно передбачити можливість відновлення даних у разі технічного збою.

Таким чином, рішення щодо масштабування та безпеки платформи спрямовані на створення системи, яка може почати роботу як відносно компактний MVP, але не буде обмежена цим форматом у майбутньому. Масштабування забезпечується за рахунок модульної логіки, реляційної структури даних, можливості поступового додавання нових функцій і використання інфраструктури, придатної для подальшого розширення. Безпека забезпечується через автентифікацію, авторизацію, захищені маршрути, перевірку прав на рівні серверної логіки, захист репутаційної системи, контроль статусів замовлень, антиспам-механізми та адміністративну модерацію. Такий підхід дозволяє створити платформу, яка буде не лише функціонально придатною для Garry’s Mod-спільноти, а й достатньо надійною для зберігання даних, підтримки довіри та подальшого розвитку.

РОЗДІЛ 3. РОЗРОБКА, ІНТЕГРАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ПЛАТФОРМИ МАРКЕТПЛЕЙСУ ФРІЛАНС-ПОСЛУГ

3.1. Опис реалізації основних функціональних модулів

Реалізація веб-платформи управління замовленнями для Garry’s Mod-спільноти виконувалась із використанням сучасного full-stack підходу, при якому клієнтська та серверна частини об’єднані в межах одного застосунку. Основою системи виступає Next.js із використанням App Router, що дозволило поєднати серверний рендеринг, API-логіку та клієнтські інтерфейси в єдиній архітектурі. Для забезпечення типобезпечної взаємодії між frontend і backend було використано tRPC, а для роботи з базою даних – Prisma ORM у поєднанні з PostgreSQL. Такий стек дозволив реалізувати модульну, масштабовану та придатну до подальшого розвитку систему.

У процесі реалізації платформа була поділена на окремі функціональні модулі, кожен із яких відповідає за певний набір бізнес-процесів. Основними модулями стали модуль автентифікації, модуль профілів користувачів, модуль замовлень, модуль рейтингу та відгуків, а також адміністративно-модераційний функціонал. Подібний підхід дозволив розділити логіку системи за предметними областями та уникнути надмірної залежності між окремими частинами застосунку.

Модуль автентифікації забезпечує ідентифікацію користувачів, створення сесії та контроль доступу до захищеного функціоналу. Для реалізації було використано Better Auth із підтримкою OAuth-провайдерів. Основними зовнішніми сервісами авторизації виступають Discord і Steam, оскільки саме ці платформи є найбільш поширеними серед Garry’s Mod-спільноти. Конфігурація Better Auth представлена у вигляді об’єкту (рис 3.1) з предетермінованими ключами.

Після успішної авторизації система автоматично створює запис користувача в базі даних та формує профіль із базовою інформацією. Такий підхід дозволяє мінімізувати кількість дій, необхідних для початку роботи з платформою. Додатково реалізовано механізм прив’язки зовнішніх акаунтів, що дозволяє

відображати відповідні бейджі в профілі користувача та підвищувати рівень довіри між учасниками платформи.

auth.ts TypeScript

```typescript
export const auth = betterAuth({
  database: prismaAdapter(createPrismaClient(), { provider: "postgresql" }),
  account: {
    accountLinking: {
      allowDifferentEmails: true,
    },
  },
  socialProviders: {
    discord: {
      clientId: env.DISCORD_CLIENT_ID,
      clientSecret: env.DISCORD_CLIENT_SECRET,
    },
  },
  plugins: [
    genericOAuth({
      config: [
        {
          providerId: "steam",
          clientId: env.STEAM_CLIENT_ID,
          clientSecret: env.STEAM_CLIENT_SECRET,

          discoveryUrl: new URL(
            "/.well-known/openid-configuration",
            env.STEAM_IDP_URL,
          ).href,
          scopes: ["openid", "profile"],
          disableSignUp: true,

          mapProfileToUser: (profile) => ({
            email: profile.email ?? `${profile.steam_id}@${env.STEAM_IDP_URL}`,
            id: profile.steam_id,
          }),
        },
      ],
    }),
    nextCookies(),
  ],
});
```

Рис 3.1 Конфігурація BetterAuth

Для захисту функціоналу платформи реалізовано middleware-перевірки та protected procedures на рівні tRPC (рис 3.2). Завдяки цьому створення замовлень,

подання заявок, залишення відгуків та інші критичні дії доступні лише авторизованим користувачами.

trpc/init.ts TypeScript

```typescript
export const createTRPCContext = async (opts: { headers: Headers }) => {
  const session = await auth.api.getSession({
    headers: opts.headers,
  });
  // Ensure a Profile exists for first-time sign-ins
  if (session?.user?.id) {
    try {
      const { createProfileIfMissing } =
        await import("@/server/modules/profiles/service");
      await createProfileIfMissing(
        session.user.id,
        session.user.name,
        session.user.image,
      );
    } catch (e) {
      // Don't block context creation on profile creation errors
      console.error("Failed to ensure profile for user", e);
    }
  }

  return {
    auth: null,
    session,
  };
};

export const protectedProcedure = t.procedure
  .use(timingMiddleware)
  .use(({ ctx, next }) => {
    if (!ctx.session) {
      throw new TRPCError({
        code: "UNAUTHORIZED",
        message: "You must be logged in to access this resource.",
        cause: "No session found.",
      });
    }

    return next({
      ctx: {
        ...ctx,
        session: ctx.session,
      },
    });
  });
```

Рис 3.2 Налаштування контексту виконання процедур tRPC та захищена процедура.

Модуль профілів відповідає за зберігання та відображення інформації про користувача. Кожен профіль містить ім’я, аватар, опис, прив’язані зовнішні акаунти, репутаційні показники та історію активності. Інформація профілю використовується як основний механізм оцінювання надійності користувача перед початком співпраці.

У межах реалізації було створено окремі серверні сервіси для отримання профілю користувача, оновлення персональної інформації та агрегації статистики. Для оптимізації продуктивності репутаційні дані не обчислюються динамічно під час кожного відкриття профілю, а зберігаються в агрегованому вигляді.

Інтерфейс профілю, показаного на рисунку 3.3, реалізовано як окрему сторінку з використанням server components для первинного завантаження даних і client components для інтерактивних елементів редагування.

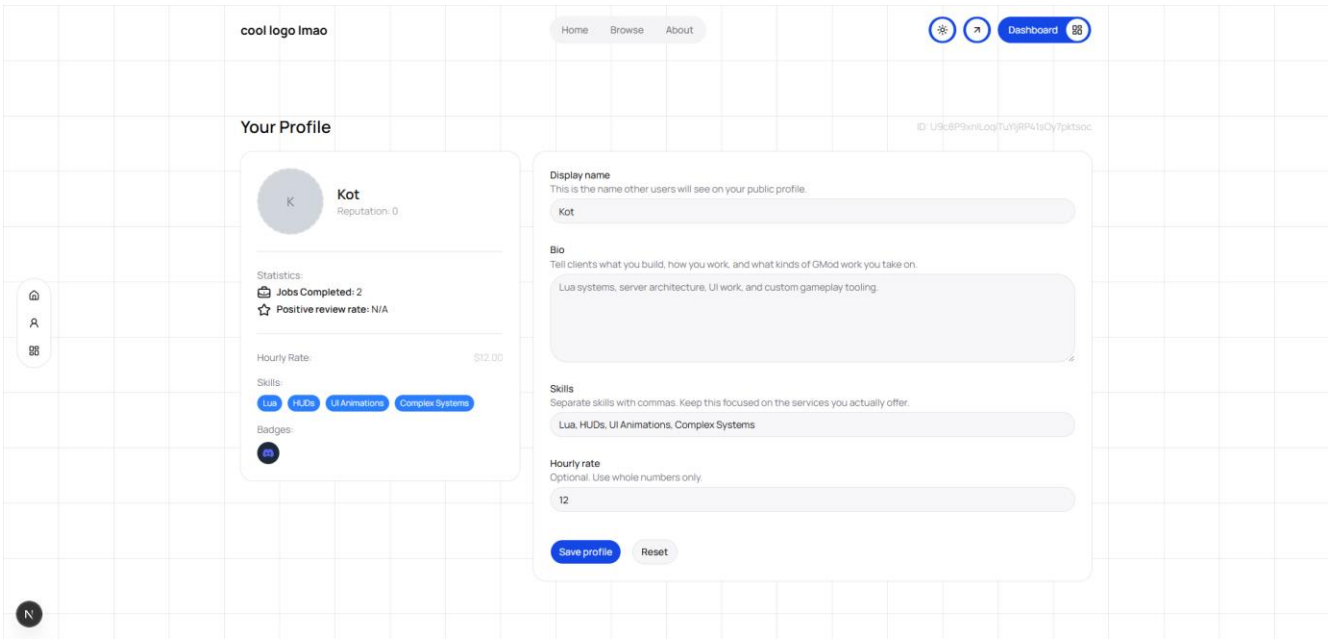


Рис 3.3 Інтерфейс редагування профілю в панелі керування

Модуль замовлень є центральним елементом системи, оскільки саме навколо нього побудована взаємодія між замовниками та виконавцями. Реалізація модуля включає створення, редагування, перегляд, фільтрацію та зміну статусу замовлень.

