

**МІНІСТЕРСТВО ФІНАНСІВ УКРАЇНИ
ДЕРЖАВНИЙ ПОДАТКОВИЙ УНІВЕРСИТЕТ**

**ФАКУЛЬТЕТ ФІНАНСІВ ТА ЦИФРОВИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І
СИСТЕМ**

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Розробка соціальної платформи для ІТ-спільнот»

Здобувача вищої освіти першого (бакалаврського) рівня,
IV курсу, групи КБ-22-1, спеціальності 122 «Комп'ютерні науки»,
освітня програма «Інформаційні управляючі системи і технології (за галузями)


(підпис)

Савін Нікіта
Вікторович
(прізвище, ім'я, по батькові)

Науковий керівник


(підпис)

Редич Олександр
Володимирович, канд. екон.
наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Керівник освітньо-
професійної програми


(підпис)

Ніжегородцев Владислав
Олександрович, канд. пед.
наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Допущено до захисту за рішенням кафедри комп'ютерних та інформаційних
технологій і систем протокол від «12» травня 2026 №20

Завідувач кафедри


(підпис)

Філоненко Михайло
Миколайович, канд. фіз.-мат.
наук, доцент
(прізвище, ім'я, по батькові, науковий ступінь,
вчене звання)

Ірпінь – 2026

Реєстрація № 56

28.05.2026
(дата)


(підпис відповідальної особи)

Результати перевірки

до захисту
(до захисту, на доопрацювання)

12.05.26
(дата)


(підпис)

Результат захисту

5 (відмінно)
(оцінка)

09.06.2026
(дата)


(підпис)

Голова комісії:


(підпис)

О.С. Дюбанов

Члени комісії:


(підпис)

М.М. Філоненко


(підпис)

В.О. Ніжегородцев


(підпис)

А.А. Омельчук

БЛАНК ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Об'єкти контролю	Діапазон оцінки в балах	Фактична кількість балів	Примітки	Дата, П.І.Б.: керівника, голови комісії членів комісії, секретаря, (підписи)
ОЦІНЮВАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ КОМІСІЄЮ ПО ЗАХИСТУ				
1. Зміст випускної роботи (якість теоретичного матеріалу, його зв'язок з практичним, вміння представляти матеріал в логічній послідовності, наявність висновків та пропозицій)	від 0 до 30	28		09.06.2026 Редич О.В. Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерхов Л.П.
2. Дотримання вимог щодо оформлення	від 0 до 10	9		
3. Вчасне виконання роботи на всіх етапах у встановлені кафедрою терміни	від 0 до 5	4		
4. Рецензент	від 0 до 5	5		
РАЗОМ БАЛІВ	від 0 до 50	46		

Оцінювання кваліфікаційної роботи науковим керівником – 45 балів.

ПЕРЕВЕДЕННЯ ОЦІНКИ ЗА 100-БАЛЬНОЮ ШКАЛОЮ ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою	Дата	П.І.Б.: голови, членів комісії, секретаря, (підписи)
91	A	5 (відмінно)	09.06.2026	Дюбанов О.С. Філоненко М.М. Ніжегородцев В.О. Омельчук А.А. Суховерхов Л.П.

ПРАВИЛА ПЕРЕВЕДЕННЯ ОЦІНОК ДО НАЦІОНАЛЬНОЇ ШКАЛИ ТА ШКАЛИ ЄКТС

Оцінка за 100-бальною шкалою	Оцінка за шкалою ЄКТС	Оцінка за національною шкалою
90-100	A	5 (відмінно)
80-89	B	4 (добре)
70-79	C	
60-69	D	
50-59	E	3 (задовільно)
35-49	FX	2 (незадовільно)

Факультет фінансів та цифрових технологій
Кафедра комп'ютерних та інформаційних технологій і систем
Галузь знань: 12 «Інформаційні технології»
Спеціальність: 122 «Комп'ютерні науки»
Освітня програма: «Інформаційні управляючі системи і технології (за галузями)»

Затверджую
Завідувач кафедри  Філоненко М.М.
«01» грудня 2025

Завдання на кваліфікаційну роботу
здобувачу вищої освіти

Савіну Никіті Вікторовичу

(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи «Розробка соціальної платформи для ІТ-спільнот» затверджена наказом ректора від «28» листопада 2025 № 1526
2. Строк здачі закінченої роботи «08» травня 2026
3. Мета роботи: розробка веб-застосунку та веб-сервісу для соціальної платформи для ІТ-спільнот.
Об'єкт дослідження: методи та технології розробки веб-додатку, веб-сервісу для розробки централізованої соціальної платформи для ІТ-спільнот.
Предмет дослідження: інструментарій розробки веб-застосунків та веб-сервісів з використанням Django, Django-ninja, Node JS, Next JS, PostgreSQL.
Орієнтовна база дослідження: соціальні мережі, веб форуми, наукові праці.
4. Вхідні дані до роботи: відкриті дані та документація по Django, Django-ninja, Next JS, Node JS, Python.
5. Перелік графічного матеріалу : рисунки інтерфейсів додатку, структурні схеми, алгоритми, лістинги програм.
6. Зміст кваліфікаційної роботи (перелік питань за кожним розділом)
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ СОЦІАЛЬНИХ МЕРЕЖ ТА ФОРУМІВ.
1.1 Огляд соціальних платформ
1.2. Порівняльний аналіз функціональних можливостей соціальних мереж
1.3 Проблеми використання поширених варіантів веб платформ для централізованої мережі для it-спільнот
1.4 Обґрунтування необхідності розробки власної системи
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ.
2.1. Вибір технологій та інструментів для розробки
2.2 Архітектура системи
2.3 Проєктування бази даних та основного функціоналу
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ ТА RESTAPI-SERVICES
3.1 Налаштування середовища розробки та налаштування проєкту
3.2 Адміністрування серверу
3.3 Реалізація веб-сервісу restapi
3.4 Реалізація фронтенд частини
3.5 Тестування системи

7. Календарний план виконання роботи

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	
		За планом	Фактично
1	2	3	4
1	Вибір теми, підбір літературних джерел, формування структури кваліфікаційної роботи	До 30.10.2025	<u>30.10.2025</u>
2	I розділ (розкрити сутність і стан дослідження наукової проблематики роботи, після чого навести її загальну характеристику; огляд літератури за темою і вибір конкретних напрямів досліджень)	26.12.2025	<u>26.12.2025</u>
3	II розділ (аналіз експериментальної частини про методики досліджень)	18.02.2026	<u>18.02.2026</u>
4	III розділ (проведені теоретичні і (або) експериментальні дослідження та проектні рішення)	16.04.2026	<u>16.04.2026</u>
5	Висновки	01.05.2026	<u>01.05.2026</u>
6	Доопрацювання роботи з урахуванням зауважень наукового керівника	07.05.2026	<u>04.05.2026</u>
7	Отримання відгуку керівника	29.05.2026	<u>29.05.2026</u>
8	Отримання зовнішньої рецензії	01.06.2026	<u>01.06.2026</u>

8. Дата видачі завдання «01» грудня 2025

9. Керівник кваліфікаційної роботи


(підпис)

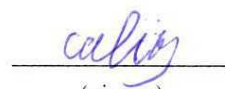
Редич О.В.
(прізвище, ініціали)

10. Керівник освітньо-професійної програми


(підпис)

Ніжегородцев В.О.
(прізвище, ініціали)

11. Завдання до виконання отримав здобувач вищої освіти


(підпис)

Савін Н.В.
(прізвище, ініціали)

АНОТАЦІЯ

Розробка соціальної платформи для ІТ-спільнот

Мета роботи полягає у розробці соціальної платформи для ІТ-спільнот, яка забезпечує професійну взаємодію користувачів, створення тематичних спільнот, публікацію дописів, обмін досвідом та підтримку командної роботи в межах єдиного цифрового середовища.

Завданнями роботи є:

- проведення аналізу існуючих соціальних платформ та професійних сервісів для ІТ-фахівців;
- визначення основних проблем сучасних платформ для організації ІТ спільнот;
- обґрунтування вибору технологічного стеку для реалізації проєкту;
- проєктування архітектури системи та структури бази даних реалізація серверної частини вебзастосунку на базі Django та Django Ninja;
- реалізація клієнтської частини із використанням Next.js, React та TypeScript;
- реалізація механізмів автентифікації, авторизації та керування користувачами;
- тестування функціональності та оцінка працездатності розробленої системи.

Предметом дослідження є методи, технології та інструменти розробки соціальних вебплатформ із використанням сучасних full-stack технологій.

Об'єктом дослідження є процес розроблення вебзастосунків та вебсервісів для організації професійної взаємодії учасників ІТ-спільнот.

Базою дослідження виступає розроблена соціальна платформа «Stack-Fate».

За результатами дослідження спроектовано та реалізовано соціальну платформу для ІТ-спільнот. Для реалізації проєкту використано мову програмування Python, фреймворки Django та Django Ninja, систему керування базами даних PostgreSQL, а також Next.js, React, TypeScript і Tailwind CSS для створення клієнтської частини.

Одержані результати можуть бути використані для створення професійних онлайн-спільнот, організації командної роботи, професійного розвитку користувачів та подальшого розширення функціональних можливостей платформи шляхом впровадження систем рекомендацій, інструментів модерації та сервісів пошуку спеціалістів.

Кваліфікаційна робота містить 61 сторінку, 11 таблиць, 14 рисунків, список використаних джерел із 48 найменувань, __ додатка .

Ключові слова: соціальна платформа, ІТ-спільнота, вебзастосунок, Django, Django Ninja, Next.js, PostgreSQL, REST API, TypeScript, full-stack розробка.

Рік виконання кваліфікаційної роботи: 2026.

Рік захисту роботи: 2026.

ABSTRACT

Development of a Social Platform for IT Communities

The aim of this work is to develop a social platform for IT communities that facilitates professional interaction among users, the creation of specialized communities, the publication of posts, the exchange of expertise, and the promotion of teamwork within a unified digital environment.

The tasks of the work are as follows:

- to analyze existing social platforms and professional services for IT specialists;
- to identify the main challenges of current platforms for organizing IT communities;
- to justify the choice of technology stack for project implementation;- designing the system architecture and database structure;
- implementing the server-side of the web application using Django and Django Ninja;
- implementing the client-side using Next.js, React, and TypeScript;- implementing authentication, authorization, and user management mechanisms;
- testing functionality and evaluating the performance of the developed system.

The subject of the research technologies, and tools used to develop social web platforms using modern full-stack technologies.

The object of the research is the process of developing web applications and web services to facilitate professional interaction among members of IT communities.

The study is based on the “Stack-Fate” social platform.

According to the research results, a social platform for IT communities was designed and implemented. The project was implemented using the Python programming language, the Django and Django Ninja frameworks, the PostgreSQL database management system, as well as Next.js, React, TypeScript, and Tailwind CSS for creating the client-side.

The obtained results can be used to create professional online communities, organize teamwork, facilitate user professional development, and further expand the platform’s functionality by implementing recommendation systems, moderation tools, and specialist search services.

The qualification work consists of 61 pages, 11 tables, 14 figures, a list of used sources of 48 items, __ appendices.

Keywords: social platform, IT community, web application, Django, Django Ninja, Next.js, PostgreSQL, REST API, TypeScript, full-stack development.

Year of completion of the qualification work: 2026

Year of defense: 2026

Зміст

ВСТУП	3
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ СОЦІАЛЬНИХ МЕРЕЖ ТА ФОРУМІВ.....	6
1.1 Огляд сучасних соціальних платформ	6
1.2 Порівняльний аналіз функціональних можливостей соціальних мереж	8
1.3 Проблеми використання поширених варіантів веб-платформ для централізованої мережі ІТ-спільнот.....	11
1.4 Обґрунтування необхідності розробки власної системи.....	13
РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ	16
2.1 Вибір технологій та інструментів для розробки	16
2.2 Архітектура системи	17
2.3 Проєктування бази даних та основного функціоналу	25
РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ ТА RESTAPI- СЕРВІСУ	32
3.1 Налаштування середовища розробки та налаштування проєкту	32
3.2 Адміністрування серверу	37
3.3 Реалізація веб-сервісу REST API	39
3.4 Реалізація frontend частини	44
3.5 Тестування системи.....	50
ВИСНОВКИ.....	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	58

ВСТУП

У сучасних умовах стрімкого розвитку інформаційних технологій цифрові платформи стали невід’ємною частиною професійної діяльності, навчання та комунікації. Особливо важливу роль вони відіграють у сфері інформаційних технологій, де ефективна взаємодія між спеціалістами, студентами, менторами та роботодавцями безпосередньо впливає на швидкість професійного розвитку, створення інноваційних рішень та реалізацію спільних проєктів. Розвиток ІТ-галузі в Україні супроводжується постійним збільшенням кількості спеціалістів, навчальних програм, стартапів та професійних спільнот, що створює потребу у сучасних цифрових системах для організації взаємодії між учасниками галузі.

Незважаючи на значну кількість професійних платформ і сервісів, користувачі змушені одночасно працювати з декількома окремими системами для вирішення різних завдань. LinkedIn використовується для професійного нетворкінгу та пошуку роботи, GitHub — для спільної роботи над програмним кодом, Discord і Telegram — для комунікації, Jira та Trello — для керування проєктами. Така фрагментація цифрового середовища ускладнює професійну взаємодію, створює додаткове навантаження на користувачів і знижує ефективність командної роботи. Особливо актуальною ця проблема є для студентів, молодих спеціалістів та роботодавців, які потребують єдиного середовища для професійного розвитку, демонстрації навичок, пошуку команди, реалізації проєктів і налагодження професійних зв’язків.

Актуальність теми кваліфікаційної роботи обумовлена необхідністю створення сучасної централізованої соціальної платформи для ІТ-спільнот, яка б об’єднувала функціональність професійної соціальної мережі, системи керування проєктами, сервісу менторської підтримки та інструментів професійного розвитку користувачів. Використання сучасних веб-технологій дозволяє створити масштабовану та гнучку систему, здатну забезпечити ефективну взаємодію між учасниками професійного середовища.

Об'єктом дослідження: є процес розробки веб-застосунків та веб-сервісів для організації професійної взаємодії учасників ІТ-спільнот.

Предметом дослідження: є методи, технології та інструменти розробки соціальної платформи з використанням Django, Django Ninja, Next.js, PostgreSQL та REST API архітектури.

Метою кваліфікаційної роботи: є розробка сучасної соціальної платформи для ІТ-спільнот, яка забезпечує професійну взаємодію користувачів, створення спільнот, публікацію дописів, керування профілями та підтримку командної роботи із використанням сучасних веб-технологій.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

- провести аналіз існуючих соціальних платформ та професійних сервісів;
- визначити основні проблеми сучасних платформ для ІТ-спільнот;
- обґрунтувати необхідність створення власної системи;
- здійснити вибір технологій та інструментів для реалізації проєкту;
- спроектувати архітектуру системи та структуру бази даних;
- реалізувати REST API сервіс;
- реалізувати frontend частину веб-застосунку;
- забезпечити систему авторизації та керування користувачами;
- провести тестування реалізованої системи.

Практичне значення отриманих результатів полягає у створенні функціональної full-stack платформи для ІТ-спільнот, яка може бути використана для професійної взаємодії користувачів, створення та розвитку спільнот, поширення дописів та організації цифрового професійного середовища. Розроблена система має перспективи подальшого розвитку у напрямку реалізації систем рекомендацій, інструментів модерації, механізмів

взаємодії роботодавців із кандидатами та підтримки командної роботи над проєктами.