Замовлення зберігається в базі даних як окрема сутність із прив'язкою до автора, категорії та статусу. Реалізація моделі такої сутності представлена на рисунку 3.4. Для забезпечення цілісності даних використано enum-типи Prisma для статусів замовлення (OPEN, IN_PROGRESS, COMPLETED, CANCELLED). Такий підхід дозволяє обмежити допустимі стани замовлення на рівні бази даних.

prisma/schema/shema.prisma — Prisma

```
enum OrderStatus {
  OPEN
  IN_PROGRESS
  COMPLETED
  CANCELLED
}

model Order {
  id String @id @default(uuid())

  customerId String
  customer   User   @relation(fields: [customerId], references: [id], onDelete: Cascade)

  acceptedApplicantId String?
  acceptedApplicant   User?   @relation("acceptedContractor", fields:
[acceptedApplicantId], references: [id], onDelete: SetNull)

  title       String
  description String

  categoryId String
  category   Category @relation(fields: [categoryId], references: [id])

  budgetMin Int
  budgetMax Int
  deadline  DateTime

  status OrderStatus @default(OPEN)

  createdAt DateTime @default(now())
  updatedAt DateTime @updatedAt

  applications Application[]
  reviews      Review[]
  reports      Report[]

  @@index([customerId])
  @@index([acceptedApplicantId])
  @@index([status])
  @@index([deadline])
}
```

Рис 3.4 Представлення моделі даних в PrismaORM

Сторінка перегляду замовлень реалізована у вигляді каталогу з фільтрацією за назвою, категоріями та бюджетом (рис 3.5). Для оптимізації навантаження використовується серверне отримання даних із можливістю пагінації.

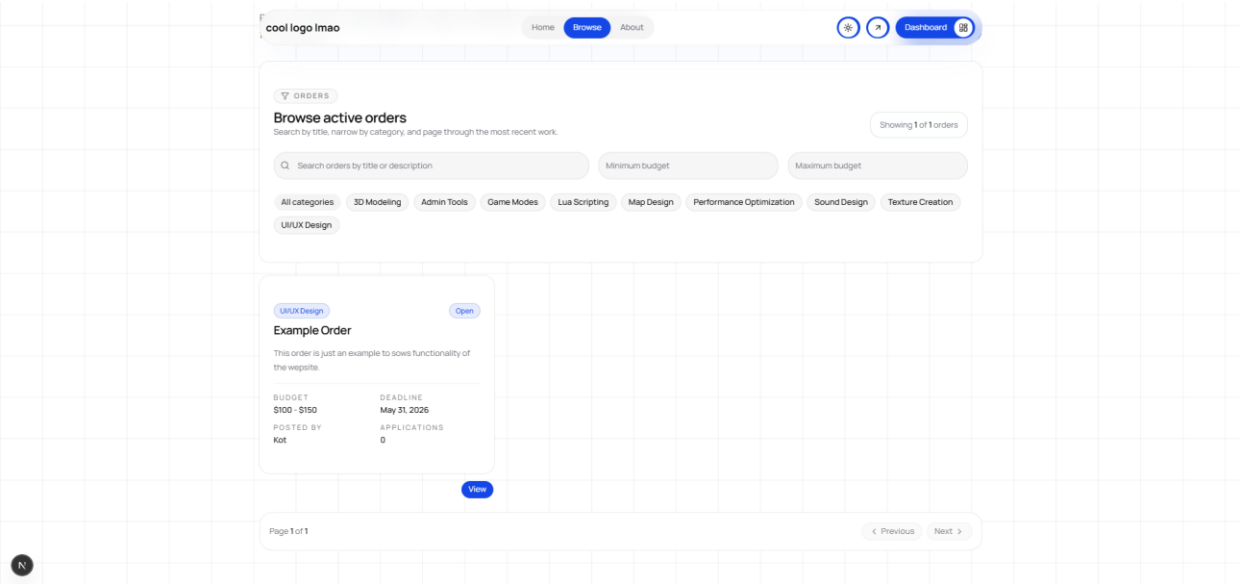


Рис 3.5 Сторінка перегляду активних замовлень

Окремо реалізовано сторінку детального перегляду замовлення (рис 3.6), яка містить опис роботи, бюджет, дедлайн, інформацію про автора та список заявок виконавців.

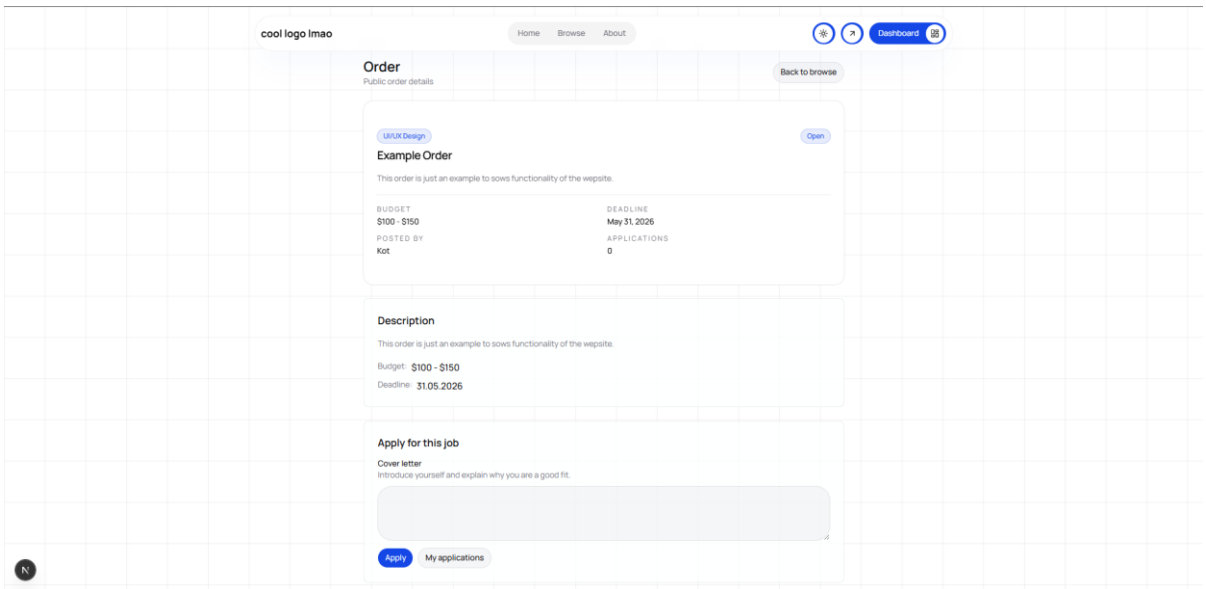


Рис 3.6 Сторінка детального перегляду замовлення.

Модуль заявок забезпечує механізм подання пропозицій виконавцями. Кожна заявка пов’язується з конкретним замовленням і користувачем. Під час реалізації було враховано низку обмежень бізнес-логіки:

- користувач не може подати заявку на власне замовлення;
- подання заявки можливе лише для відкритих замовлень;
- після вибору виконавця подання нових заявок блокується.

Реалізація функції з наведеною бізнес-логікою представлена на рисунку 3.7. Усі перевірки виконуються на серверному рівні перед створенням запису в базі даних.

server/modules/orders/service.ts TypeScript

```typescript
export const applyToOrder = async (userId: string, dto: ApplyToOrderInput) => {
  // Check if user already applied
  const existing = await prisma.application.findUnique({
    where: {
      orderId_applicantId: {
        orderId: dto.orderId,
        applicantId: userId,
      },
    },
  });

  if (existing) {
    throw new Error("You have already applied to this order");
  }

  // Check if order exists and is OPEN
  const order = await prisma.order.findUnique({
    where: { id: dto.orderId },
  });

  if (!order) {
    throw new Error("Order not found");
  }

  if (order.status !== OrderStatus.OPEN) {
    throw new Error("This order is no longer accepting applications");
  }

  if (order.customerId === userId) {
    throw new Error("You cannot apply to your own order");
  }

  return prisma.application.create({
    data: {
      id: randomUUID(),
      orderId: dto.orderId,
      applicantId: userId,
      coverLetter: dto.coverLetter,
    },
    include: {
      applicant: {
        select: { id: true, name: true, image: true, profile: true },
      },
      order: true,
    },
  });
};
```

Рис 3.7 Функція подання заявки на замовлення

Після подання заявки замовник отримує можливість переглядати список кандидатів і порівнювати їх за репутацією, запропонованою ціною та строком виконання. Процес вибору виконавця реалізовано як окрему серверну процедуру яка одночасно фіксує обраного виконавця, змінює статус замовлення та блокує можливість подання нових заявок. Такий підхід дозволяє уникнути неузгоджених станів системи.

Для підтримки безпеки платформи реалізовано базову систему модерації та скарг. Користувач може поскаржитися на інший профіль або замовлення, вказавши причину звернення. Скарга створюється як окремий запис у базі даних і може бути оброблена адміністрацією.

На етапі MVP модераційний функціонал має спрощений характер, однак архітектура передбачає можливість подальшого розширення системи, зокрема:

- введення ролей модераторів;
- автоматичне блокування контенту;
- історію адміністративних дій;
- систему попереджень.

3.2. Розробка алгоритму автоматизованого підбору виконавців

Однією з ключових задач веб-платформи управління замовленнями у сфері Garry’s Mod-фрилансу є спрощення процесу пошуку відповідного виконавця для конкретного замовлення. У межах традиційних систем взаємодії замовник змушений самостійно аналізувати значну кількість кандидатів, переглядати профілі, оцінювати репутацію, порівнювати вартість і терміни виконання. Такий підхід ускладнює процес вибору, особливо в умовах великої кількості заявок і відсутності структурованої оцінки релевантності виконавців. Для вирішення цієї проблеми в межах платформи було реалізовано алгоритм автоматизованої підтримки підбору виконавців, основною задачею якого є формування впорядкованого списку кандидатів відповідно до визначених критеріїв оцінювання.

Особливістю Garry’s Mod-фрилансу є різноманітність напрямів діяльності. Частина замовлень пов’язана з Lua-розробкою та серверною логікою, інші – зі створенням UI, моделей, текстур, карт або конфігурацій. Через це алгоритм не може оцінювати виконавців лише за загальним рейтингом, оскільки користувач із високою репутацією в категорії дизайну інтерфейсів не обов’язково буде оптимальним кандидатом для програмування серверних систем. Саме тому при розробці алгоритму було враховано не лише загальні показники активності, а й контекст попередньої діяльності виконавця.

Реалізований алгоритм виконує функцію рекомендаційного ранжування заявок. Система не призначає виконавця автоматично та не виключає участі замовника у процесі прийняття рішення. Натомість вона аналізує характеристики кандидатів і сортує заявки відповідно до їхньої релевантності конкретному замовленню. Такий підхід є доцільним для фриланс-платформи, оскільки остаточний вибір часто залежить не лише від формальних критеріїв, а й від суб’єктивних факторів: стилю комунікації, творчого бачення, специфіки завдання або індивідуальних побажань замовника.

Основою алгоритму є система інтегрального оцінювання, у межах якої кожен виконавець отримує певний рейтинг відповідно до набору параметрів. Під час розрахунку враховується репутація користувача, кількість завершених замовлень, відповідність категорії роботи попередньому досвіду виконавця, запропонована ціна, термін виконання, а також додаткові фактори довіри, пов’язані з активністю користувача та прив’язкою зовнішніх акаунтів. Для кожного параметра визначається окремий ваговий коефіцієнт, після чого формується загальний показник релевантності.

Загальна формула оцінювання має вигляд:

$$R = \omega_1 * P + \omega_2 * C + \omega_3 * S + \omega_4 * B + \omega_5 * D + \omega_6 * A, \qquad (3.1)$$

де:

R – інтегральний рейтинг виконавця;

P – показник репутації;

C – кількість завершених замовлень;

S – відповідність спеціалізації категорії замовлення;

B – коефіцієнт відповідності бюджету;

D – коефіцієнт терміну виконання;

A – додаткові фактори довіри;

ω_n – вагові коефіцієнти окремих критеріїв.

У межах MVP-версії найбільший вплив на результат мають репутаційні показники, кількість успішно завершених робіт і відповідність категорії замовлення. Таке рішення пояснюється тим, що саме ці параметри найбільше впливають на рівень довіри між користувачами. Наприклад, виконавець із великим досвідом у категорії Lua-розробки отримує вищий коефіцієнт релевантності для серверного програмування, ніж користувач, основна активність якого пов’язана зі створенням текстур або моделей.

Репутаційний показник формується на основі співвідношення позитивних і негативних відгуків, однак система враховує також загальну кількість завершених замовлень. Це дозволяє уникнути ситуацій, коли користувач із одним позитивним відгуком отримує такий самий рівень довіри, як виконавець із десятками успішно завершених проєктів. Для стабілізації рейтингу використовується логарифмічна залежність від кількості відгуків:

$$P = \frac{positiveReviews}{totalReviews} * \log(1 + totalReviews), \qquad (3.2)$$

Такий підхід дозволяє зменшити вплив випадкових одиничних оцінок і зробити репутаційну систему більш стійкою до маніпуляцій.

Важливу роль у роботі алгоритму відіграє аналіз спеціалізації виконавця. Для цього система досліджує історію завершених замовлень і визначає, у яких категоріях користувач працював найчастіше. Якщо категорія нового замовлення відповідає попередній активності виконавця, алгоритм підвищує його

релевантність. Таким чином формується базовий механізм спеціалізації, який дозволяє рекомендувати кандидатів із найбільш релевантним досвідом. Спрощений принцип роботи алгоритму наведено на рисунку 3.8.

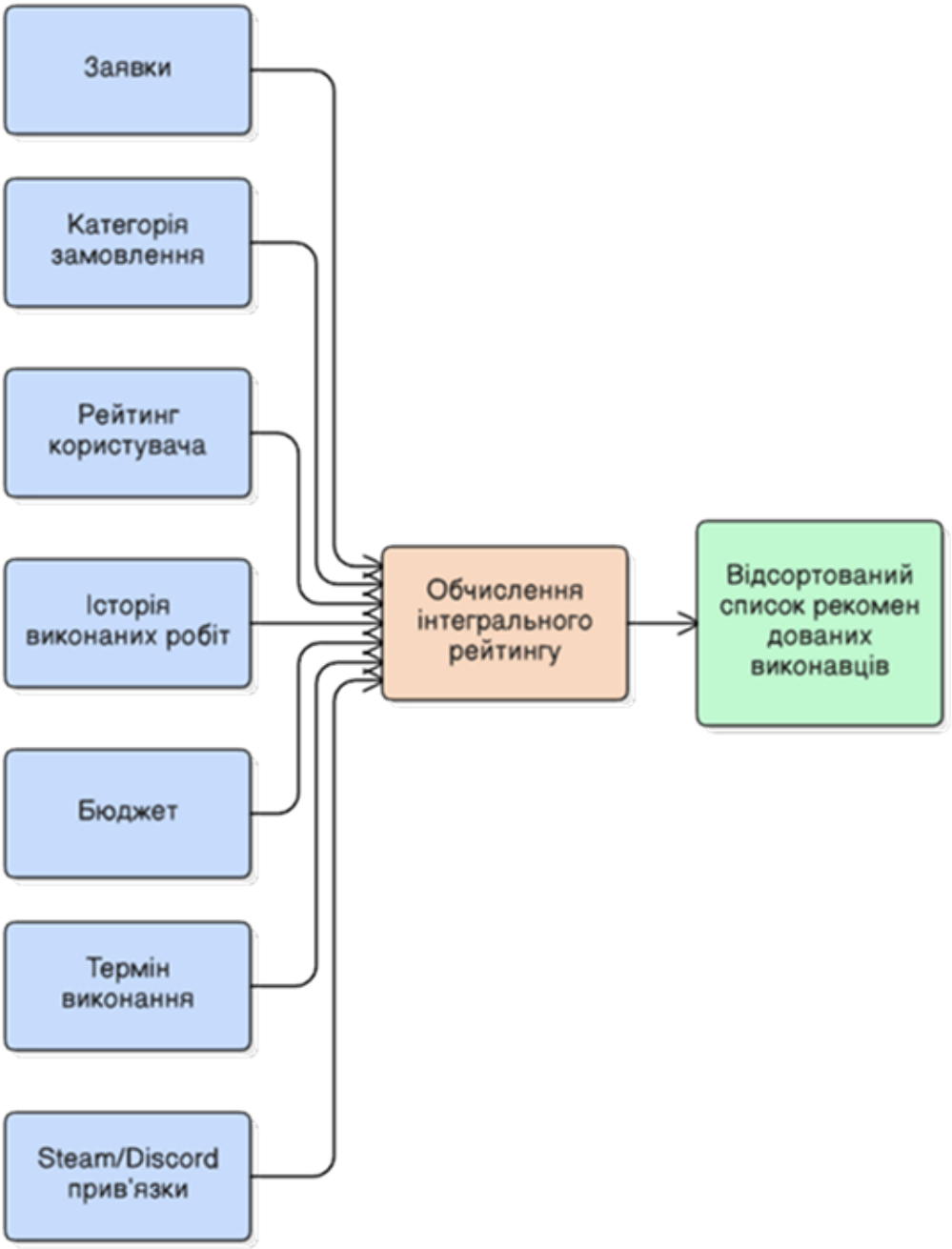


Рис 3.8 Спрощений принцип роботи алгоритму обчислення рейтингу виконавця

Під час оцінювання бюджету система аналізує різницю між очікуваним бюджетом замовника та запропонованою ціною виконавця. Якщо вартість суттєво

перевищує бюджет замовлення, коефіцієнт релевантності знижується. Водночас алгоритм не надає автоматичної переваги найдешевшим заявкам, оскільки низька ціна не завжди є показником високої якості роботи. Через це бюджетний коефіцієнт використовується як допоміжний критерій, а не як основний фактор ранжування.

Аналогічний підхід використовується для оцінювання терміну виконання. Якщо виконавець пропонує надто короткий або надмірно великий термін, система знижує відповідний коефіцієнт. Найбільш релевантними вважаються заявки, строки яких знаходяться близько до очікувань замовника. Це дозволяє уникнути ситуацій, коли нереалістичні обіцянки використовуються лише для підвищення привабливості заявки.

Додатковими параметрами оцінювання виступають фактори довіри. До них належать прив’язка Steam-акаунта, наявність Discord-профілю, тривалість активності на платформі, кількість завершених замовлень і відсутність скарг або санкцій. Ці показники мають меншу вагу порівняно з основними критеріями, однак вони можуть впливати на підсумкове сортування кандидатів у випадку близьких результатів.

Розрахунок рейтингу виконується на серверному рівні. Такий підхід забезпечує централізоване керування алгоритмом, унеможливлює маніпуляцію логікою сортування з боку клієнта та дозволяє змінювати формулу оцінювання без необхідності модифікації frontend-частини застосунку.

Після завершення розрахунку система сортує заявки у порядку спадання релевантності та передає результат замовнику. У результаті користувач отримує вже структурований список кандидатів, що значно скорочує час аналізу та спрощує процес вибору виконавця.

При реалізації алгоритму також були враховані обмеження предметної області. Виконавець не може бути рекомендований для власного замовлення, заблоковані користувачі виключаються з аналізу, а заявки на закриті або завершені замовлення не враховуються системою. Крім того, користувач без завершених робіт не може отримати максимальний рейтинг незалежно від інших параметрів, оскільки система повинна враховувати фактичний досвід взаємодії на платформі.