Структура кваліфікаційної роботи складається зі вступу, трьох розділів, висновків, списку використаних джерел та додатків. У першому розділі проведено аналіз існуючих соціальних платформ та визначено основні проблеми сучасних сервісів для ІТ-спільнот. У другому розділі описано процес проєктування системи, вибір технологій та архітектуру платформи. Третій розділ присвячений реалізації веб-застосунку, REST API сервісу та тестуванню системи.

Апробація результатів роботи відбулась публікацією тез доповідей у збірниках науково-практичних конференцій: «Інноваційні цифрові технології: від алгоритмів до безпечних систем» Ірпінь, ДПУ (2026) та «Математичні методи та цифрові технології в економіці, сучасні проблеми математики» Ірпінь, ДПУ, 2026 [9, 10]. Є публікація у фаховому виданні [48];

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ СОЦІАЛЬНИХ МЕРЕЖ ТА ФОРУМІВ

1.1 Огляд сучасних соціальних платформ

У процесі цифрової трансформації суспільства інформаційно-комунікаційні технології стали невід'ємною складовою професійної діяльності, освіти та соціальної взаємодії. Активна діджиталізація різних сфер життя сприяла появі великої кількості цифрових сервісів і платформ, які забезпечують обмін інформацією, комунікацію та спільну роботу користувачів. Одним із результатів цього процесу стало поширення соціальних платформ, що виступають інструментом для формування професійних і тематичних спільнот, обміну знаннями та налагодження взаємодії між учасниками цифрового середовища [1].

Стрімкий розвиток інформаційних технологій сприяв появі великої кількості платформ, орієнтованих на різні категорії користувачів. Частина з них призначена для широкого кола осіб і забезпечує загальну соціальну взаємодію, тоді як інші орієнтовані на професійне спілкування або спільну роботу над проєктами. Соціальна платформа являє собою програмну систему, яка забезпечує створення профілів користувачів, публікацію контенту, обмін повідомленнями та організацію спільної діяльності у віртуальному просторі.

Однією з найпоширеніших соціальних платформ є Facebook, який надає можливість створення особистих профілів, тематичних груп, сторінок організацій та публікації мультимедійного контенту [2]. Основною перевагою платформи є велика кількість користувачів та розвинуті механізми поширення інформації. Для підтримки комунікації використовуються групові обговорення, особисті повідомлення, коментарі та реакції на публікації. Завдяки цьому користувачі можуть швидко взаємодіяти між собою та формувати тематичні спільноти за інтересами.

Для професійного спілкування значного поширення набула платформа LinkedIn [3]. На відміну від традиційних соціальних мереж, її функціональність зосереджена на професійних контактах, розвитку кар'єри та пошуку роботи. Користувачі можуть створювати детальні професійні профілі, демонструвати власні навички та досвід роботи, брати участь у професійних групах і взаємодіяти з представниками компаній. Такий підхід сприяє формуванню якісних професійних зв'язків та розвитку галузевих спільнот.

Особливе місце серед сучасних платформ займає Discord [4]. Спочатку сервіс створювався для спільнот геймерів, однак згодом став універсальним інструментом для організації онлайн-комунікації. Платформа підтримує текстові та голосові канали, відеозв'язок, систему ролей і розмежування прав доступу. Завдяки структурі серверів користувачі можуть створювати окремі тематичні простори для обговорень, що значно покращує організацію комунікації у великих спільнотах.

Ще одним прикладом сучасної платформи є Reddit [5], функціонування якого побудовано навколо тематичних спільнот (subreddits). Користувачі мають можливість створювати публікації, брати участь у дискусіях та оцінювати контент інших учасників. Механізм голосування дозволяє автоматично просувати найбільш корисні та актуальні матеріали, що сприяє самоорганізації спільнот та підвищенню якості інформаційного наповнення.

Для IT-фахівців також важливу роль відіграють спеціалізовані ресурси, зокрема GitHub та Stack Overflow [6, 7]. Хоча вони не є класичними соціальними мережами, їх функціональність забезпечує активну взаємодію користувачів через спільну розробку програмного забезпечення, обмін знаннями та технічними консультаціями. На GitHub комунікація здійснюється через репозиторії, обговорення та систему керування завданнями, тоді як Stack Overflow дозволяє отримувати допомогу шляхом створення тематичних запитань і відповідей.

Аналіз сучасних соціальних платформ показує, що ефективність комунікації значною мірою залежить від наявності спеціалізованих механізмів взаємодії. До таких механізмів належать тематичні групи, канали спілкування, система рекомендацій контенту, приватні повідомлення, засоби модерації та інструменти пошуку однодумців. Саме ці функції забезпечують швидкий обмін інформацією, підтримують активність користувачів і сприяють формуванню сталих онлайн-спільнот.

Разом із тим більшість популярних платформ орієнтовані або на масову аудиторію, або на виконання окремих спеціалізованих завдань. У результаті користувачам ІТ-сфери часто доводиться використовувати декілька різних сервісів одночасно: LinkedIn для професійних контактів, Discord для комунікації в спільнотах, GitHub для спільної розробки проєктів та інші ресурси для обміну знаннями. Така фрагментація цифрового середовища ускладнює професійну взаємодію та створює додаткове навантаження на користувачів. Саме тому актуальним є створення єдиної соціальної платформи для ІТ-спільнот, яка поєднуватиме можливості професійного нетворкінгу, управління проєктами, обміну знаннями та розвитку професійних навичок користувачів.

1.2 Порівняльний аналіз функціональних можливостей соціальних мереж

Для визначення основних переваг та недоліків сучасних соціальних платформ доцільно провести порівняльний аналіз їх функціональних можливостей. Аналіз дозволяє визначити, які саме функції є найбільш важливими для ІТ-спільнот та які проблеми залишаються невирішеними у сучасних системах.

LinkedIn забезпечує високий рівень професійного нетворкінгу. Платформа підтримує створення професійного профілю, публікацію резюме, пошук вакансій та встановлення професійних контактів. Важливою перевагою є

підтримка рекомендацій та підтвердження навичок. Однак система недостатньо орієнтована на командну взаємодію та створення професійних технічних спільнот.

Stack Overflow спеціалізується на обміні технічними знаннями. Основною функцією платформи є система запитань та відповідей. Перевагою сервісу є ефективна система репутації, що дозволяє формувати рейтинг користувачів відповідно до якості їх відповідей. Проте платформа не підтримує повноцінних механізмів соціальної взаємодії та організації командної роботи.

Discord забезпечує високий рівень комунікації між користувачами. Платформа підтримує голосові канали, текстові чати та інтеграцію з різними сервісами. Це робить її популярною серед ІТ-команд та навчальних спільнот. Разом із тим Discord не має повноцінної системи керування професійними профілями або структурованого представлення досягнень користувачів.

GitHub є важливим інструментом для collaborative development. Платформа забезпечує підтримку системи контролю версій Git, pull request механізмів та open-source взаємодії. Однак GitHub орієнтований переважно на програмний код та не забезпечує широкий набір соціальних функцій.

Telegram та Facebook Groups забезпечують швидку комунікацію між учасниками спільнот. Водночас дані сервіси не підтримують професійного профілювання, механізмів оцінювання навичок та централізованої системи розвитку користувачів.

Для ІТ-спільнот важливими є такі функціональні можливості:

- професійний профіль користувача;
- система публікацій;
- командна взаємодія;
- система спільнот;
- управління проєктами;

- менторська підтримка;
- система рекомендацій;
- механізми пошуку спеціалістів;
- підтримка REST API та інтеграцій.

Також важливим фактором є підтримка адаптивного інтерфейсу та мобільних пристроїв. Більшість сучасних платформ реалізують responsive дизайн, що дозволяє забезпечити зручну взаємодію користувачів із системою незалежно від типу пристрою.

У результаті проведеного аналізу можна зробити висновок, що сучасні соціальні платформи забезпечують окремі функціональні можливості професійної взаємодії, однак не формують єдиного централізованого середовища для розвитку IT-спільнот.

Таблиця 1.1

Порівняльний аналіз сучасних соціальних платформ

Платформа	Профіль	Команди	Спільноти	Проекти
LinkedIn	Так	Частково	Так	Ні
Stack Overflow	Частково	Ні	Частково	Ні
Discord	Частково	Так	Так	Частково
GitHub	Так	Так	Частково	Так
Reddit	Ні	Ні	Так	Ні
Telegram	Ні	Частково	Так	Ні
Stack-Fate	Так	Так	Так	Так

Аналіз показав, що жодна з розглянутих платформ не поєднує механізми професійного профілювання, створення спільнот, командної взаємодії та

управління проєктами в межах єдиного середовища. Саме це стало підставою для розробки власної соціальної платформи Stack-Fate.

1.3 Проблеми використання поширених варіантів веб-платформ для централізованої мережі IT-спільнот

Незважаючи на стрімкий розвиток сучасних соціальних платформ та цифрових сервісів, більшість із них не забезпечують комплексного підходу до організації професійної взаємодії IT-спільнот. Кожна платформа орієнтована переважно на виконання окремих задач, через що користувачі змушені використовувати велику кількість різних сервісів одночасно. Така фрагментація цифрового середовища є однією з головних проблем сучасної професійної комунікації у сфері інформаційних технологій.

Для повноцінної професійної діяльності IT-спеціалісти зазвичай використовують декілька різних платформ. LinkedIn застосовується для професійного нетворкінгу та пошуку вакансій, GitHub — для роботи з програмним кодом, Discord або Telegram — для комунікації, Jira чи Trello — для керування задачами та проєктами, а Stack Overflow — для отримання технічної допомоги. Постійне перемикання між сервісами створює додаткове когнітивне навантаження, ускладнює робочі процеси та негативно впливає на продуктивність користувачів.

Однією з важливих проблем сучасних платформ є відсутність єдиного централізованого професійного профілю користувача. Інформація про навички, досягнення, професійний досвід, участь у проєктах та технічні компетенції розподілена між різними сервісами. Це ускладнює оцінювання рівня спеціаліста, пошук кандидатів для командної роботи та формування професійного портфоліо.

Суттєвою проблемою є також інформаційне перевантаження користувачів. Велика кількість повідомлень, сповіщень, професійного контенту та каналів комунікації призводить до появи інформаційного шуму. У результаті користувачі витрачають значну кількість часу на обробку інформації замість ефективної професійної діяльності.

Для студентів та молодих спеціалістів особливо актуальною є проблема професійної інтеграції у сферу інформаційних технологій. Більшість сучасних платформ не підтримують повноцінні механізми наставництва, перевірки прогресу або системи професійного розвитку. У результаті нові користувачі змушені самостійно шукати джерела знань, професійні контакти та можливості для розвитку.

Ще однією важливою проблемою є складність формування команд для спільної роботи над проєктами. Сучасні платформи не забезпечують ефективних механізмів пошуку спеціалістів відповідно до їх навичок, досвіду або професійних інтересів. Це особливо актуально для студентських команд, стартапів та open-source проєктів.

Проблемою сучасних веб-платформ є також недостатня інтеграція між сервісами. Навіть за наявності API та зовнішніх інтеграцій користувачі все одно змушені дублювати інформацію між різними системами. Це створює додаткові складнощі при підтримці актуальності даних та організації командної взаємодії.

У сучасних умовах важливого значення набуває проблема кібербезпеки та довіри між користувачами цифрових платформ. Відсутність ефективної системи верифікації користувачів створює ризики появи фейкових акаунтів, ботів та поширення недостовірної інформації. Для професійних IT-спільнот питання безпеки є особливо важливим, оскільки платформи часто використовуються для командної роботи над реальними проєктами та обміну професійною інформацією.

Ще однією проблемою є обмежені можливості модерації контенту та спільнот. На багатьох сучасних платформах процес модерації або недостатньо ефективний, або надмірно централізований. Це може негативно впливати на якість професійного контенту та рівень комунікації між учасниками спільноти.

Таким чином, сучасні веб-платформи мають ряд суттєвих обмежень, які ускладнюють створення ефективного централізованого середовища для професійної взаємодії IT-спільнот. Це створює необхідність розробки нової системи, яка б забезпечувала інтеграцію основних функцій професійного розвитку, комунікації та командної роботи у межах єдиного цифрового середовища.

1.4 Обґрунтування необхідності розробки власної системи

Створення централізованої соціальної платформи для IT-спільнот є актуальним напрямом розвитку сучасних цифрових сервісів. Аналіз існуючих платформ показав, що більшість сучасних систем не забезпечують комплексного підходу до організації професійної взаємодії, командної роботи та професійного розвитку користувачів.

Розробка власної системи дозволяє інтегрувати основні функціональні можливості сучасних соціальних платформ у межах єдиного цифрового середовища. Користувачі отримують можливість створення професійного профілю, публікації дописів, організації спільнот, пошуку команд та участі у професійній взаємодії без необхідності використання великої кількості сторонніх сервісів.

Однією з головних переваг централізованої платформи є створення єдиного професійного профілю користувача. Система дозволяє зберігати інформацію про навички, досвід, професійні досягнення, участь у проєктах та

активність користувача у межах однієї платформи. Це значно спрощує процес професійної взаємодії та формування цифрового портфоліо.

Власна система також дозволяє реалізувати механізми формування та розвитку професійних спільнот. Користувачі можуть створювати тематичні communities, організовувати обговорення, ділитися професійним контентом та взаємодіяти між собою у межах спільного цифрового середовища.

Особливого значення набуває можливість реалізації системи командної взаємодії. Платформа може підтримувати створення проєктів, пошук учасників команд, розподіл ролей та організацію спільної роботи. Це особливо актуально для студентських проєктів, стартапів та open-source спільнот.

Важливою перевагою власної системи є можливість реалізації role-based моделі доступу. **У системі можуть існувати різні типи користувачів:**

- студенти;
- ментори;
- роботодавці;
- адміністратори спільнот;
- адміністратори платформи.

Використання role-based архітектури дозволяє забезпечити гнучке керування доступом до функціональних можливостей системи та реалізувати більш безпечну модель взаємодії.

Ще однією перевагою є можливість створення централізованої системи модерації контенту. Це дозволяє підтримувати якість професійного середовища, контролювати контент та забезпечувати дотримання правил спільноти.

Для сучасного IT-середовища важливим є також використання сучасних веб-технологій та масштабованої архітектури. Реалізація платформи на основі Django Ninja, Next.js, PostgreSQL та Docker дозволяє створити сучасну full-stack

систему із підтримкою REST API, контейнеризації та незалежного розгортання frontend і backend компонентів.

Використання REST API архітектури забезпечує можливість інтеграції з мобільними застосунками, сторонніми сервісами та зовнішніми платформами. Це створює основу для подальшого розвитку цифрової екосистеми та масштабування системи.

Також важливою перевагою є можливість реалізації системи рекомендацій, професійного пошуку та аналітики користувацької активності. У майбутньому це дозволить реалізувати механізми підбору команд, рекомендацій контенту та взаємодії між роботодавцями та потенційними кандидатами.

Таким чином, розробка власної централізованої соціальної платформи для ІТ-спільнот дозволяє вирішити ключові проблеми сучасних веб-платформ та створити єдине цифрове середовище для професійної взаємодії, розвитку та командної роботи користувачів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ СИСТЕМИ

2.1 Вибір технологій та інструментів для розробки

Вибір технологій для розробки соціальної платформи для ІТ-спільнот є одним із ключових етапів проєктування системи, оскільки саме технологічний стек визначає продуктивність, масштабованість, безпеку, зручність підтримки та подальший розвиток програмного продукту. Для реалізації платформи Stack-Fate було обрано full-stack підхід, який передбачає окрему реалізацію frontend та backend частини системи. Такий підхід дозволяє розділити відповідальність між клієнтською та серверною частинами, спростити підтримку коду, забезпечити незалежне розгортання компонентів та підвищити гнучкість архітектури.