Важливою особливістю реалізованого підходу є його масштабованість. Архітектура алгоритму дозволяє додавати нові критерії оцінювання без повної перебудови системи. У перспективі це дає можливість інтегрувати механізми машинного навчання, персоналізовані рекомендації, аналіз тексту відгуків або автоматичне виявлення підозрілих заявок. Наприклад, система може аналізувати історію успішних співпраць і виявляти закономірності між типами замовлень та характеристиками виконавців, поступово переходячи від простого бального ранжування до складнішої рекомендаційної моделі.

Водночас автоматизований підбір виконавців не повинен повністю замінювати рішення людини. У сфері творчих і технічних послуг існує значна кількість факторів, які складно формалізувати математично. До них належать стиль роботи, креативність, особливості комунікації або індивідуальні вимоги замовника. Через це алгоритм у межах платформи виконує роль системи підтримки прийняття рішень, а не механізму автоматичного призначення виконавця.

Для оцінки ефективності алгоритму можуть використовуватися показники середнього часу вибору виконавця, кількості успішно завершених замовлень, частоти повторної співпраці між користувачами та кількості конфліктних ситуацій. Очікується, що використання системи автоматизованого ранжування дозволить скоротити час пошуку виконавців, покращити якість підбору кандидатів і підвищити загальний рівень довіри до платформи.

Таким чином, у межах веб-платформи було реалізовано алгоритм автоматизованої підтримки підбору виконавців, який базується на інтегральній системі оцінювання та враховує репутацію користувача, спеціалізацію, історію виконаних робіт, відповідність бюджету й термінів, а також додаткові фактори довіри. Реалізований підхід дозволяє структурувати процес вибору кандидатів, зменшити навантаження на замовника та створює основу для подальшого розвитку рекомендаційної системи платформи.

3.3. Налаштування, конфігурація та розгортання веб-платформи

Важливим етапом розробки веб-платформи стало налаштування програмного середовища, конфігурація серверної інфраструктури та підготовка системи до розгортання. Оскільки платформа проєктувалася як сучасний full-stack застосунок із можливістю подальшого масштабування, особлива увага приділялася стандартизації середовища виконання, автоматизації запуску сервісів і забезпеченню стабільної роботи застосунку в production-умовах.

Основою системи виступає Next.js-застосунок, який поєднує frontend і backend-логіку в межах єдиного проєкту. Для забезпечення роботи системи було налаштовано Node.js-середовище, менеджер пакетів pnpm [29], а також систему конфігурації змінних середовища через .env-файли.

Використання окремих змінних середовища дозволило ізолювати конфіденційні параметри застосунку від програмного коду та забезпечити гнучке налаштування системи залежно від середовища запуску. Для забезпечення функціональності застосунку, імплементована валідація середовища за допомогою T3 Env [30] наведена на рисунку 3.8.

lib/env.ts TypeScript

```typescript
export const env = createEnv({
  server: {
    DATABASE_URL: z.string(),
    BETTER_AUTH_SECRET: z.string(),
    BETTER_AUTH_URL: z.string(),

    // Auth
    DISCORD_CLIENT_ID: z.string(),
    DISCORD_CLIENT_SECRET: z.string(),

    STEAM_IDP_URL: z.string(),
    STEAM_CLIENT_ID: z.string(),
    STEAM_CLIENT_SECRET: z.string(),
  },
  client: {},

  runtimeEnv: {
    DATABASE_URL: process.env.DATABASE_URL,
    BETTER_AUTH_SECRET: process.env.BETTER_AUTH_SECRET,
    BETTER_AUTH_URL: process.env.BETTER_AUTH_URL,

    // Auth
    DISCORD_CLIENT_ID: process.env.DISCORD_CLIENT_ID,
    DISCORD_CLIENT_SECRET: process.env.DISCORD_CLIENT_SECRET,
    STEAM_IDP_URL: process.env.STEAM_IDP_URL,
    STEAM_CLIENT_ID: process.env.STEAM_CLIENT_ID,
    STEAM_CLIENT_SECRET: process.env.STEAM_CLIENT_SECRET,
  },

  emptyStringAsUndefined: true,
});
```

Рис 3.9 Валідація робочого середовища

Для роботи з базою даних PostgreSQL використовувалась Prisma ORM. Після налаштування з’єднання система автоматично застосовує міграції та синхронізує структуру бази даних із Prisma schema. Це дозволяє централізовано керувати моделями даних і спрощує процес оновлення структури БД у процесі розвитку платформи.

Для production-розгортання веб-платформи передбачено використання VPS-сервера з Linux-середовищем. Такий підхід забезпечує повний контроль над конфігурацією системи, дозволяє масштабувати інфраструктуру та оптимізувати

використання ресурсів. Розгортання може виконуватися як вручну через Docker Compose, так і за допомогою платформ автоматизованого деплою, зокрема Coolify.

Coolify використовується як self-hosted PaaS-рішення, яке автоматизує процес деплою контейнеризованих застосунків. У процесі налаштування платформа отримує доступ до Git-репозиторію, автоматично виконує збірку контейнерів і запускає сервіси після оновлення коду. Це дозволяє реалізувати CI/CD-підхід навіть у межах MVP-проєкту. Принцип роботи розгортання додатку з використанням Coolify наведено на рисунку 3.10.

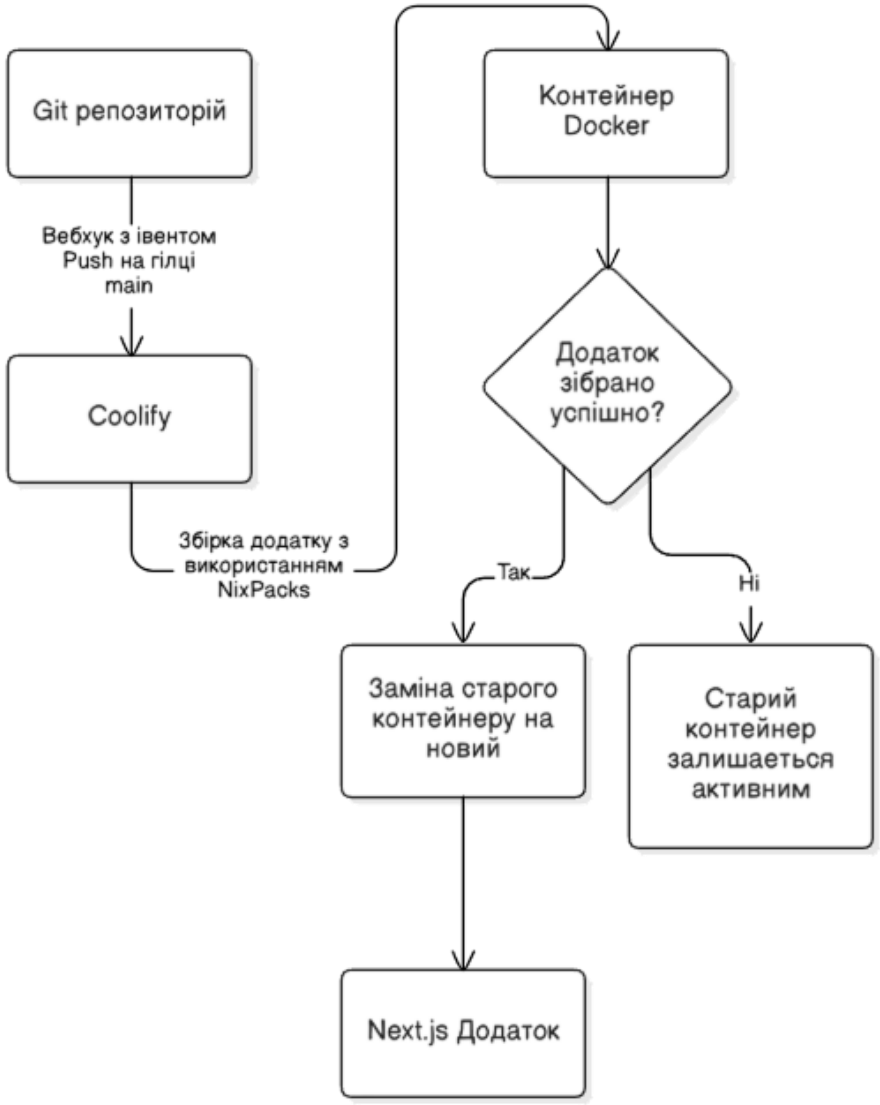


Рис. 3.10 Схема розгортання додатку з використанням Coolify

3.4. Тестування розробленої системи, оцінка ефективності впровадження та перспектив розвитку

Після завершення реалізації основних функціональних модулів веб-платформи було проведено тестування системи з метою перевірки коректності роботи функціоналу, стабільності взаємодії між компонентами застосунку та відповідності реалізованого рішення поставленим задачам. Тестування виступає важливим етапом розробки будь-якої інформаційної системи, оскільки дозволяє виявити помилки, оцінити надійність роботи застосунку та визначити готовність програмного продукту до використання в реальних умовах.

У межах тестування особлива увага приділялася перевірці ключових бізнес-процесів платформи, оскільки саме вони формують основу взаємодії між замовниками та виконавцями. Насамперед було протестовано модуль аутентифікації користувачів. Перевірялася коректність авторизації через Discord і Steam, створення користувацької сесії, автоматичне формування профілю після першого входу та механізми захисту доступу до закритого функціоналу системи. У процесі тестування підтверджено коректну роботу middleware-перевірок і protected procedures, що забезпечують обмеження доступу до захищених операцій.

Окрему увагу було приділено тестуванню модуля замовлень, оскільки він є центральним компонентом усієї платформи. Перевірялася коректність створення, редагування, закриття та скасування замовлень, а також зміна статусів залежно від дій користувачів. У процесі тестування система успішно обробляла перевірки валідації, обмеження доступу та зміну станів замовлення відповідно до бізнес-логіки.

Під час тестування також було перевірено механізм подання заявок виконавцями. Система коректно обмежувала можливість подання заявки на власне замовлення, запобігала дублюванню заявок, блокувала створення нових заявок після вибору виконавця та коректно валідувала введені замовником дані (рис. 3.11). Це підтвердило правильність реалізації серверних перевірок і цілісність бізнес-логіки.

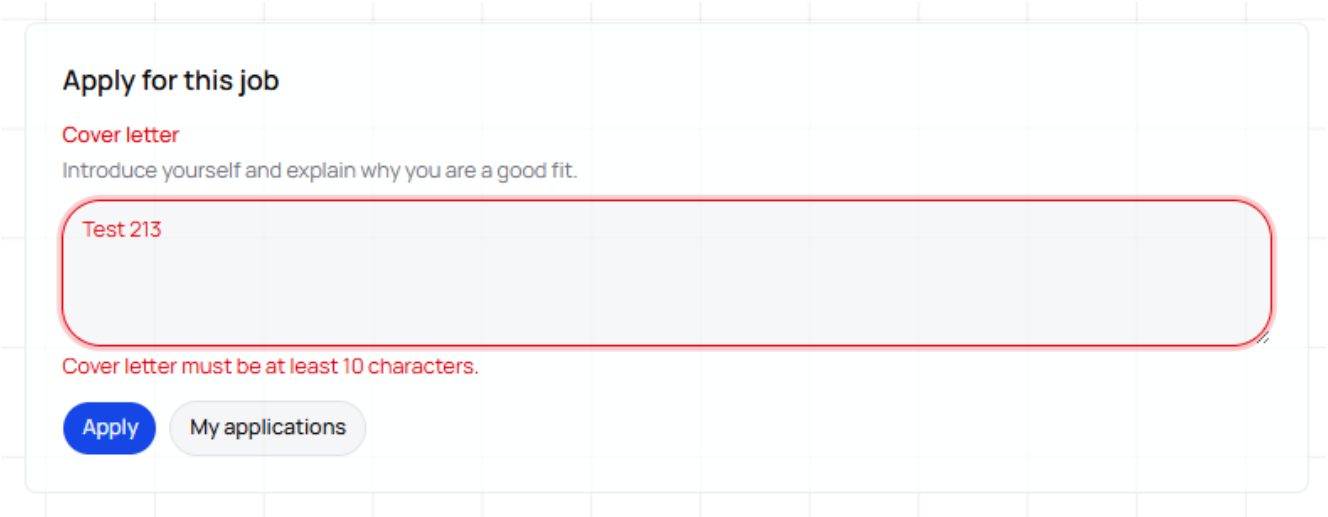


Рис 3.11 End-to-end тестування валідації даних

Окремим етапом стало тестування алгоритму автоматизованого підбору виконавців. Для цього було створено декілька тестових сценаріїв із різними параметрами користувачів, репутацією, категоріями робіт і запропонованими умовами виконання. Результати тестування показали, що система коректно формує рейтинг кандидатів та відображає найбільш релевантних виконавців у верхній частині списку. При цьому алгоритм зберігає гнучкість і не виключає можливість ручного вибору виконавця замовником.

Для перевірки стабільності роботи клієнтського інтерфейсу проводилося тестування адаптивності сторінок і взаємодії компонентів на різних розмірах екрана. Було перевірено коректність роботи навігації, модальних вікон, форм, dropdown-меню та системи повідомлень. Інтерфейс платформи коректно відображався як на десктопних пристроях, так і на мобільних екранах (рис 3.12).

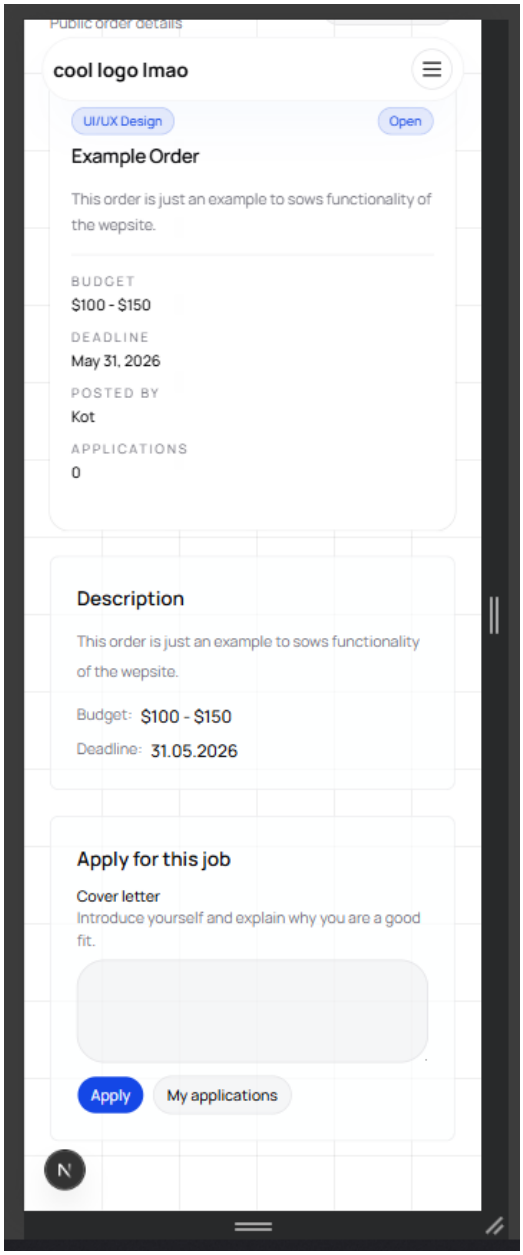


Рис 3.12 Тестування мобільної адаптивності сторінки замовлення

У межах тестування також перевірялась продуктивність системи під час роботи з великою кількістю даних. Аналіз виконання серверних запитів показав, що використання Prisma ORM у поєднанні з PostgreSQL забезпечує стабільну швидкість отримання інформації навіть при значній кількості замовлень і заявок. Використання server-side rendering і server components у Next.js дозволило зменшити навантаження на клієнтську частину застосунку та прискорити первинне завантаження сторінок.

Під час тестування безпеки особлива увага приділялася перевірці серверної валідації даних і механізмів авторизації. Було підтверджено, що критичні операції неможливо виконати без активної сесії користувача, а всі вхідні дані проходять перевірку через Zod-схеми. Додатково перевірялася стійкість системи до спроб несанкціонованого доступу до захищених API-процедур.

Оцінка ефективності впровадження платформи базується на аналізі впливу автоматизації на основні бізнес-процеси взаємодії між користувачами. Використання єдиної централізованої системи дозволяє зменшити кількість ручних дій, необхідних для пошуку виконавців і організації співпраці. Замовники отримують структурований каталог робіт, механізм автоматизованого ранжування кандидатів і систему оцінювання репутації, що значно спрощує процес вибору виконавця. У свою чергу виконавці отримують єдину платформу для пошуку проєктів і формування власного професійного профілю.

Використання репутаційної системи підвищує рівень довіри між учасниками платформи та створює механізм довгострокової оцінки надійності користувачів. Це особливо важливо для Garry’s Mod-спільноти, де значна частина співпраці традиційно відбувається через Discord-сервери або приватні домовленості без централізованої системи контролю.

Реалізований алгоритм автоматизованого підбору виконавців також позитивно впливає на ефективність взаємодії користувачів. Завдяки ранжуванню кандидатів замовники витрачають менше часу на аналіз великої кількості заявок, а система допомагає швидше знаходити виконавців із релевантним досвідом і репутацією.

Важливою перевагою реалізованої системи є її масштабованість. Архітектура застосунку побудована таким чином, щоб забезпечити можливість подальшого розширення функціоналу без критичної перебудови всієї платформи. Модульна структура проєкту дозволяє додавати нові сервіси та механізми взаємодії між користувачами, не порушуючи роботу вже реалізованих компонентів.

Перспективи розвитку платформи пов’язані насамперед із розширенням функціоналу автоматизації та соціальної взаємодії. У майбутньому система може

бути доповнена внутрішнім месенджером, push-сповіщеннями, системою escrow-платежів, розширеною модерацією та механізмами захисту угод між користувачами. Також перспективним напрямом є розвиток рекомендаційної системи з використанням машинного навчання для більш точного підбору виконавців на основі історії співпраці та поведінки користувачів.

Окремий потенціал розвитку має інтеграція з іншими сервісами Garry’s Mod-екосистеми, зокрема маркетплейсами контенту, системами продажу аддонів або сервісами статистики серверів. Це дозволить створити єдину цифрову екосистему для розробників і власників серверів Garry’s Mod.

Перспективним напрямом також є розширення системи репутації шляхом введення спеціалізацій, професійних рівнів, досягнень або верифікації виконавців. Подібний підхід дозволить підвищити якість підбору кандидатів і створити більш структуровану професійну спільноту всередині платформи.

Таким чином, результати тестування підтвердили працездатність і стабільність реалізованої системи, а проведений аналіз ефективності показав доцільність автоматизації процесів взаємодії між замовниками та виконавцями у сфері Garry’s Mod-фрилансу. Реалізована платформа створює основу для централізованої організації співпраці, спрощує пошук виконавців, підвищує рівень довіри між користувачами та має значний потенціал для подальшого розвитку й масштабування.

ВИСНОВКИ

У результаті виконання дипломної роботи було проведено дослідження процесів організації фриланс-взаємодії у середовищі Garry’s Mod та розроблено інформаційну систему управління замовленнями на маркетплейсі фриланс-послуг. У ході роботи було проаналізовано особливості предметної області, визначено основні проблеми існуючих способів взаємодії між замовниками та виконавцями, а також обґрунтовано необхідність створення спеціалізованої веб-платформи для автоматизації відповідних бізнес-процесів.