Таблиця 2.1

Технології, використані при розробці платформи Stack-Fate

Компонент системи	Технологія	Призначення
Backend	Django, Django Ninja	Реалізація REST API та бізнес-логіки
Frontend	Next.js, React	Побудова користувацького інтерфейсу
Мова frontend	TypeScript	Типізація клієнтського коду
Стилізація	Tailwind CSS	Створення адаптивного інтерфейсу
База даних	PostgreSQL	Зберігання даних системи
Авторизація	JWT	Автентифікація користувачів
Контейнеризація	Docker	Ізоляція та розгортання сервісів
Вебсервер	Nginx	Маршрутизація та HTTPS

Для реалізації платформи Stack-Fate було обрано full-stack підхід, який передбачає окрему реалізацію клієнтської та серверної частин системи. Такий

підхід забезпечує незалежну розробку компонентів, спрощує підтримку коду та створює можливість подальшого масштабування системи.

Серверна частина реалізована мовою Python із використанням фреймворків Django та Django Ninja. Django забезпечує роботу з базою даних, механізми автентифікації та засоби безпеки, а Django Ninja використовується для реалізації REST API та автоматичної генерації OpenAPI-документації.

Для зберігання даних використовується PostgreSQL — реляційна система керування базами даних, яка забезпечує надійне зберігання інформації та підтримку складних зв'язків між сутностями системи.

Клієнтська частина реалізована за допомогою Next.js, React, TypeScript і Tailwind CSS. Використання цих технологій дозволяє створити адаптивний інтерфейс користувача, забезпечити типізацію клієнтського коду та спростити підтримку компонентної архітектури застосунку.

Взаємодія між frontend та backend реалізована на основі REST API, що забезпечує незалежність компонентів системи та створює можливість подальшої інтеграції мобільних застосунків і сторонніх сервісів. Для автентифікації користувачів використовується JWT, а контейнеризація компонентів виконується за допомогою Docker. Для маршрутизації запитів та підтримки HTTPS використовується вебсервер Nginx.

Таким чином, обраний технологічний стек забезпечує реалізацію основного функціоналу платформи, підтримує масштабованість, безпеку та створює основу для подальшого розвитку системи.

2.2 Архітектура системи

Архітектура системи є одним із ключових етапів проєктування соціальної платформи для ІТ-спільнот, оскільки вона визначає принципи взаємодії між

компонентами застосунку, обробки даних, авторизації користувачів, масштабування та подальшої підтримки програмного продукту. Для реалізації платформи Stack-Fate використано клієнт-серверну архітектуру з чітким розділенням frontend та backend частини.

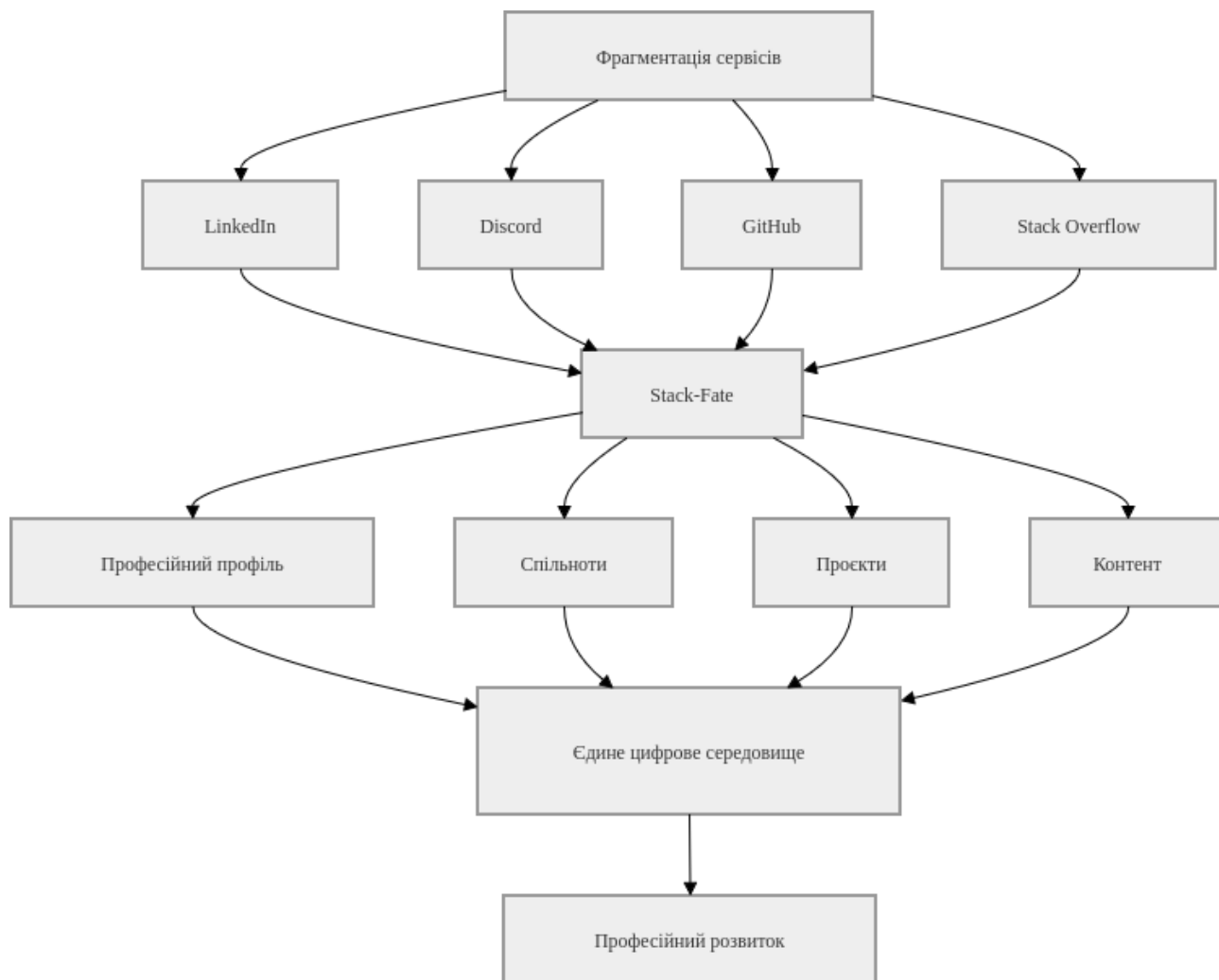
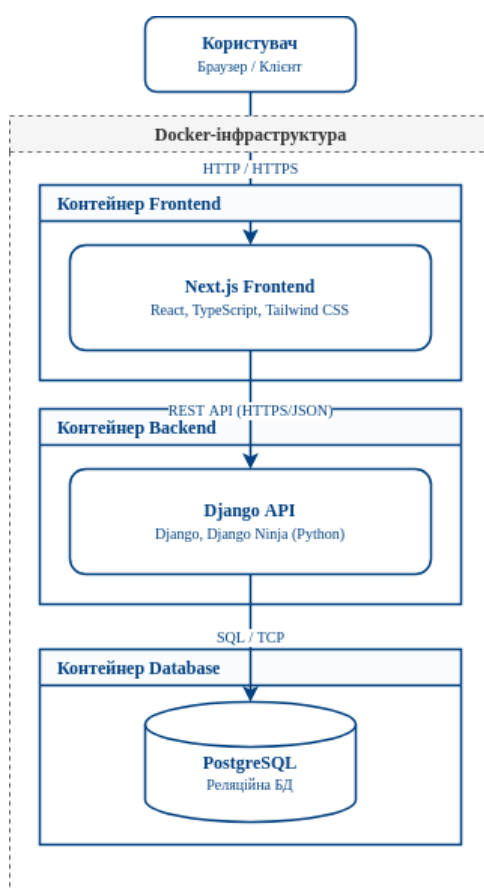


Рис. 2.1 – Концептуальна модель взаємодії користувачів платформи Stack-Fate

Запропонована концептуальна модель демонструє взаємозв'язок основних функціональних підсистем платформи. Центральним елементом системи є користувач та його професійний профіль. На основі інформації про навички, досягнення та активність користувач може брати участь у спільнотах, формувати професійні зв'язки, долучатися до проєктів та взаємодіяти з іншими учасниками платформи. На відміну від існуючих рішень, запропонована

структура поєднує механізми комунікації, професійного розвитку та командної роботи в межах єдиного цифрового середовища.

Загальна архітектура системи побудована за принципом full-stack вебзастосунку. Клієнтська частина відповідає за відображення інтерфейсу користувача та взаємодію з REST API, а серверна частина — за бізнес-логіку, обробку запитів, авторизацію, перевірку прав доступу та роботу з базою даних. Такий підхід дозволяє розділити відповідальність між рівнями системи та



забезпечити гнучкість її подальшого розвитку.

Рис. 2.2 – Загальна схема архітектури системи Stack-Fate

Frontend частина системи реалізована на основі Next.js та React. Вона відповідає за маршрутизацію сторінок, обробку форм, відображення даних, отриманих із сервера, та взаємодію користувача з платформою. До основних функціональних сторінок належать сторінки реєстрації та авторизації, профіль

користувача, стрічка дописів, сторінки спільнот, налаштування профілю та форми створення контенту.

Backend частина реалізована на основі Django та Django Ninja. Вона виконує роль центрального сервісу, який приймає HTTP-запити від frontend частини, перевіряє їх коректність, виконує бізнес-логіку, взаємодіє з базою даних PostgreSQL та повертає відповідь у форматі JSON. Використання Django Ninja дозволяє створити структурований REST API із підтримкою типізації та автоматичною генерацією OpenAPI-документації.

Основна взаємодія між frontend та backend частинами здійснюється через REST API. Frontend надсилає HTTP-запити до серверної частини, а backend обробляє їх, виконує необхідні операції з даними та повертає структуровані JSON-відповіді. Завдяки такому підходу клієнтська та серверна частини можуть розроблятися, тестуватися та масштабуватися незалежно одна від одної.

Для автентифікації користувачів у системі використовується JWT-авторизація. Після успішного входу користувач отримує access token та refresh token. Access token використовується для доступу до захищених ресурсів, а refresh token — для оновлення сесії користувача без повторного введення облікових даних. Захищені сторінки та API endpoint-и доступні лише авторизованим користувачам із дійсним токеном доступу.

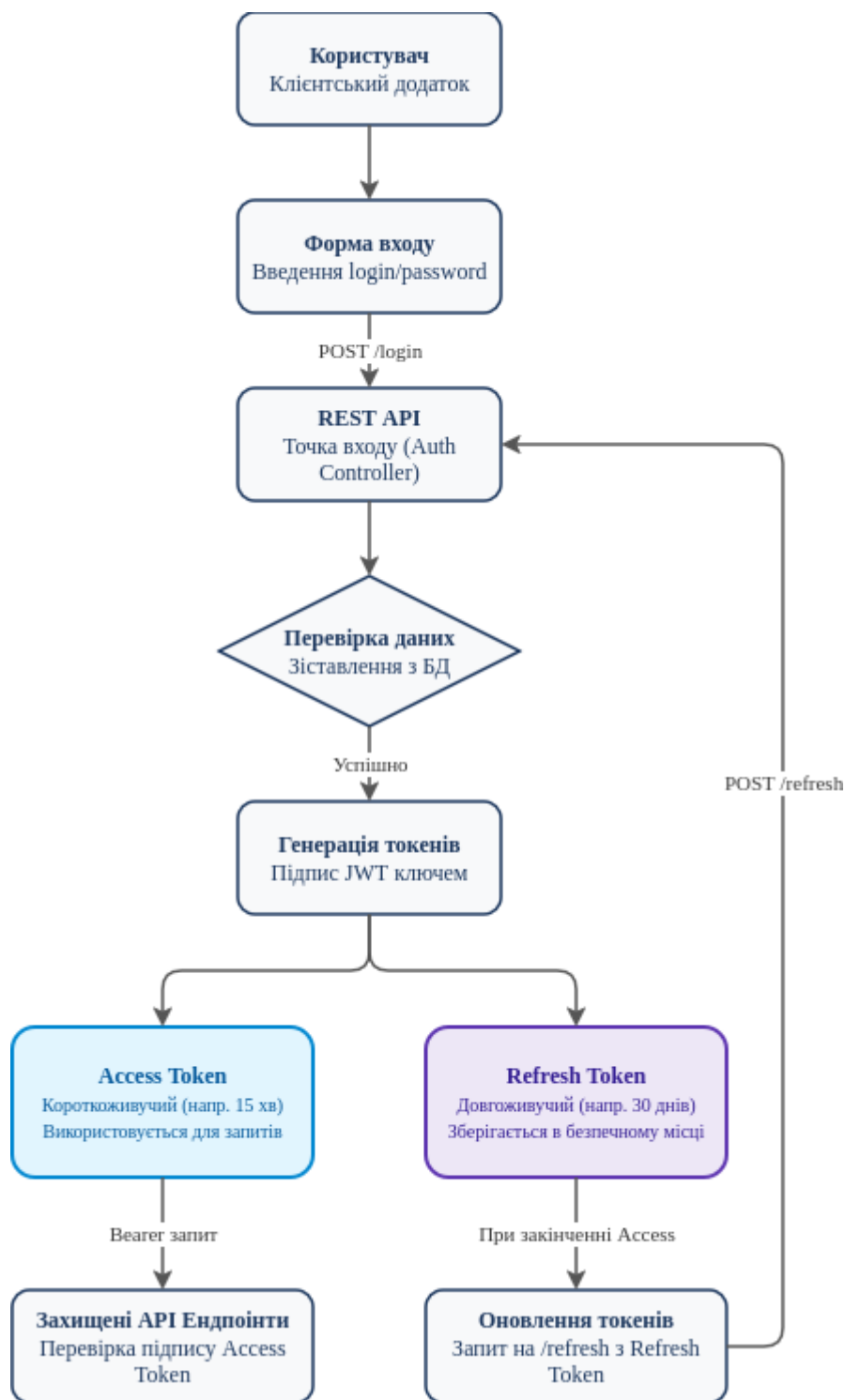


Рис. 2.3 – JWT авторизація

Важливим елементом архітектури є база даних PostgreSQL. Вона використовується для зберігання інформації про користувачів, профілі, спільноти, дописи, заявки на вступ, ролі учасників та медіафайли. Backend взаємодіє з базою даних через ORM Django, що дозволяє працювати з

таблицями як із Python-класами, використовувати міграції та підтримувати цілісність структури даних.

Архітектура backend частини побудована з урахуванням розділення відповідальності між окремими шарами. На рівні моделей описується структура даних. На рівні схем визначаються формати вхідних і вихідних даних API. Роутери відповідають за приймання HTTP-запитів і повернення відповідей. Сервіси містять бізнес-логіку системи, а репозиторії та селектори використовуються для доступу до даних.

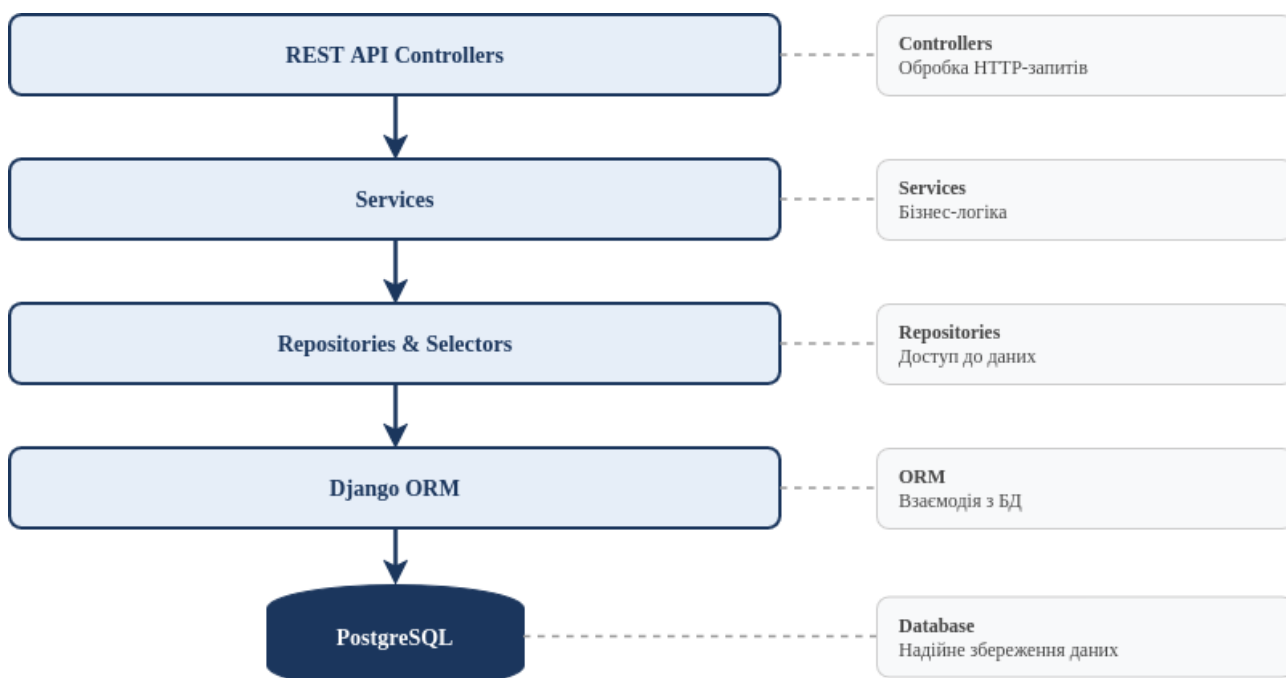


Рис. 2.4 – Архітектура Backend

Такий підхід дозволяє уникнути надмірного навантаження на контролери та зробити код більш підтримуваним. Наприклад, роутер не повинен містити складну бізнес-логіку, а лише приймати запит, передавати дані у відповідний сервіс та повертати результат. Перевірка бізнес-правил, створення користувача, оновлення профілю, створення допису або обробка заявки на вступ до спільноти виконуються на рівні сервісів.

Frontend частина системи також побудована за модульним принципом. Основні функціональні блоки розділені на окремі модулі: auth, profile, posts,

communities та settings. Кожен модуль містить компоненти інтерфейсу, функції взаємодії з API, типи даних та допоміжну логіку, пов'язану з конкретною предметною областю. Це спрощує підтримку коду та подальше розширення функціональності платформи.

У системі передбачено роботу з медіафайлами, зокрема з аватаром користувача та зображеннями дописів. Для цього backend реалізує endpoint-и для завантаження файлів, а frontend забезпечує відповідні форми вибору та надсилення зображень. Під час обробки файлів необхідно враховувати перевірку типу, розміру та безпечність завантажених даних.

Окрему роль в архітектурі відіграє система спільнот. Спільнота є логічним об'єднанням користувачів навколо певної теми або напрямку. У межах спільноти користувачі можуть переглядати учасників, створювати дописи, подавати заявки на вступ та взаємодіяти з адміністраторами. Це формує основу платформи як середовища професійної взаємодії IT-спільнот.

Система дописів є одним із базових функціональних модулів платформи. Вона дозволяє користувачам створювати, переглядати, редагувати та видаляти публікації. Дописи можуть бути пов'язані з конкретною спільнотою або відображатися у загальній стрічці платформи. Такий функціонал забезпечує базову соціальну взаємодію між користувачами.

Для розгортання системи використовується контейнеризація за допомогою Docker. Frontend, backend та база даних запускаються як окремі контейнери, що забезпечує ізоляцію компонентів і спрощує налаштування середовища. Docker Compose використовується для опису взаємодії між контейнерами та запуску всієї інфраструктури однією командою.

У production середовищі для обробки зовнішніх HTTP/HTTPS-запитів використовується Nginx як reverse proxy. Він приймає запити від користувачів і перенаправляє їх до відповідного сервісу. Наприклад, запити до клієнтської

частини обробляються frontend застосунком, а запити до API — backend сервісом Django Ninja.

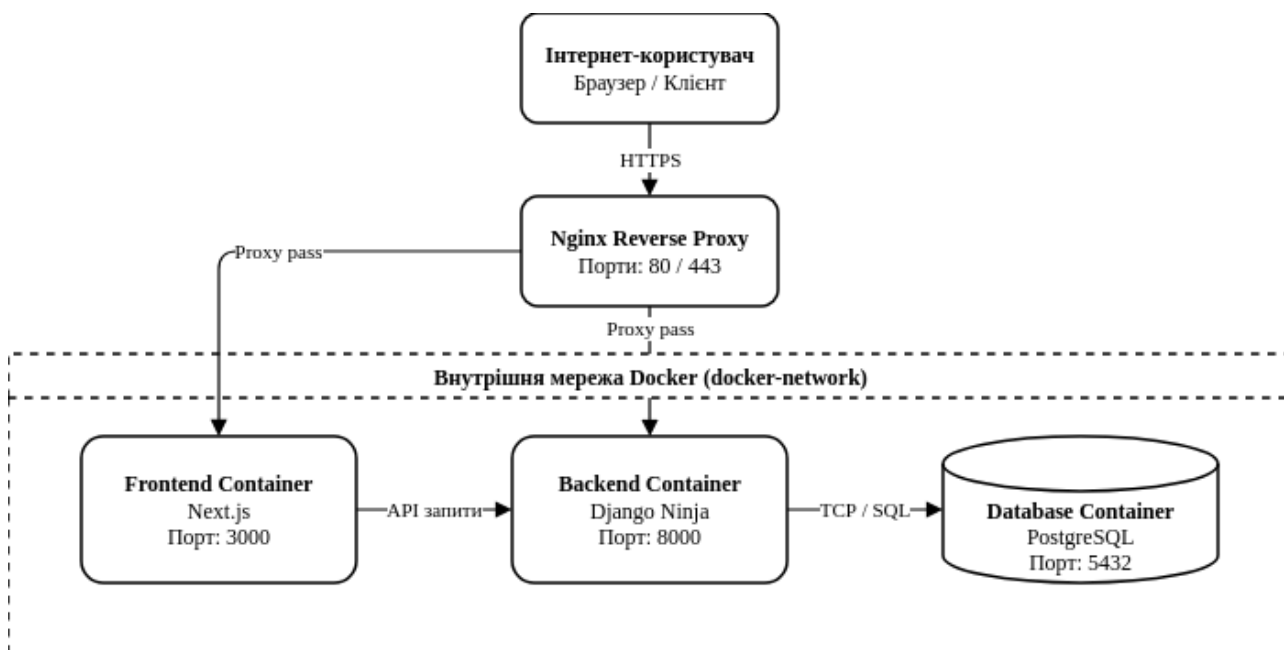


Рис. 2.5 – Контейнеризоване розгортання системи

Обрана архітектура дозволяє незалежно масштабувати окремі компоненти системи. При збільшенні навантаження на клієнтську частину можна окремо оптимізувати frontend. Якщо зростає кількість API-запитів, можна масштабувати backend сервіс. При збільшенні обсягу даних можлива оптимізація структури бази даних, індексів та конфігурації PostgreSQL.

Безпека системи забезпечується на декількох рівнях. На рівні авторизації використовується JWT. На рівні API виконується перевірка доступу до захищених endpoint-ів. На рівні бази даних забезпечується цілісність інформації через реляційні зв'язки та обмеження. На рівні інфраструктури використовується ізоляція контейнерів і можливість налаштування HTTPS через Nginx.

Обрана архітектура також створює основу для подальшого розвитку системи. У майбутньому до платформи можна додати модуль модерації спільнот, систему рекомендацій дописів, функціонал для роботодавців, пошук кандидатів, запрошення учасників до проєктів, публікацію особистих або

community-проектів, а також мобільний застосунок, який зможе взаємодіяти з тим самим REST API.

Таким чином, архітектура системи Stack-Fate побудована на основі сучасного клієнт-серверного підходу з розділенням frontend, backend та рівня зберігання даних. Вона забезпечує масштабованість, підтримуваність, безпечну взаємодію користувачів, надійне зберігання інформації та можливість подальшого розширення функціональності платформи.

2.3 Проектування бази даних та основного функціоналу

Одним із найважливіших етапів проектування соціальної платформи для ІТ-спільнот є розробка структури бази даних та визначення основного функціоналу системи. Від якості проектування бази даних залежить ефективність зберігання інформації, швидкість виконання запитів, підтримка цілісності даних та можливість подальшого масштабування програмного продукту.

Під час проектування бази даних було проведено аналіз предметної області та визначено основні сутності системи, а також зв'язки між ними. Основною метою проектування є створення структури даних, яка забезпечує реалізацію функцій реєстрації користувачів, керування профілями, створення спільнот, публікації дописів та взаємодії між учасниками платформи.

Для реалізації системи використовується реляційна система керування базами даних PostgreSQL. Реляційна модель дозволяє ефективно описувати взаємозв'язки між користувачами, спільнотами та контентом, забезпечуючи цілісність і узгодженість даних.

На основі аналізу функціональних вимог було визначено основні сутності системи та зв'язки між ними. Загальну структуру бази даних і взаємозв'язки між сутностями наведено на рис. 2.5.

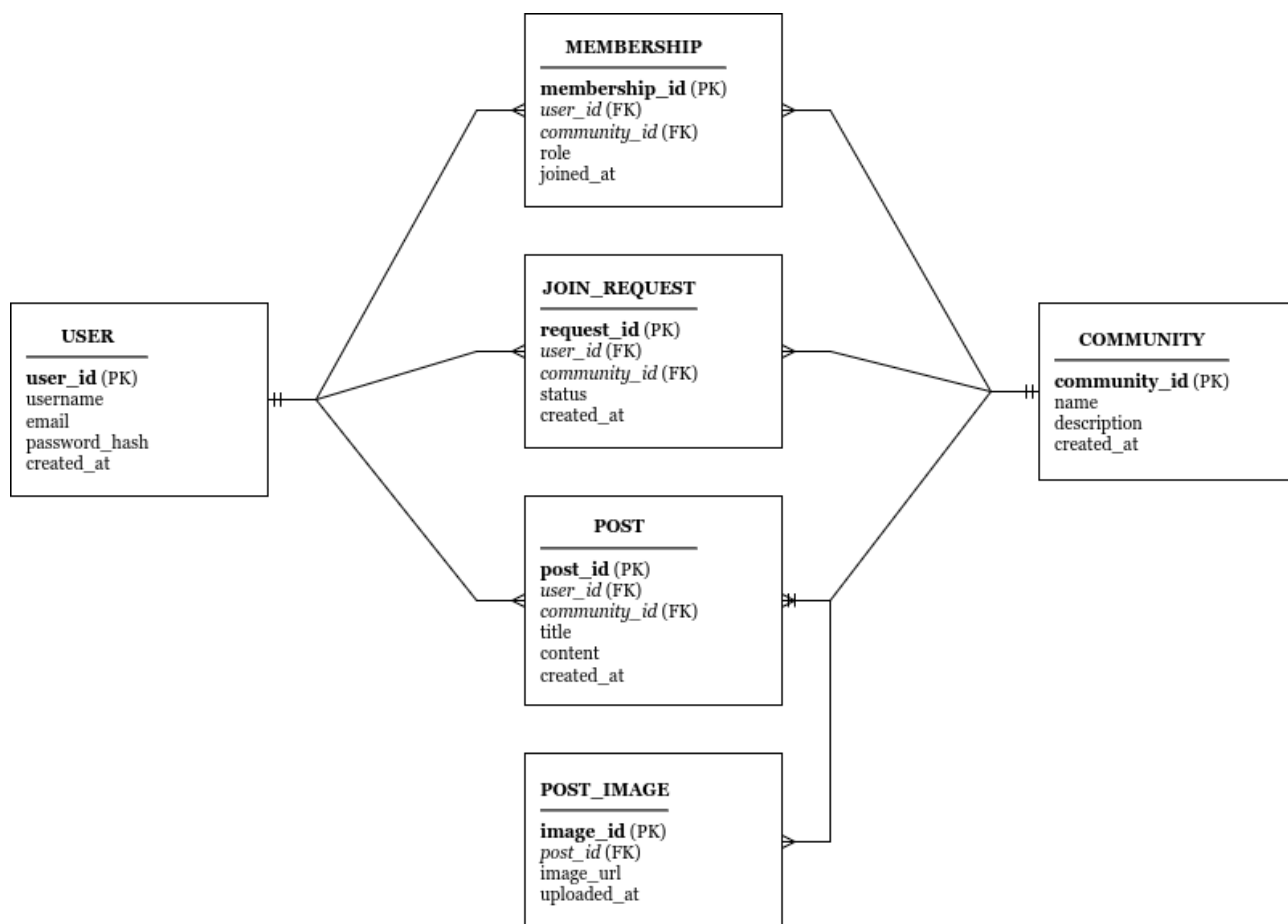


Рис. 2.6 – ER-діаграма бази даних

Центральним елементом системи є сутність користувача (User). Вона забезпечує зберігання інформації про обліковий запис користувача, його профіль та налаштування. Користувач може створювати дописи, вступати до спільнот, подавати заявки на вступ та взаємодіяти з іншими учасниками платформи.

Сутність Community використовується для організації користувачів за інтересами, технологіями або напрямками діяльності. Кожна спільнота має власний інформаційний простір для публікації контенту та взаємодії учасників.

Для реалізації зв'язку типу «багато до багатьох» між користувачами та спільнотами використовується сутність Membership. Вона також забезпечує зберігання ролей учасників спільноти, що дозволяє реалізувати гнучку систему керування правами доступу.

Сутність JoinRequest використовується для контролю процесу вступу до закритих спільнот. Вона містить інформацію про користувача, спільноту, повідомлення заявки та її поточний статус.

Сутність Post відповідає за зберігання користувацького контенту. Дописи можуть бути прив'язані до конкретної спільноти або відображатися у загальній стрічці платформи.

Для підтримки мультимедійного контенту використовується сутність PostImage, яка забезпечує можливість прикріплення одного або декількох зображень до допису.