У межах дослідження встановлено, що значна частина взаємодії всередині Garry’s Mod-спільноти здійснюється через Discord-сервери, форуми та приватні домовленості, що ускладнює пошук виконавців, оцінювання їхньої репутації та контроль виконання робіт. Аналіз існуючих універсальних і спеціалізованих платформ показав, що наявні рішення або не враховують специфіку моддингового середовища, або мають обмежений функціонал для організації повноцінної фриланс-взаємодії. Це підтвердило актуальність теми дослідження та доцільність розробки окремої системи, орієнтованої саме на Garry’s Mod-екосистему.

У процесі виконання роботи було сформовано функціональні та нефункціональні вимоги до веб-платформи, визначено основні сценарії взаємодії користувачів і спроєктовано архітектуру системи. Для реалізації застосунку обрано сучасний full-stack стек технологій, до якого увійшли Next.js, TypeScript, tRPC, Prisma ORM і PostgreSQL. Використання цих технологій дозволило створити масштабовану модульну архітектуру, придатну для подальшого розвитку та розширення функціоналу.

У межах проєктування системи було реалізовано структуру бази даних, моделі користувачів, замовлень, заявок і репутаційної взаємодії. Окрему увагу приділено системі автентифікації та безпеки. Для цього реалізовано підтримку авторизації через Discord і Steam, механізми server-side перевірки доступу, middleware-захист і валідацію вхідних даних.

Одним із ключових результатів роботи стала розробка алгоритму автоматизованого підбору виконавців. Алгоритм базується на інтегральній системі оцінювання та враховує репутацію користувача, кількість завершених робіт, відповідність категорії замовлення, бюджет, терміни виконання та додаткові фактори довіри. Реалізований підхід дозволяє формувати впорядкований список найбільш релевантних кандидатів і значно спрощує процес вибору виконавця для замовника.

У ході практичної реалізації було створено основні функціональні модулі системи, включаючи управління профілями користувачів, створення та редагування замовлень, подання заявок, систему відгуків і механізми взаємодії між учасниками платформи. Також було виконано налаштування середовища розгортання, конфігурацію бази даних і контейнеризацію застосунку через Docker.

Проведене тестування підтвердило працездатність реалізованої системи та коректність роботи основного функціоналу. Перевірка механізмів авторизації, створення замовлень, обробки заявок, роботи репутаційної системи й алгоритму автоматизованого підбору виконавців показала стабільність взаємодії між компонентами платформи та відповідність системи поставленим задачам. Отримані результати свідчать про доцільність використання автоматизованих механізмів управління взаємодією між замовниками та виконавцями у сфері Garry’s Mod-фрилансу.

Практична цінність роботи полягає у створенні повноцінної веб-платформи, яка може бути використана як основа для організації централізованого фриланс-маркетплейсу в Garry’s Mod-спільноті. Реалізована система дозволяє оптимізувати процес пошуку виконавців, підвищити рівень довіри між користувачами та створити структуроване середовище для організації співпраці.

Перспективи подальшого розвитку платформи пов’язані з розширенням функціоналу автоматизації та соціальної взаємодії. У майбутньому система може бути доповнена внутрішнім месенджером, escrow-платежами, розширеними механізмами модерації, push-сповіщеннями, інтеграцією з іншими сервісами

Garry’s Mod-екосистеми та вдосконаленими рекомендаційними алгоритмами на основі машинного навчання.

Таким чином, у межах дипломної роботи було досягнуто поставленої мети – розроблено інформаційну систему управління замовленнями на маркетплейсі фриланс-послуг у середовищі Garry’s Mod, яка забезпечує автоматизацію ключових процесів взаємодії між замовниками та виконавцями та створює основу для подальшого розвитку спеціалізованої цифрової платформи у сфері моддингу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Online Labour Index 2020: new ways to measure the world's remote freelancing market / F. Stephany та ін. *Big data & society*. 2021. Т. 8, № 2. С. 205395172110432. URL: https://doi.org/10.1177/20539517211043240 (дата звернення: 19.05.2026).
2. *Unlock Development Insights with World Bank Blogs*. URL: https://blogs.worldbank.org/content/dam/sites/blogs/img/detail/mgr/capture_36.png (дата звернення: 19.05.2026).
3. Kässi O., Lehdonvirta V. Online labour index: measuring the online gig economy for policy and research. *Technological forecasting and social change*. 2018. Т. 137. С. 241–248. URL: https://doi.org/10.1016/j.techfore.2018.07.056 (дата звернення: 19.05.2026).
4. Good gig, bad gig: autonomy and algorithmic control in the global gig economy / A. J. Wood та ін. *Work, employment and society*. 2018. Т. 33, № 1. С. 56–75. URL: https://doi.org/10.1177/0950017018785616 (дата звернення: 19.05.2026).
5. Freelancing statistics 2026: the complete industry analysis & market trends. *Jobbers*. URL: https://www.jobbers.io/ultimate-freelancing-statistics-for-2025-the-complete-industry-analysis-that-changes-everything/ (дата звернення: 19.05.2026).
6. *Fiverr*. URL: https://www.fiverr.com/ (дата звернення: 19.05.2026).
7. *Upwork*. URL: https://www.upwork.com/work (дата звернення: 19.05.2026).
8. Talent marketplace. *Upwork*. URL: https://www.upwork.com/talent-marketplace (дата звернення: 19.05.2026).
9. Browse products. *GmodStore*. URL: https://www.gmodstore.com/market/browse (дата звернення: 19.05.2026).
10. Facepunch Studios. Garry's mod. Valve, 2006. URL: https://store.steampowered.com/app/4000/Garrys_Mod/ (дата звернення: 19.05.2026).
11. Garry's Mod steam charts. *SteamDB*. URL: https://steamdb.info/app/4000/charts (дата звернення: 19.05.2026).
12. Next.js by Vercel - the React framework. *Next.js*. URL: https://nextjs.org/ (дата звернення: 19.05.2026).
13. tRPC - Move fast and break nothing. End-to-end typesafe APIs made easy. *tRPC*. URL: https://trpc.io (дата звернення: 19.05.2026).
14. Prisma | Serverless Postgres, Type-Safe ORM, and Database Tools. *Prisma*. URL: https://www.prisma.io (дата звернення: 19.05.2026).
15. PostgreSQL. *PostgreSQL*. URL: https://www.postgresql.org/ (дата звернення: 19.05.2026).
16. Better Auth. *Better Auth*. URL: https://better-auth.com/ (date of access: 19.05.2026).
17. GitHub - byo-software/steam-openid-connect-provider: Steam OpenID Connect Identity Provider (IdP). *GitHub*. URL: https://github.com/byo-software/steam-openid-connect-provider (дата звернення: 19.05.2026).

18. Coolify. *Coolify*. URL: https://coolify.io (дата звернення: 19.05.2026).

19. *Next.js*. URL: https://nextjs.org/_next/image?url=https://h8DxKfmAPhn8O0p3.public.blob.vercel-storage.com/docs/dark/nested-file-conventions-component-hierarchy.png&w=1920&q=75 (дата звернення: 19.05.2026).

20. Vercel. Next.js Docs: App Router. *Next.js*. URL: https://nextjs.org/docs/app (дата звернення: 19.05.2026).

21. tRPC. *tRPC*. URL: https://trpc.io/docs/ (дата звернення: 19.05.2026).

22. *Better Stack*. URL: https://imagedelivery.betterstackcdn.com/xZXo0QFi-1_4Zimer-T0XQ/ba219e20-7065-4b1d-814f-fa7ee048c500/orig (дата звернення: 19.05.2026).

23. What is Prisma ORM? (Overview). *Prisma*. URL: https://www.prisma.io/docs/orm (date of access: 19.05.2026).

24. 5.5. Constraints. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/ddl-constraints.html (date of access: 19.05.2026).

25. 3.4. Transactions. *PostgreSQL Documentation*. URL: https://www.postgresql.org/docs/current/tutorial-transactions.html (date of access: 19.05.2026).

26. Introduction. *Better Auth*. URL: https://better-auth.com/docs/introduction (date of access: 19.05.2026).

27. Coolify Docs. *Coolify*. URL: https://coolify.io/docs/ (date of access: 19.05.2026).

28. CI/CD with Git Providers. *Coolify*. URL: https://coolify.io/docs/applications/ci-cd/introduction (date of access: 19.05.2026).

29. Fast, disk space efficient package manager. *pnpm*. URL: https://pnpm.io/ (date of access: 19.05.2026).

30. Env. *T3 Env*. URL: https://env.t3.gg/ (date of access: 19.05.2026).

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

Дипломна робота на тему

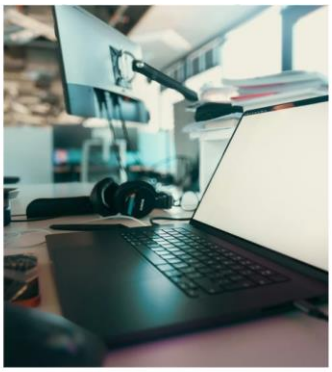
“Інформаційна система управління замовленнями та репутацією виконавців на маркетплейсі фріланс-послуг у середовищі моддингу Garry’s Mod”

Кваліфікаційна робота здобувача вищої освіти спеціальності
126 «Інформаційні системи та технології».

Виконав: ст. гр. ІСД-42 Тищенко Д.В.
Керівник: Данильченко В.М., доктор філософії.
Державний університет інформаційно-комунікаційних технологій.