Таблиця 2.2

Основні сутності бази даних

Сутність	Призначення
User	Дані користувачів
Community	Дані спільнот
Membership	Членство та ролі
JoinRequest	Заявки на вступ
Post	Дописи користувачів
PostImage	Зображення дописів

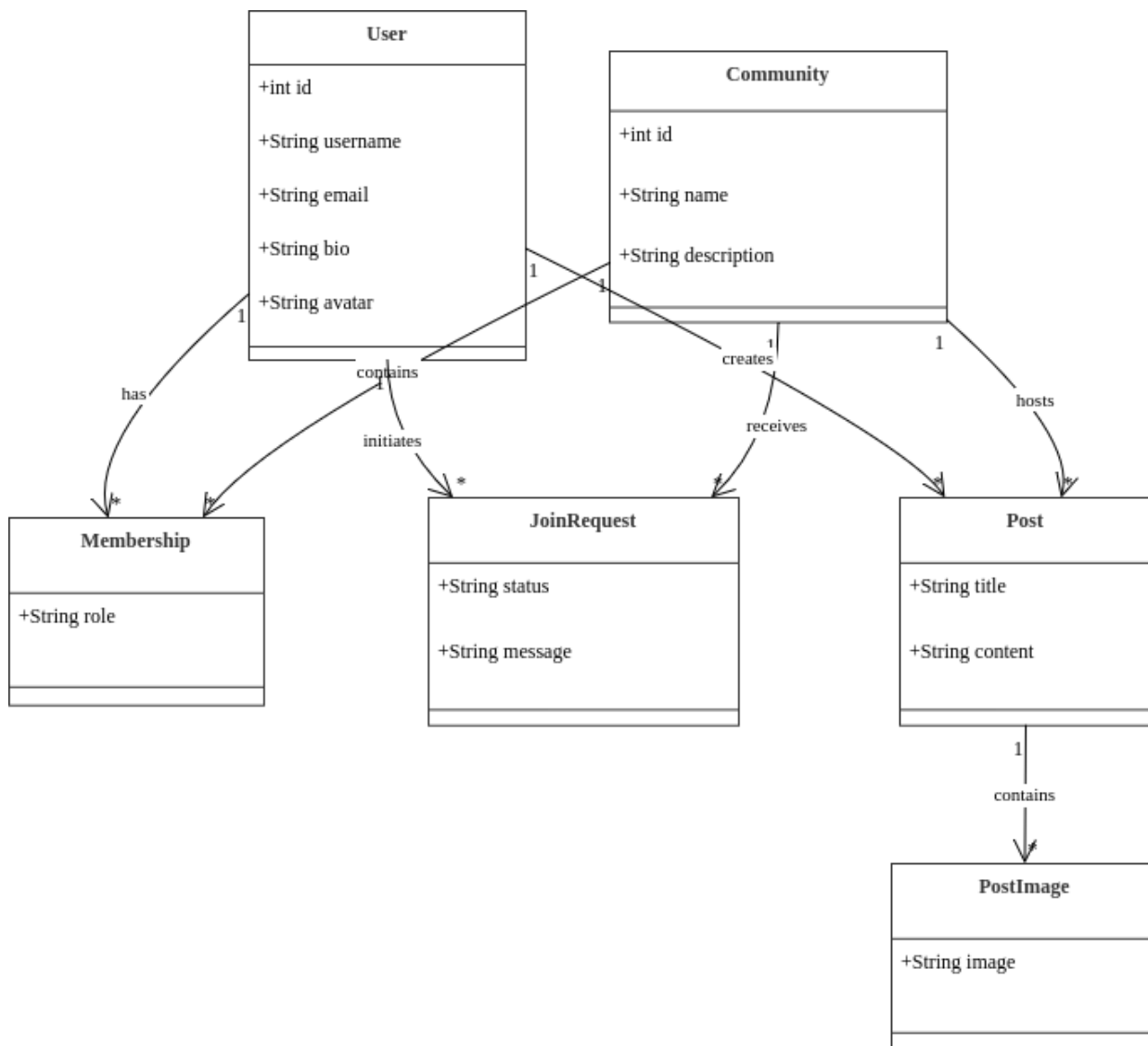
Під час проєктування бази даних особлива увага приділялася нормалізації структури таблиць. Використання нормалізованої моделі дозволяє мінімізувати дублювання даних, забезпечити цілісність інформації та зменшити витрати ресурсів на її зберігання.

Для забезпечення продуктивності системи використовуються індекси для полів, які найчастіше застосовуються під час пошуку та фільтрації інформації. Індексация використовується для електронної пошти користувачів, імен

користувачів, назв спільнот, дат створення дописів та зовнішніх ключів між таблицями.

Рис. 2.7 – Основні сутності системи

Після проектування структури бази даних було визначено основний функціонал платформи. На поточному етапі реалізації система підтримує

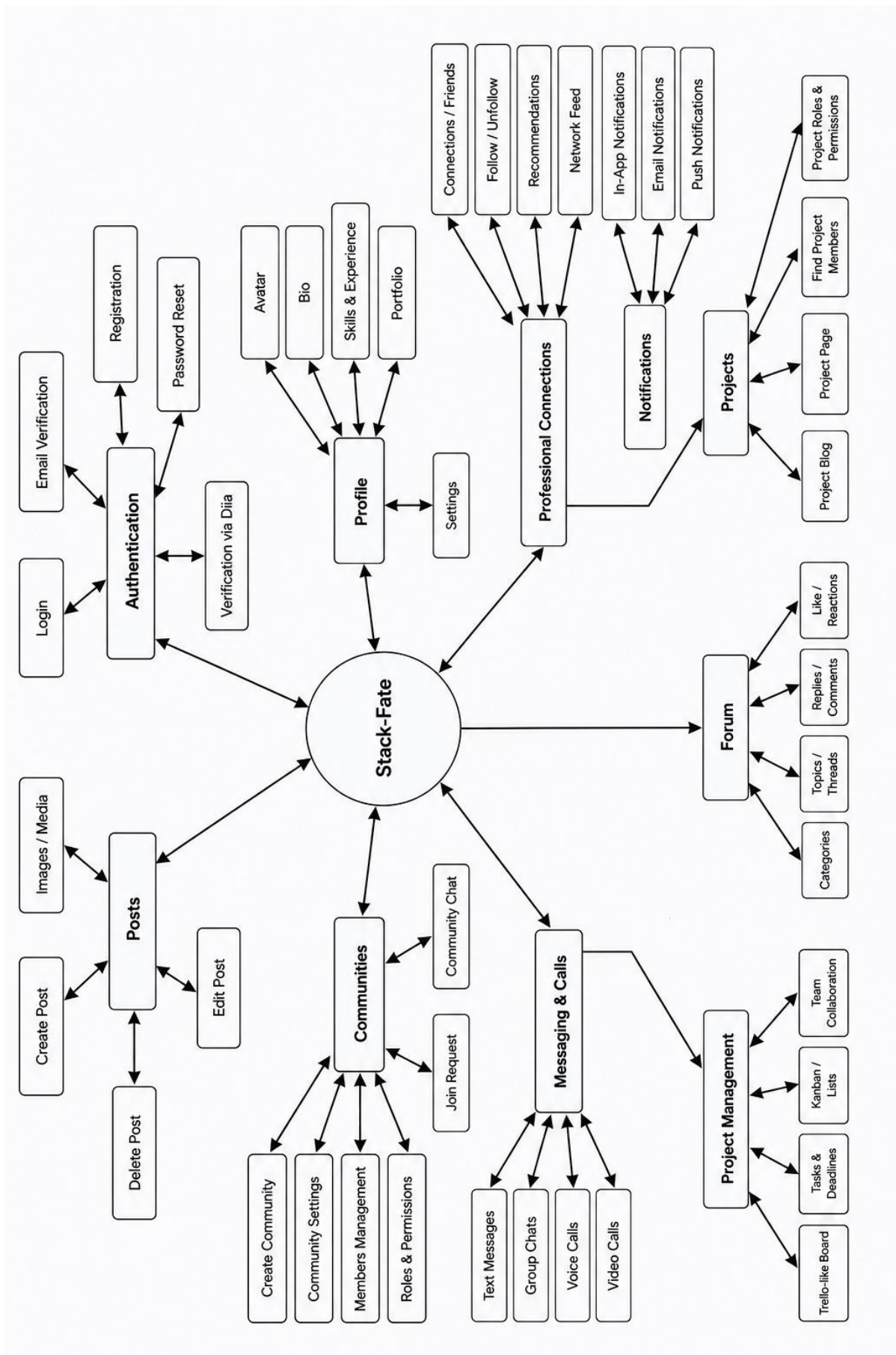


базовий набір функцій соціальної мережі та професійної платформи для IT-спільнот.

До функціональних можливостей системи належать:

- реєстрація користувачів;

- підтвердження електронної пошти;
- авторизація та автентифікація;
- відновлення пароля;
- керування профілем користувача;
- завантаження аватара;
- створення спільнот;
- подання заявок на вступ до спільнот;
- керування учасниками спільнот;
- створення та редагування дописів;
- завантаження зображень;
- перегляд стрічки контенту.



Для забезпечення безпеки системи використовується JWT-автентифікація. Після успішної авторизації користувач отримує токен доступу, який використовується для виконання захищених операцій. Такий підхід дозволяє реалізувати сучасну модель взаємодії між frontend та backend частинами системи.

Важливою особливістю проєкту є використання REST API як основного механізму взаємодії між клієнтською та серверною частинами. Усі операції із користувачами, спільнотами та дописами виконуються через спеціалізовані API endpoint-и. Це забезпечує незалежність frontend та backend компонентів та створює можливість подальшої реалізації мобільного застосунку без необхідності зміни серверної логіки.

Запроєктована структура бази даних дозволяє у майбутньому розширювати платформу шляхом додавання нових сутностей та функціональних модулів без суттєвих змін існуючої архітектури.

Таким чином, проєктування бази даних дозволило сформувати структуровану модель предметної області, яка забезпечує реалізацію основного функціоналу соціальної платформи для ІТ-спільнот. Запропонована структура даних є масштабованою, підтримуваною та придатною для подальшого розвитку системи відповідно до потреб користувачів та майбутніх функціональних вимог.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ВЕБ-ДОДАТКУ ТА RESTAPI-SERVICE

3.1 Налаштування середовища розробки та налаштування проєкту

Після завершення етапу проєктування системи було виконано налаштування середовища розробки та створення структури проєкту. Правильна організація середовища розробки є важливою складовою процесу створення сучасних вебзастосунків, оскільки впливає на стабільність програмного забезпечення, зручність командної розробки, швидкість тестування та подальше розгортання системи.

Для реалізації соціальної платформи Stack-Fate використано full-stack підхід із розділенням frontend та backend частини. Така архітектура дозволяє незалежно розробляти клієнтський та серверний компоненти, спрощує підтримку програмного коду та забезпечує можливість масштабування окремих частин системи.

Розробка проєкту виконувалася із використанням операційної системи Linux та редактора Visual Studio Code. Для керування вихідним кодом застосовувалася система контролю версій Git, а віддалене зберігання репозиторію здійснюється на платформі Github.

Таблиця 3.1

Основні інструменти середовища розробки

Категорія	Інструмент
Операційна система	Linux
Редактор коду	Visual Studio Code
Контроль версій	Git
Віддалений репозиторій	Github
Backend	Django, Django Ninja

Frontend	Next.js, React, TypeScript
База даних	PostgreSQL
Контейнеризація	Docker
Менеджер залежностей	uv
Форматування коду	Biome
API тестування	Swagger UI, Postman

Однією з основних вимог до сучасних програмних систем є забезпечення однакових умов виконання незалежно від операційної системи або апаратного забезпечення. Для вирішення цього завдання у проєкті використовується контейнеризація за допомогою Docker.

Docker дозволяє запускати окремі компоненти системи в ізольованих контейнерах, які містять усі необхідні залежності та налаштування. Такий підхід усуває проблему несумісності програмного забезпечення та забезпечує однакову поведінку застосунку на різних пристроях і серверах.

Для спрощення керування контейнерами використовується Docker Compose. За його допомогою вся інфраструктура проєкту описується у вигляді конфігураційного файлу та запускається однією командою.

У межах системи використовуються окремі контейнери для frontend застосунку, backend сервісу, PostgreSQL та Redis. Взаємодія між сервісами здійснюється через внутрішню Docker-мережу.

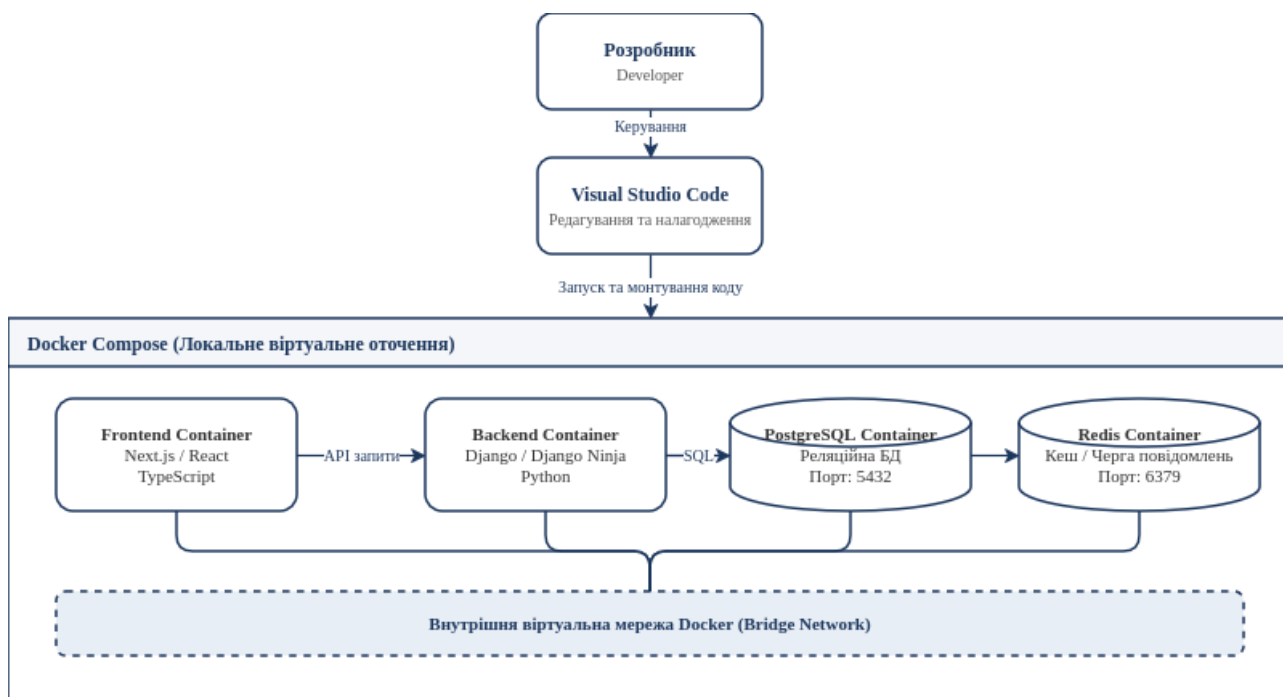


Рис. 3.1 – Інфраструктура середовища розробки

Для реалізації серверної частини проєкту використовується Python та сучасний менеджер залежностей `uv`. Даний інструмент забезпечує швидке встановлення пакетів, централізоване керування залежностями та фіксацію версій бібліотек, що сприяє стабільності роботи застосунку.

Після створення проєкту Django було виконано налаштування конфігурації бази даних, системи авторизації, логування, роботи зі статичними файлами та медіаданими. Для зберігання конфіденційних параметрів використовуються змінні середовища, зокрема секретний ключ Django, параметри підключення до PostgreSQL, JWT-конфігурація та налаштування SMTP-сервера.

Backend частина побудована відповідно до принципів багатошарової архітектури. Структура проєкту включає моделі, репозиторії, селектори, сервіси, схеми та контролери. Такий підхід дозволяє розділити відповідальність між окремими компонентами системи та забезпечити кращу підтримуваність програмного коду.

Моделі використовуються для опису структури даних та взаємодії з базою даних. Репозиторії відповідають за операції створення, оновлення та видалення даних. Селектори реалізують логіку читання інформації. Сервіси містять бізнес-логіку системи. Схеми використовуються для валідації даних та опису структури API, а контролери забезпечують взаємодію між HTTP-запитами та сервісним рівнем.

Для реалізації REST API використовується Django Ninja. Усі endpoint-и згруповані за функціональними модулями, що забезпечує логічну організацію проєкту та спрощує його підтримку.

У межах системи реалізовано модулі користувачів, авторизації, спільнот, дописів та заявок на вступ до спільнот. Кожен модуль має власний набір endpoint-ів та ізольовану бізнес-логіку.

Frontend частина проєкту реалізована за допомогою Next.js та TypeScript. Для забезпечення типобезпеки використовується статична типізація, що дозволяє зменшити кількість помилок під час взаємодії з REST API та обробки даних.

Для побудови користувацького інтерфейсу використовується Tailwind CSS. Даний інструмент забезпечує швидке створення адаптивних компонентів та спрощує підтримку стилів застосунку.

Структура frontend застосунку організована за модульним принципом. Основними функціональними модулями є auth, profile, communities, posts та settings. Кожен модуль містить власні компоненти, типи даних, API-запити та логіку взаємодії із сервером.

Для роботи з HTTP-запитами використовується централізований API-шар, який забезпечує взаємодію між клієнтською та серверною частинами системи. Такий підхід спрощує підтримку проєкту та дозволяє швидко вносити зміни до механізмів взаємодії з API.

Особлива увага була приділена налаштуванню процесу автентифікації користувачів. Після успішної авторизації користувач отримує JWT-токени, які використовуються для доступу до захищених ресурсів системи. Frontend автоматично додає токен доступу до HTTP-запитів та виконує його оновлення за допомогою refresh token.

Для забезпечення якості програмного коду використовуються інструменти статичного аналізу та форматування. На frontend стороні використовується Biome, який забезпечує автоматичну перевірку стилю коду та пошук потенційних помилок. Для backend частини застосовуються інструменти типізації та аналізу Python-коду.

Під час розробки також було налаштовано систему міграцій бази даних. Міграції дозволяють керувати змінами структури бази даних та підтримувати її актуальний стан у процесі розвитку програмного продукту.

Для тестування REST API використовується Swagger UI, який автоматично генерується Django Ninja на основі описаних схем та endpoint-ів. Це дозволяє швидко перевіряти працездатність API без використання додаткових інструментів.

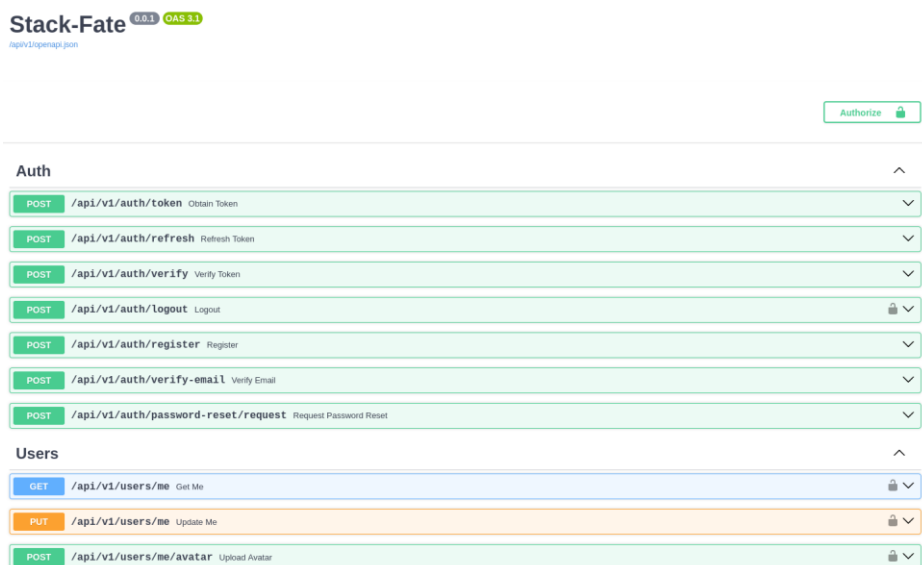


Рис. 3.2 – Swagger документація

Додатково для тестування серверної частини використовувався Postman. За допомогою цього інструменту виконувалася перевірка роботи авторизації, реєстрації користувачів, створення спільнот та взаємодії з дописами.

Для запуску проекту в середовищі розробки достатньо виконати запуск Docker Compose, після чого автоматично запускаються всі необхідні сервіси системи. Frontend стає доступним через веббраузер, backend починає обробляти API-запити, а PostgreSQL забезпечує зберігання даних.

Таким чином, у межах даного етапу було виконано налаштування повноцінного середовища розробки для створення соціальної платформи Stack-Fate. Використання Docker, Django Ninja, PostgreSQL, Next.js та TypeScript дозволило сформувати сучасну full-stack інфраструктуру, яка забезпечує зручність розробки, високу продуктивність та можливість подальшого масштабування системи.

3.2 Адміністрування серверу

Після завершення розробки програмного забезпечення важливим етапом є його розгортання та адміністрування серверної інфраструктури. Для тестування

та подальшої експлуатації соціальної платформи Stack-Fate було використано одноплатний комп'ютер Orange Pi 3B, який виконує роль фізичного сервера.

Обрана апаратна конфігурація включає процесор ARM-архітектури, 8 ГБ оперативної пам'яті та SSD-накопичувач Samsung M.2 обсягом 250 ГБ. Такі характеристики є достатніми для розміщення веб-застосунку, REST API сервісу та системи керування базами даних PostgreSQL у межах одного фізичного пристрою.

В якості серверної операційної системи використовується Armbian на базі Debian Trixie. Вибір даної операційної системи обумовлений її стабільністю, низьким споживанням ресурсів, широкою підтримкою Linux-інструментів та сумісністю з Docker-контейнерами. Крім того, Debian є однією з найпоширеніших серверних операційних систем, що спрощує адміністрування та підтримку програмного забезпечення.

Для забезпечення ізоляції компонентів системи використовується контейнеризація за допомогою Docker. Кожен компонент платформи запускається в окремому контейнері, що дозволяє розділити відповідальність між сервісами та спростити процес оновлення системи. У межах проєкту окремо запускаються frontend застосунок на Next.js, backend сервіс на Django Ninja, база даних PostgreSQL та допоміжні сервіси.

Для керування контейнерами використовується Docker Compose. Він дозволяє централізовано описати конфігурацію всієї інфраструктури та автоматизувати запуск необхідних сервісів. Такий підхід значно спрощує процес розгортання та забезпечує відтворюваність середовища на інших серверах.

Доступ користувачів до платформи забезпечується через вебсервер nginx, який виконує функцію reverse proxy. Сервер приймає HTTP-запити від клієнтів та перенаправляє їх до відповідних контейнерів системи. Крім маршрутизації

запитів, nginx може використовуватися для обробки SSL-сертифікатів, роботи з HTTPS та кешування статичних ресурсів.

Для забезпечення безпеки серверної інфраструктури використовуються стандартні механізми Linux, зокрема обмеження доступу через SSH, використання ключової автентифікації, контроль мережових портів та ізоляція контейнерів Docker. Такий підхід дозволяє знизити ризики несанкціонованого доступу до системи та підвищити загальний рівень безпеки платформи.

Використання Orange Pi як серверної платформи демонструє можливість розгортання сучасних веб-застосунків навіть на відносно недорогому обладнанні. Завдяки використанню Docker, PostgreSQL, Django Ninja та Next.js система залишається масштабованою та може бути перенесена на більш потужну серверну інфраструктуру без значних змін у програмному коді.

Таким чином, налаштована серверна інфраструктура забезпечує стабільну роботу соціальної платформи Stack-Fate, підтримує контейнеризоване розгортання компонентів системи та створює основу для подальшого розвитку й масштабування програмного продукту.

3.3 Реалізація веб-сервісу REST API

Реалізація веб-сервісу REST API є одним із ключових етапів створення соціальної платформи Stack-Fate, оскільки саме серверна частина забезпечує обробку запитів користувачів, виконання бізнес-логіки, взаємодію з базою даних, автентифікацію, авторизацію та передачу даних між клієнтською частиною і системою зберігання даних.

Для реалізації REST API було використано фреймворк Django Ninja, який працює поверх Django та забезпечує підтримку типізації, автоматичну валідацію даних і генерацію OpenAPI-документації. Такий підхід дозволив поєднати можливості Django ORM, системи міграцій та вбудованих механізмів безпеки з сучасною архітектурою API-сервісів.

Основною задачею REST API у межах платформи Stack-Fate є забезпечення взаємодії між клієнтською частиною, реалізованою на основі Next.js, та серверною логікою застосунку. Frontend надсилає HTTP-запити до backend сервісу, після чого сервер виконує перевірку прав доступу, обробляє бізнес-логіку, взаємодіє з PostgreSQL та повертає відповідь у форматі JSON.

У межах реалізованого REST API було створено декілька функціональних модулів.

Таблиця 3.2**Основні модулі REST API**

Модуль	Призначення
Authentication	Автентифікація та авторизація
Users	Робота з профілями користувачів
Communities	Керування спільнотами
Memberships	Керування учасниками
Join Requests	Обробка заявок на вступ
Posts	Робота з дописами
Media	Робота із зображеннями

Кожен із зазначених модулів має власний набір endpoint-ів та відповідає за окрему частину функціональності платформи. Такий підхід дозволяє забезпечити модульність архітектури та спрощує подальше розширення системи.

Основою серверної частини є Django-проект, у якому окремі домени винесені в самостійні застосунки. Логіка користувачів, спільнот, дописів та

автентифікації реалізована у вигляді окремих модулів, що відповідає принципу розділення відповідальності.

Для опису структури даних використовуються Django models. На основі моделей автоматично створюються міграції, які застосовуються для формування структури бази даних PostgreSQL.

Для обміну даними між клієнтом і сервером використовуються схеми Django Ninja. Вони визначають структуру вхідних і вихідних даних API, виконують автоматичну валідацію та забезпечують типобезпечну взаємодію між компонентами системи.

Одним із перших реалізованих модулів став модуль автентифікації та авторизації. Він забезпечує реєстрацію користувачів, вхід до системи, оновлення токенів доступу, підтвердження електронної пошти та відновлення пароля.

Для реалізації механізму автентифікації використовується JWT-підхід. Після успішної авторизації користувач отримує access token та refresh token. Access token використовується для доступу до захищених endpoint-ів, а refresh token — для оновлення access token без повторного введення облікових даних.

До основних endpoint-ів модуля автентифікації належать:

Таблиця 3.3

Основні endpoint-и модуля автентифікації

Endpoint	Призначення
/register	Реєстрація користувача
/token	Отримання JWT-токенів
/refresh	Оновлення access token
/verify	Перевірка токена
/logout	Вихід із системи

/verify-email	Підтвердження електронної пошти
/reset-password	Відновлення пароля

Реалізація даних endpoint-ів забезпечує повний цикл роботи з обліковим записом користувача — від створення акаунта до відновлення доступу до нього.

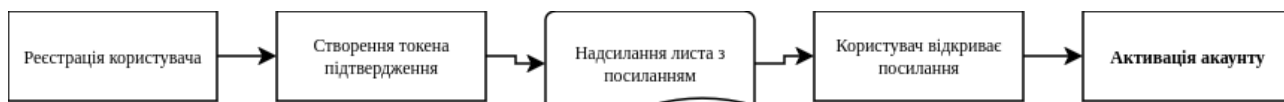


Рис. 3.2 – Підтвердження електронної пошти

Модуль користувачів відповідає за роботу з профілем користувача. У межах даного модуля реалізовано отримання інформації про поточного користувача, оновлення профілю та завантаження аватара. Для захисту персональної інформації всі операції доступні лише авторизованим користувачам.

Модуль спільнот забезпечує створення, перегляд та керування communities. У межах API реалізовано створення спільнот, перегляд списку спільнот, перегляд окремої спільноти, роботу з учасниками та механізм подання заявок на вступ.

Для реалізації контролю доступу використовується рольова модель, відповідно до якої окремі дії можуть виконувати лише власники або адміністратори спільнот.

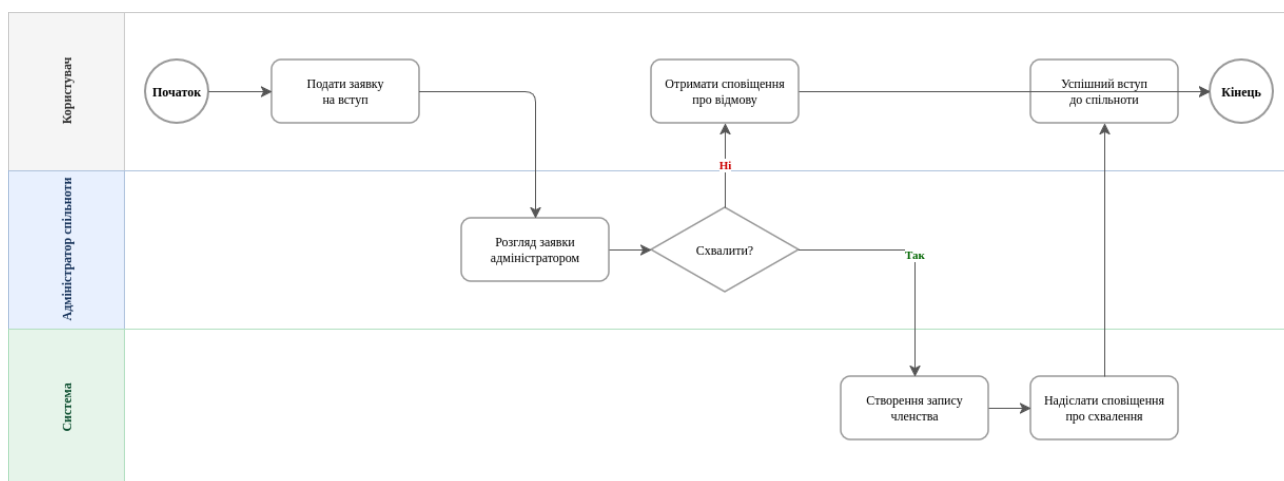


Рис. 3.3 – Схема вступу до спільноти

Окрему роль у системі відіграє модуль дописів. Він забезпечує створення, перегляд, редагування та видалення публікацій. Дописи можуть бути прив'язані до конкретної спільноти або відображатися у загальній стрічці новин платформи.

Під час створення допису сервер перевіряє коректність отриманих даних, визначає автора публікації та виконує збереження інформації у базі даних. Для операцій редагування та видалення реалізовано перевірку прав доступу.

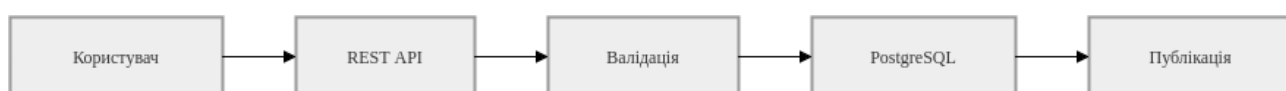


Рис. 3.4 – Створення допису

Для забезпечення коректної роботи REST API реалізовано стандартизовану систему обробки помилок. Залежно від результату виконання запиту сервер повертає відповідні HTTP-статуси, зокрема 200 OK, 201 Created, 400 Bad Request, 401 Unauthorized, 403 Forbidden та 404 Not Found.

Важливою перевагою використання Django Ninja є автоматична генерація OpenAPI-документації. Усі endpoint-и, схеми запитів, параметри та структури відповідей автоматично відображаються у Swagger UI.

Swagger UI дозволяє виконувати тестові запити безпосередньо з браузера та перевіряти роботу реалізованих endpoint-ів без використання сторонніх інструментів.

Додатково для тестування API використовувався Postman, за допомогою якого перевірялася робота авторизації, реєстрації користувачів, створення спільнот та роботи з дописами.

Особлива увага була приділена структурі backend-коду. Для забезпечення підтримуваності та масштабованості проєкту використовується багат шарова архітектура, яка включає routers, schemas, services, repositories, selectors та models.