01

Актуальність теми



- Стрімкий розвиток цифрової економіки та дистанційної зайнятості сприяв активному поширенню фриланс-платформ.
- У середовищі Garry’s Mod взаємодія між замовниками та виконавцями часто відбувається через Discord, форуми або приватні повідомлення.
- Такий підхід ускладнює пошук виконавців, контроль виконання робіт та формування довіри між сторонами.
- Існуючі платформи не враховують специфіку Garry’s Mod-моддингу.
- Виникає потреба у створенні спеціалізованої веб-платформи для автоматизації управління замовленнями та репутацією виконавців.

02

03

Мета, об’єкт, предмет та задачі дослідження

Мета дослідження

Розробка веб-платформи для автоматизації управління замовленнями та підтримки взаємодії міжзамовниками й виконавцями у сфері Garry’s Mod-фрилансу.

Об’єкт дослідження

Процес взаємодії між замовниками та виконавцями у сфері фриланс-послуг у середовищі Garry’s Mod.

Предмет дослідження

Методи та засоби проєктування і реалізації інформаційної системи управління замовленнями та автоматизованого підбору виконавців.

Задачі дослідження

Проаналізувати предметну область, спроєктувати архітектуру системи, розробити алгоритм підбору виконавців та реалізація веб-платформи.

04

Аналіз існуючих платформ

1 fiverr.

універсальна платформа цифрових послуг
велика кількість виконавців
відсутня спеціалізація під Garry’s Mod

2 Upwork

універсальна платформа цифрових послуг
велика кількість виконавців
відсутня спеціалізація під Garry’s Mod

3 gmod store

спеціалізований маркетплейс Garry’s Mod
підтримка Job Market
орієнтація переважно на продаж готових addon-рішень

Висновок

Жодна з існуючих платформ повністю не забезпечує комплексне управління фриланс-замовленнями у Garry’s Mod-середовищі.

05

Постановка задачі та вимоги до системи

Основною задачею роботи стало створення спеціалізованої веб-платформи для централізованого управління фриланс-замовленнями у сфері Garry's Mod-моддингу. Система повинна забезпечувати:

- Створення та редагування замовлень
- Підтримку системи заявок виконавців
- Автоматизований підбір виконавців
- Управління статусами замовлень
- Формування системи репутації користувачів

Основні вимоги до системи:

- Інтуїтивно зрозумілий інтерфейс
- Надійність та захищеність даних користувачів
- Масштабованість та продуктивність системи
- Підтримка різних ролей користувачів
- Зручність взаємодії між замовниками та виконавцями

06

Вибір технологічного стеку

Для реалізації веб-платформи було обрано сучасний технологічний стек, який забезпечує високу швидкість розробки, масштабованість і підтримку інтеграції із зовнішніми сервісами.

Користувацький інтерфейс

Серверна логіка застосунку

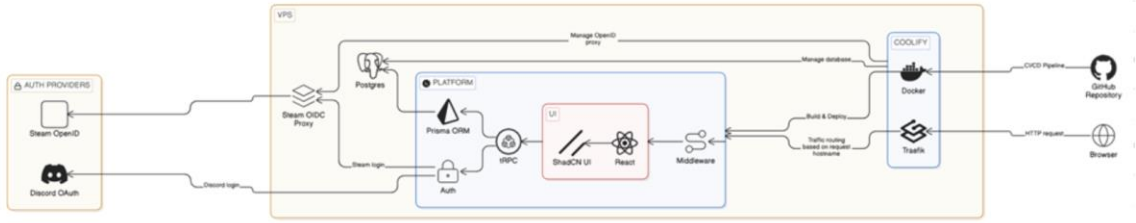
Робота за даними

Середовище розгортання

Зовнішня автентифікація та інтеграції

Steam OpenID Connect Provider

Архітектура веб-платформи



Архітектура системи побудована за модульним принципом і складається з клієнтської частини, серверної логіки, бази даних, системи автентифікації та зовнішніх інтеграцій. Такий підхід забезпечує гнучкість системи, спрощує її масштабування та дозволяє розширювати функціонал у майбутньому.

07

Моделювання функціоналу системи

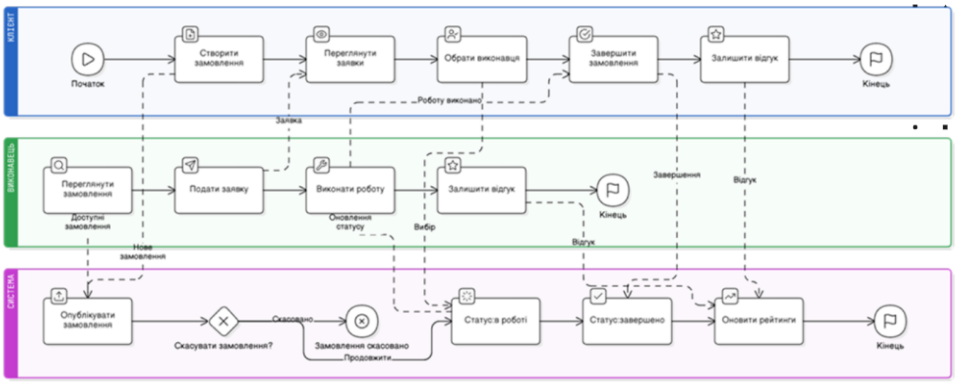
У процесі проєктування було змодельовано основні сценарії роботи системи. Замовник може створити замовлення, вказати категорію, бюджет і технічний опис роботи. Виконавець отримує можливість переглядати релевантні замовлення, подавати заявки та взаємодіяти із замовником.

Система підтримує управління статусами замовлення, формування репутації та збереження історії взаємодії між сторонами.

08

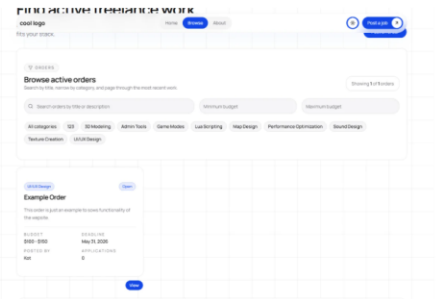
Життєвий цикл замовлення

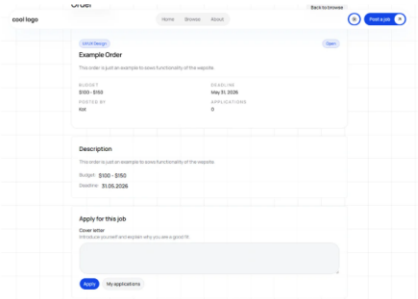
09



Реалізація веб-платформи та інтерфейсу

У межах роботи було реалізовано систему авторизації, модуль профілів користувачів, систему створення замовлень, механізм подання заявок та систему репутації. Інтерфейс платформи побудований із урахуванням сучасних принципів UX/UI-дизайну та орієнтований на швидку й зрозумілу взаємодію між користувачами.





10

11

Авторизація

На скріншоті показано конфігурацію авторизації в додатку, реалізовану за допомогою бібліотеки **better-auth**. Використовується **PostgreSQL** (через Prisma) для зберігання даних користувачів і облікових записів.

Discord OAuth

Налаштовано авторизацію через Discord для входу користувачів в систему.

Steam через OIDC

Steam не підтримує стандартний OAuth, тому використовується додатковий сервіс, який виступає OIDC-провайдером у вигляді проксі між Steam та додатком.

auth.ts — TypeScript

```ts
export const auth = betterAuth({
  database: prismaAdapter(createPrismaClient(), { provider: "postgresql" }),
  account: {
    accountLinking: {
      allowDifferentEmails: true,
    },
  },
  socialProviders: {
    discord: {
      clientId: env.DISCORD_CLIENT_ID,
      clientSecret: env.DISCORD_CLIENT_SECRET,
    },
  },
  plugins: [
    genericOAuth({
      config: [
        {
          providerId: "steam",
          clientId: env.STEAM_CLIENT_ID,
          clientSecret: env.STEAM_CLIENT_SECRET,

          discoveryUrl: new URL(
            "/.well-known/openid-configuration",
            env.STEAM_IDP_URL,
          ).href,
          scopes: ["openid", "profile"],
          disableSignUp: true,

          mapProfileToUser: (profile) => ({
            email: profile.email ?? `${profile.steam_id}@${env.STEAM_IDP_URL}`,
            id: profile.steam_id,
          }),
        },
      ],
    }),
    nextCookies(),
  ],
});
```

12

Алгоритм автоматизованого підбору виконавців

Одним із ключових елементів системи є алгоритм автоматизованого підбору виконавців. Алгоритм аналізує наявну репутацію виконвця, кількість завершених замовлень, відповідність спеціалізації до поточної категорії замовлення, коефіцієнт відповідності бюджету, коефіцієнт терміну виконання та додаткові фактори довіри.

$$R = \omega_1 * P + \omega_2 * C + \omega_3 * S + \omega_4 * B + \omega_5 * D + \omega_6 * A$$

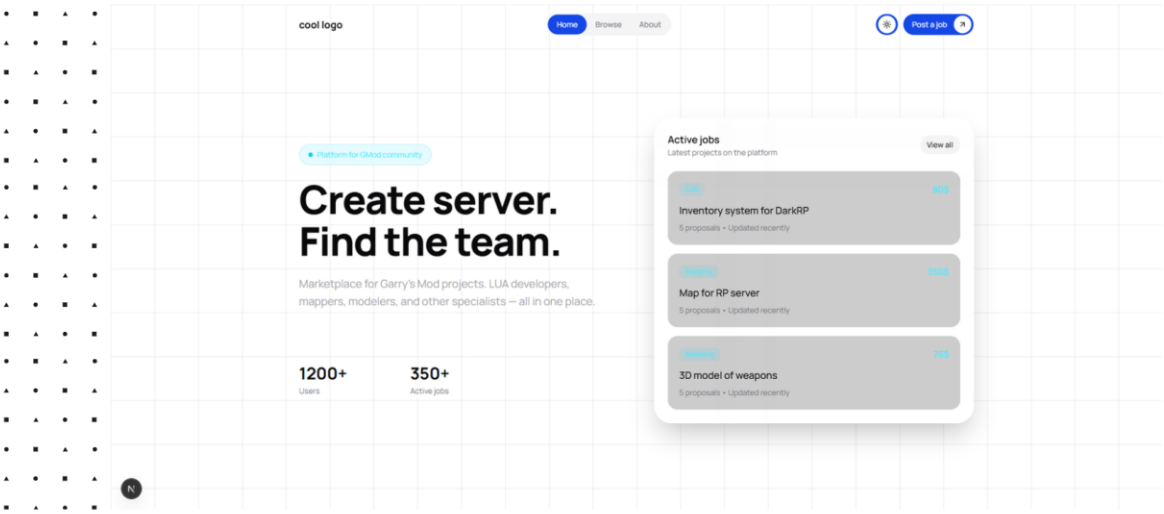
На кожен параметр впливає додатковий кофіцієнт важливості, що дозволяє балансувати значення рейтенгу виконавця як для активних користувачів платформи, так і для нових. На основі цих параметрів система формує перелік рекомендованих виконавців для конкретного замовлення, що дозволяє підвищити швидкість пошуку кандидатів і покращити якість взаємодії між користувачами.