Routers відповідають за приймання HTTP-запитів. Schemas використовуються для опису структури даних. Services реалізують бізнес-логіку. Repositories та selectors забезпечують взаємодію з базою даних. Models визначають структуру таблиць PostgreSQL.

Такий підхід дозволяє ізолювати окремі рівні системи та значно спрощує подальшу підтримку й розвиток проєкту.

Таким чином, у межах роботи було реалізовано повноцінний REST API сервіс, який забезпечує автентифікацію користувачів, роботу з профілями, спільнотами, заявками на вступ та дописами. Використання Django Ninja дозволило створити типізований, документований та підтримуваний API, який забезпечує ефективну взаємодію між frontend частиною платформи та базою даних PostgreSQL.

3.4 Реалізація frontend частини

Frontend частина платформи Stack-Fate реалізована як окремий вебзастосунок із використанням Next.js, React та TypeScript. Основним призначенням клієнтської частини є забезпечення взаємодії користувача із системою, відображення інформації, отриманої через REST API, та надання зручного інтерфейсу для роботи з функціональними можливостями платформи.

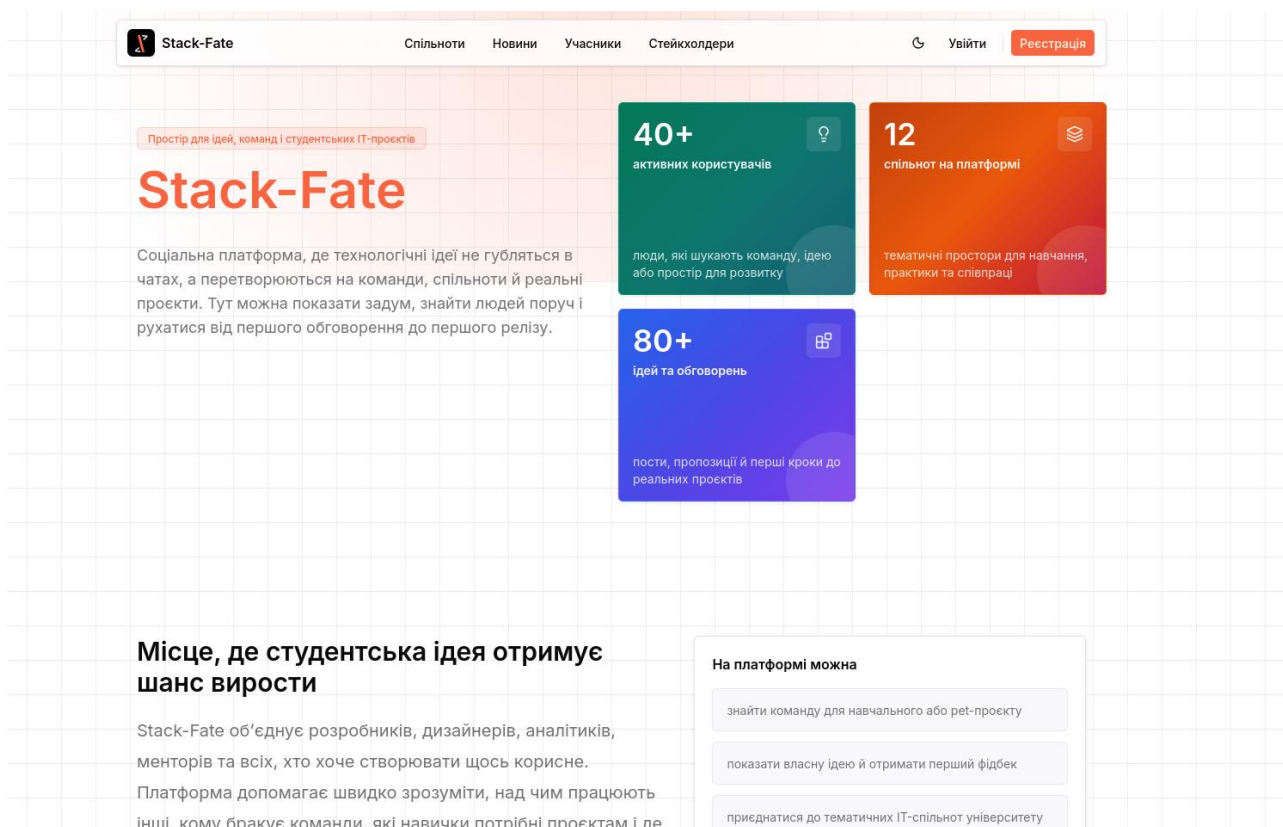


Рис. 3.5 – Головна сторінка frontend застосунку

Під час розробки frontend частини особлива увага приділялася швидкодії інтерфейсу, адаптивності, модульності коду та зручності подальшої підтримки. Для реалізації застосунку використовується Next.js App Router, який забезпечує сучасний підхід до організації маршрутизації та структури проекту.

Для забезпечення типобезпеки використовується TypeScript. Статична типізація дозволяє зменшити кількість помилок під час роботи з даними REST API та підвищує надійність програмного забезпечення.

Інтерфейс користувача реалізовано за допомогою Tailwind CSS та набору компонентів Shadcn/UI. Використання готових компонентів дозволило прискорити розробку та забезпечити єдиний стиль оформлення всіх сторінок платформи.

Архітектура frontend частини побудована за модульним принципом. Основний функціонал розділений на окремі логічні модулі.

Основні модулі frontend застосунку

Модуль	Призначення
Auth	Авторизація та реєстрація
Profile	Робота з профілем користувача
Communities	Робота зі спільнотами
Posts	Робота з дописами
Settings	Налаштування облікового запису

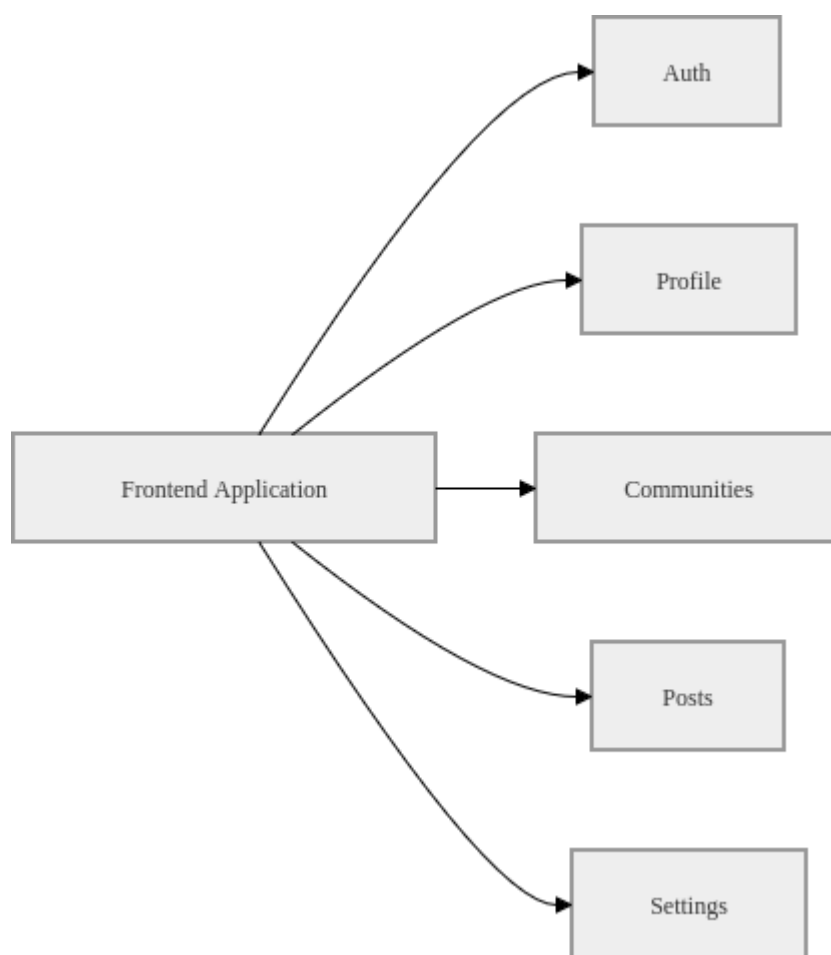


Рис. 3.6 – Структура frontend застосунку

Кожен модуль містить власні сторінки, React-компоненти, типи даних та функції взаємодії із REST API. Такий підхід дозволяє ізолювати бізнес-логіку окремих частин системи та забезпечує зручність подальшого розширення функціоналу.

Одним із перших реалізованих модулів є модуль автентифікації. Він включає сторінки реєстрації користувача, входу до системи, підтвердження електронної пошти та відновлення пароля.

Під час входу до системи користувач вводить облікові дані, після чого frontend надсилає відповідний запит до REST API. У разі успішної авторизації користувач отримує JWT-токени, які використовуються для доступу до захищених ресурсів системи.

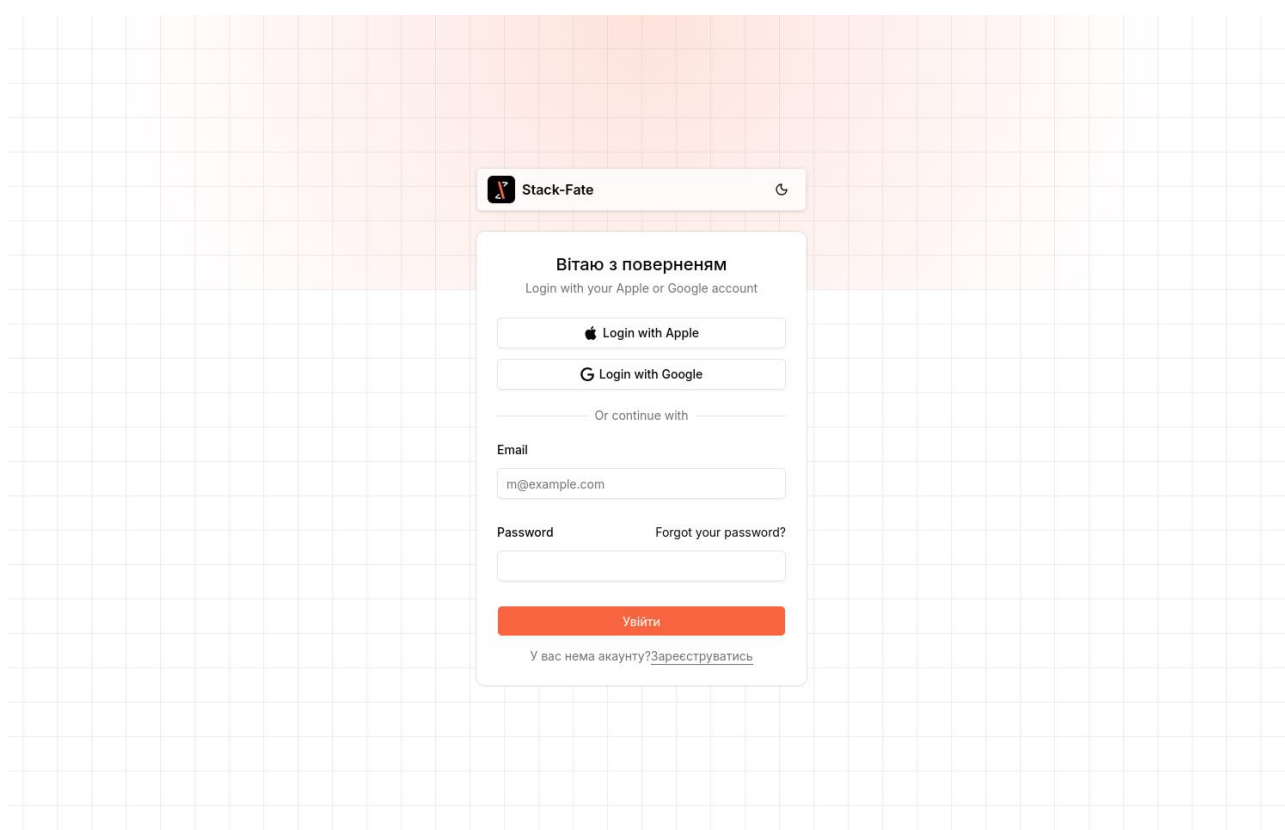


Рис. 3.7 – Сторінка авторизації

Після успішної реєстрації користувач отримує повідомлення про необхідність підтвердження електронної пошти. Для цього реалізовано окрему сторінку підтвердження електронної адреси.

Рис. 3.7 – Сторінка реєстрації

Для роботи з користувацькими даними реалізовано сторінку профілю. Користувач може переглядати та редагувати власні дані, змінювати інформацію про себе та завантажувати аватар.

Вся інформація профілю отримується через REST API та автоматично оновлюється після внесення змін.

Окрему роль у системі відіграє модуль налаштувань облікового запису. Він забезпечує можливість зміни особистих даних, оновлення пароля та керування параметрами профілю.

Для забезпечення безпеки всі операції виконуються лише після перевірки JWT-токена користувача.

Одним із центральних елементів платформи є система дописів. Для реалізації даного функціоналу створено окремі сторінки та компоненти для створення, перегляду та відображення контенту.

Користувач має можливість створювати текстові публікації, прикріплювати зображення та переглядати дописи інших учасників платформи.

Для відображення контенту використовується компонентна структура, яка включає картку допису, інформацію про автора, дату публікації, текстовий вміст та прикріплені зображення.

Такий підхід дозволяє повторно використовувати компоненти на різних сторінках системи та забезпечує однаковий стиль відображення контенту.

Для взаємодії зі спільнотами реалізовано окремий функціональний модуль. Користувач може переглядати список доступних спільнот, відкривати сторінку конкретної спільноти, переглядати її учасників та подавати заявку на вступ.

Усі операції виконуються через відповідні endpoint-и REST API.

Для забезпечення безпеки використовується механізм захищених маршрутів. Перед відображенням сторінок, доступних лише авторизованим користувачам, виконується перевірка наявності дійсного JWT-токена. У разі його відсутності користувач автоматично перенаправляється на сторінку входу.

Для взаємодії із серверною частиною реалізовано окремий API-шар. Він відповідає за надсилання HTTP-запитів до REST API, обробку відповідей та централізовану обробку помилок.

У випадку помилок авторизації користувач автоматично перенаправляється на сторінку входу. Помилки валідації відображаються безпосередньо у формах введення даних, що покращує користувацький досвід.

Для інформування користувача про результати виконання операцій реалізовано систему повідомлень про успішне виконання дій та виникнення помилок.

Важливою вимогою до сучасних вебзастосунків є адаптивність інтерфейсу. Усі сторінки платформи реалізовано відповідно до принципів responsive design та коректно відображаються на персональних комп'ютерах, планшетах і мобільних пристроях.

Для підтримки якості програмного коду використовується інструмент Biome, який забезпечує автоматичне форматування коду та перевірку відповідності встановленим правилам стилю.

Таким чином, у межах роботи було реалізовано повноцінний frontend застосунок для взаємодії із соціальною платформою Stack-Fate. Реалізований функціонал забезпечує реєстрацію, авторизацію, керування профілем, роботу зі спільнотами та дописами, а також взаємодію з REST API. Використання Next.js, React, TypeScript, Tailwind CSS та Shadcn/UI дозволило створити сучасний, адаптивний та масштабований користувацький інтерфейс.

3.5 Тестування системи

Після завершення реалізації frontend та backend частини платформи Stack-Fate було проведено комплексне тестування системи. Основною метою тестування є перевірка коректності роботи функціональних модулів, виявлення можливих помилок та підтвердження відповідності реалізованого програмного забезпечення поставленим вимогам.