Тестування та результати роботи

13

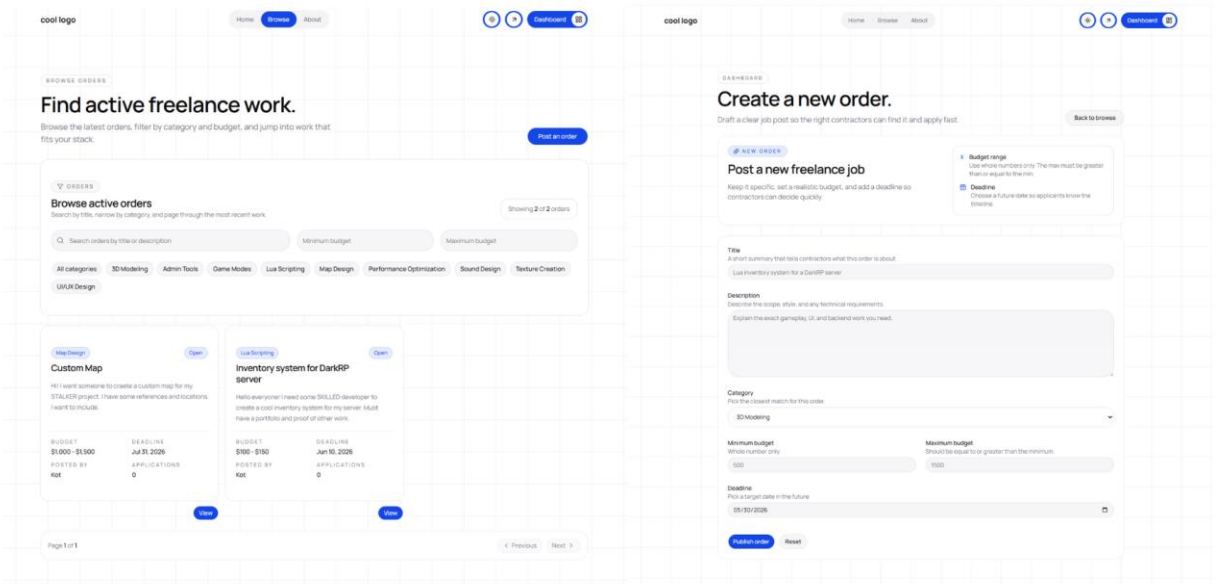
У процесі тестування було перевірено основні функціональні модулі платформи за типовими сценаріями використання. Зокрема, тестувались авторизація через Discord OAuth, створення та редагування замовлень, подання заявок виконавцями, зміна статусів замовлень і система репутації.

авторизація користувачів	Виконувалась успішно у 100% тестових сценаріїв
створення та обробка замовлень	Працювали коректно без втрати даних
механізм репутації	Коректно оновлював рейтинг після завершення замовлення;
середній час	Завантаження сторінок становив менше 2 секунд



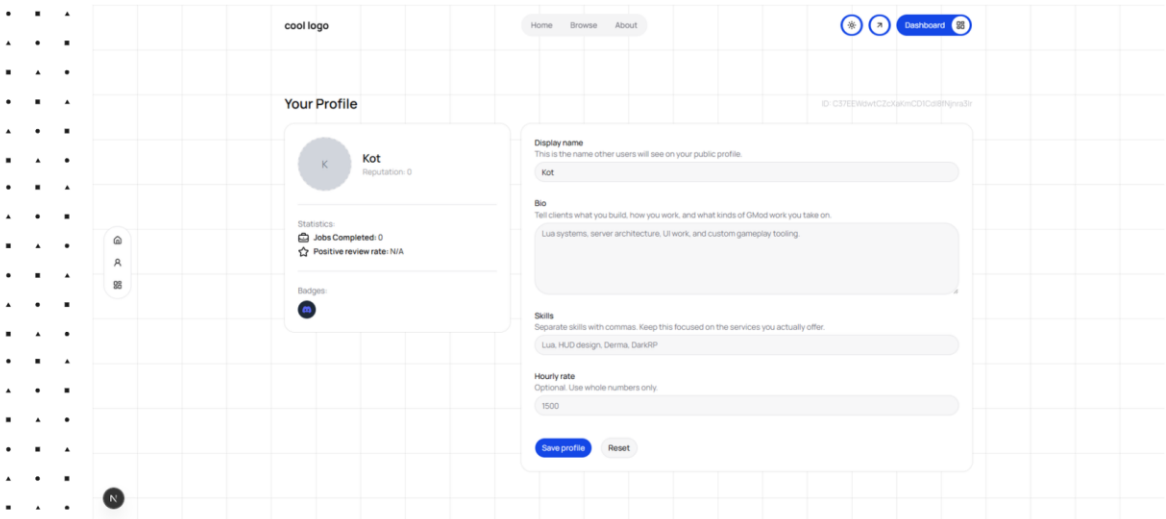
Розроблення платформа

Домашня сторінка



Розроблена платформа

Модуль замовлень



Розроблена платформа

Керування профілем

Апробація

VII Міжнародна науково-технічна конференція «Сучасний стан та перспективи розвитку IoT»
(Київ, ДУІКТ, 20 квітня 2026 р.)

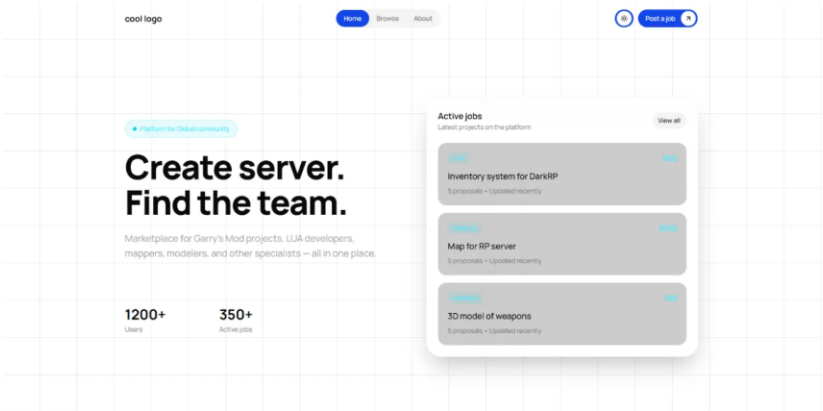
Тези доповіді на тему «МОДЕЛЬ ОЦІНЮВАННЯ РЕПУТАЦІЇ КОРИСТУВАЧІВ У ЦИФРОВИХ СЕРВІСАХ ЯК СКЛАДОВА IoT-ЕКОСИСТЕМ» опублікований у збірнику матеріалів конференції (стор. 279)

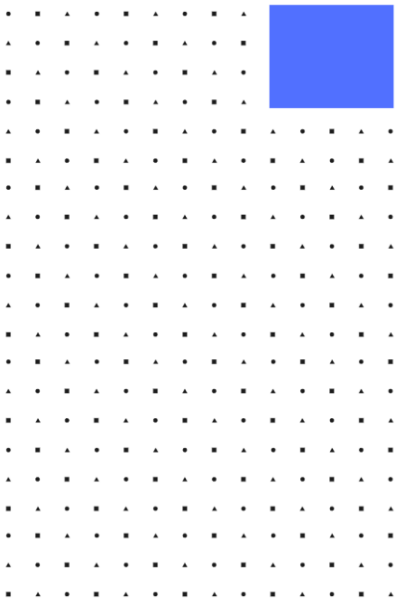
VII Міжнародної наукової конференції «Період трансформаційних процесів в світовій науці: задачі та виклики»
(Полтава, 22 червня 2026 р.)

Тези доповіді на тему «СИСТЕМА РЕПУТАЦІЇ ЯК ЕЛЕМЕНТ ДОВІРИ В МАРКЕТПЛЕЙСАХ ЦИФРОВИХ ПОСЛУГ» опублікований у збірнику матеріалів конференції (стор. 316)

Висновки

У межах дипломної роботи було проведено аналіз предметної області та існуючих платформ, сформовано вимоги до системи та спроєктовано архітектуру веб-платформи. Було реалізовано основні функціональні модулі, розроблено алгоритм автоматизованого підбору виконавців і проведено тестування системи.
Результатом роботи стала спеціалізована інформаційна система, яка може використовуватись для організації фриланс-взаємодії у середовищі моддингу Garry's Mod.





Дякую за увагу!