Тестування є важливим етапом життєвого циклу програмного забезпечення, оскільки дозволяє оцінити якість розробленого продукту, перевірити стабільність його роботи та забезпечити коректну взаємодію між окремими компонентами системи. У межах даного проєкту тестування проводилося як для серверної частини REST API, так і для клієнтського вебзастосунку.

Основними завданнями тестування були перевірка коректності роботи REST API, механізмів автентифікації та авторизації, профілю користувача, функціоналу спільнот, системи дописів, обробки помилок, адаптивності інтерфейсу та взаємодії між frontend і backend компонентами.

Для проведення тестування використовувалися різні програмні інструменти.

Таблиця 3.5

Інструменти тестування

Інструмент	Призначення
Swagger UI	Тестування REST API
Postman	Перевірка endpoint-ів
Google Chrome	Тестування frontend
Mozilla Firefox	Кросбраузерне тестування
DevTools	Перевірка адаптивності

Одним із перших етапів було тестування REST API. Для цього використовувалася автоматично згенерована документація Swagger UI, яка формується Django Ninja. За допомогою Swagger UI було перевірено роботу всіх основних endpoint-ів системи.

Під час тестування API перевірялися основні функціональні можливості системи.

Таблиця 3.6

Результати тестування REST API

Тестовий сценарій	Очікуваний результат	Результат
Реєстрація користувача	Створення акаунта	Успішно
Підтвердження пошти	Активація акаунта	Успішно
Авторизація	Отримання JWT-токенів	Успішно

Отримання профілю	Повернення даних користувача	Успішно
Редагування профілю	Оновлення інформації	Успішно
Створення спільноти	Створення нового запису	Успішно
Створення допису	Створення публікації	Успішно
Видалення допису	Видалення запису	Успішно

Усі тестові запити виконувалися з використанням як коректних, так і некоректних вхідних даних. Це дозволило перевірити роботу механізмів валідації та обробки помилок.

Особлива увага приділялася тестуванню системи автентифікації та авторизації.

Таблиця 3.7

Результати тестування авторизації

Сценарій	Очікуваний результат	Результат
Коректний логін і пароль	Успішний вхід	Успішно
Неправильний пароль	Відмова в доступі	Успішно
Відсутній JWT-токен	Відмова в доступі	Успішно
Недійсний JWT-токен	Відмова в доступі	Успішно
Оновлення access token	Отримання нового токена	Успішно

Результати тестування показали, що система коректно обмежує доступ до захищених ресурсів та забезпечує належний рівень безпеки.

Окремо проводилося тестування функціоналу реєстрації користувачів. Перевірялися сценарії успішної реєстрації, використання вже існуючої електронної пошти, некоректного формату електронної адреси та надсилання неповних даних. У результаті було підтверджено коректну роботу механізмів валідації та перевірки унікальності користувачів.

Наступним етапом стало тестування профілю користувача, роботи зі спільнотами та системою дописів. Перевірялися сценарії створення та

редагування профілю, створення спільнот, подання заявок на вступ, створення та редагування дописів.

Для перевірки взаємодії між frontend та backend частинами були протестовані основні користувацькі сценарії.

Таблиця 3.8

Тестування користувацьких сценаріїв

Сценарій	Результат
Реєстрація користувача	Успішно
Авторизація	Успішно
Перегляд профілю	Успішно
Редагування профілю	Успішно
Створення допису	Успішно
Перегляд стрічки дописів	Успішно
Перегляд спільнот	Успішно
Подання заявки на вступ	Успішно
Зміна пароля	Успішно

Усі сценарії були успішно виконані без виникнення критичних помилок. Frontend коректно отримував дані через REST API та відображав їх у користувацькому інтерфейсі.

Для перевірки адаптивності інтерфейсу використовувалися інструменти розробника браузера та різні розміри екранів.

Перевірка проводилася для персональних комп'ютерів, ноутбуків, планшетів та смартфонів. Результати показали, що інтерфейс коректно адаптується до різних роздільних здатностей екрана та зберігає функціональність на всіх типах пристроїв.

Окремо було проведено тестування обробки помилок.

Таблиця 3.9

Результати тестування обробки помилок

Тестовий випадок	Очікуваний результат	Результат
Неправильний пароль	401 Unauthorized	Успішно
Невалідний JWT-токен	401 Unauthorized	Успішно
Відсутність прав доступу	403 Forbidden	Успішно
Некоректні дані форми	400 Bad Request	Успішно
Неіснуючий ресурс	404 Not Found	Успішно

У всіх випадках система повертала відповідні HTTP-статуси та зрозумілі повідомлення про помилки, що підтверджує коректну реалізацію механізмів обробки виключних ситуацій.

Для додаткової перевірки endpoint-ів використовувався Postman.

Проведене тестування підтвердило працездатність та стабільність усіх основних компонентів платформи. REST API коректно обробляє запити, механізми авторизації забезпечують захист ресурсів, а frontend частина надає користувачам зручний інтерфейс для взаємодії із системою.

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи було досліджено сучасні підходи до створення соціальних платформ та проаналізовано особливості організації взаємодії учасників ІТ-спільнот у цифровому середовищі. Проведений аналіз існуючих рішень показав, що більшість популярних платформ орієнтовані на виконання окремих функцій, таких як професійний нетворкінг, обмін програмним кодом або комунікація між користувачами. Використання декількох незалежних сервісів ускладнює взаємодію між учасниками спільнот та знижує ефективність спільної роботи над проєктами.

У межах роботи було обґрунтовано необхідність створення єдиної соціальної платформи для ІТ-спільнот, яка поєднує можливості професійного спілкування, організації спільнот, публікації контенту та взаємодії між користувачами в одному інформаційному середовищі.

Під час виконання роботи було спроектовано архітектуру програмної системи на основі сучасного клієнт-серверного підходу із використанням технологій Next.js, TypeScript, Django Ninja та PostgreSQL. Розроблена архітектура забезпечує розділення відповідальності між окремими компонентами системи, підтримуваність програмного коду, можливість масштабування та подальшого розвитку проєкту.

Було виконано проєктування структури бази даних та визначено основні сутності системи, які забезпечують роботу користувачів, спільнот, заявок на вступ, дописів та мультимедійного контенту. Запропонована модель даних забезпечує цілісність інформації та дозволяє розширювати функціональні можливості платформи без суттєвих змін архітектури.

У процесі реалізації створено повноцінний REST API сервіс із підтримкою автентифікації та авторизації користувачів на основі JWT-токенів, підтвердження електронної пошти, керування профілем користувача, створення

спільнот, подання заявок на вступ та публікації дописів. Для забезпечення підтримуваності проєкту використано багат шарову архітектуру із розділенням на контролери, сервіси, репозиторії та селектори.

Також було реалізовано frontend частину системи, яка забезпечує зручну взаємодію користувачів із платформою через сучасний вебінтерфейс. Розроблений інтерфейс підтримує адаптивне відображення на різних типах пристроїв та забезпечує комфортну роботу користувачів із функціональними можливостями системи.

Проведене тестування підтвердило працездатність і стабільність усіх основних компонентів платформи. Результати тестування показали коректну роботу REST API, механізмів автентифікації та авторизації, функціоналу профілю користувача, спільнот і системи дописів. Також було підтверджено правильність обробки помилок та адаптивність користувацького інтерфейсу.

Практичним результатом роботи стала розроблена соціальна платформа Stack-Fate, яка може використовуватися як основа для створення професійного середовища взаємодії учасників ІТ-спільнот. Запропоноване рішення дозволяє централізувати основні процеси професійної комунікації та сприяє розвитку співпраці між розробниками, студентами, викладачами та представниками ІТ-галузі.

Отримані результати підтверджують доцільність використання сучасних підходів до проєктування та розробки складних інформаційних систем із застосуванням модульної архітектури та сучасних програмних технологій. Досвід проведення власних наукових досліджень у сфері інформаційних технологій засвідчує ефективність використання багаторівневих архітектурних рішень для забезпечення масштабованості, підтримуваності та подальшого розвитку програмних систем. Аналогічні принципи побудови складних програмних комплексів були досліджені у науковій роботі, присвяченій використанню ансамблевих моделей машинного навчання та технологій

пояснюваного штучного інтелекту, де було підтверджено важливість поєднання функціональної ефективності програмного рішення з його зрозумілістю та можливістю подальшого вдосконалення[48].

Перспективними напрямками подальшого розвитку системи є впровадження механізмів модерації спільнот, системи рекомендацій контенту, інструментів пошуку спеціалістів для роботодавців, функціоналу командної роботи над проєктами, системи особистих повідомлень та мобільного застосунку. Реалізація зазначених можливостей дозволить розширити функціональність платформи та підвищити її практичну цінність для користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Міністерство цифрової трансформації України. Міністерство цифрової трансформації України. URL: <https://thedigital.gov.ua/> (дата звернення: 31.05.2026).
2. Meta for Developers. Facebook Platform Overview. Огляд платформи Facebook. URL: <https://developers.facebook.com/docs/> (дата звернення: 31.05.2026).
3. LinkedIn About. About LinkedIn. Інформація про платформу LinkedIn. URL: <https://about.linkedin.com/> (дата звернення: 31.05.2026).
4. Discord About. About Discord. Інформація про платформу Discord. URL: <https://discord.com/company> (дата звернення: 31.05.2026).
5. Reddit Help Center. Reddit Help. Довідковий центр Reddit. URL: <https://support.reddithelp.com/> (дата звернення: 31.05.2026).
6. GitHub Features. GitHub Features. Функціональні можливості платформи GitHub. URL: <https://github.com/features> (дата звернення: 31.05.2026).
7. Stack Overflow About. About Stack Overflow. Інформація про платформу Stack Overflow. URL: <https://stackoverflow.co/company> (дата звернення: 31.05.2026).
8. IT Ukraine Association. Dnipro IT Ecosystem Research 2025: Strategic Growth in Times of Uncertainty. Дослідження ІТ-екосистеми Дніпра 2025: стратегічне зростання в умовах невизначеності. URL: <https://itukraine.org.ua/en/dnipro-it-ecosystem-research-2025-strategic-growth-in-times-of-uncertainty/> (дата звернення: 31.05.2026).
9. Савін Н.В., Редич О.В. Актуальність впровадження єдиної соціальної платформи для ІТ-спільнот: аналіз потреб та перспектив в українському середовищі : збірник тез доповідей науково-практичної конференції «Математичні методи та цифрові технології в економіці, сучасні проблеми математики». Ірпінь : ДПУ, 2026.

- 10.Савін Н.В., Редич О.В. Сучасна розробка REST API сервісу на фреймворці Django : збірник тез доповідей науково-практичної конференції «Інноваційні цифрові технології: від алгоритмів до безпечних систем». Ірпінь : ДПУ, 2026.
- 11.Martin R. C. Clean Code: A Handbook of Agile Software Craftsmanship. Чистий код: посібник зі створення якісного програмного забезпечення. Boston : Prentice Hall, 2008.
- 12.Martin R. C. Clean Architecture: A Craftsman's Guide to Software Structure and Design. Чиста архітектура: посібник зі структури та проектування програмного забезпечення. Boston : Prentice Hall, 2017.
- 13.Hunt A., Thomas D. The Pragmatic Programmer: Your Journey to Mastery. Прагматичний програміст: шлях до професійної майстерності. 20th Anniversary Edition. Boston : Addison-Wesley Professional, 2019.
- 14.Kleppmann M. Designing Data-Intensive Applications. Проектування застосунків для роботи з інтенсивними потоками даних. Sebastopol : O'Reilly Media, 2017.
- 15.Richardson L., Amundsen M. RESTful Web APIs. Проектування та реалізація REST API вебсервісів. Sebastopol : O'Reilly Media, 2013.
- 16.Silberschatz A., Korth H. F., Sudarshan S. Database System Concepts. Концепції систем баз даних. 7th ed. New York : McGraw-Hill Education, 2019.
- 17.Django Documentation. Документація фреймворку Django. URL: <https://docs.djangoproject.com/> (дата звернення: 31.05.2026).
- 18.Django Ninja Documentation. Документація фреймворку Django Ninja. URL: <https://django-ninja.dev/> (дата звернення: 31.05.2026).
- 19.PostgreSQL Documentation. Документація системи керування базами даних PostgreSQL. URL: <https://www.postgresql.org/docs/> (дата звернення: 31.05.2026).

20. Docker Documentation. Документація платформи контейнеризації Docker.
URL: <https://docs.docker.com/> (дата звернення: 31.05.2026).
21. Next.js Documentation. Документація фреймворку Next.js. URL:
<https://nextjs.org/docs> (дата звернення: 31.05.2026).
22. React Documentation. Документація бібліотеки React. URL:
<https://react.dev/> (дата звернення: 31.05.2026).
23. TypeScript Documentation. Документація мови програмування TypeScript.
URL: <https://www.typescriptlang.org/docs/> (дата звернення: 31.05.2026).
24. Tailwind CSS Documentation. Документація CSS-фреймворку Tailwind CSS. URL: <https://tailwindcss.com/docs> (дата звернення: 31.05.2026).
25. OpenAPI Specification. Специфікація стандарту OpenAPI. URL:
<https://swagger.io/specification/> (дата звернення: 31.05.2026).
26. JSON Web Tokens Introduction. Вступ до технології JSON Web Token. URL:
<https://jwt.io/introduction> (дата звернення: 31.05.2026).
27. OWASP Top 10 Web Application Security Risks. Десять найпоширеніших ризиків безпеки вебзастосунків за версією OWASP. URL: <https://owasp.org/> (дата звернення: 31.05.2026).
28. Python Documentation. Документація мови програмування Python. URL:
<https://docs.python.org/3/> (дата звернення: 31.05.2026).
29. Auth.js Documentation. Документація бібліотеки Auth.js. URL:
<https://authjs.dev/> (дата звернення: 31.05.2026).
30. Redis Documentation. Документація системи Redis. URL:
<https://redis.io/docs/> (дата звернення: 31.05.2026).
31. Pydantic Documentation. Документація бібліотеки Pydantic. URL:
<https://docs.pydantic.dev/> (дата звернення: 31.05.2026).
32. Armbian Documentation. Документація операційної системи Armbian.
URL: <https://docs.armbian.com/> (дата звернення: 31.05.2026).
33. Biome Documentation. Документація інструмента Biome. URL:
<https://biomejs.dev/> (дата звернення: 31.05.2026).

- 34.uv Documentation. Документація менеджера пакетів uv. URL:
<https://docs.astral.sh/uv/> (дата звернення: 31.05.2026).
- 35.Ruff Documentation. Документація лінера Ruff. URL:
<https://docs.astral.sh/ruff/> (дата звернення: 31.05.2026).
- 36.Shadcn/UI Documentation. Документація бібліотеки компонентів Shadcn/UI. URL: <https://ui.shadcn.com/> (дата звернення: 31.05.2026).
- 37.Cloudflare. What is DNS? Що таке DNS? URL:
<https://www.cloudflare.com/learning/dns/what-is-dns/> (дата звернення: 31.05.2026).
- 38.Cloudflare Tunnel Documentation. Документація Cloudflare Tunnel. URL:
<https://developers.cloudflare.com/cloudflare-one/networks/connectors/cloudflare-tunnel/> (дата звернення: 31.05.2026).
- 39.PEP 8 – Style Guide for Python Code. Настанова зі стилю коду Python. URL:
<https://peps.python.org/pep-0008/> (дата звернення: 31.05.2026).
- 40.JSON Specification. Специфікація формату JSON. URL: <https://jsonapi.org/> (дата звернення: 31.05.2026).
- 41.Newman S. Building Microservices. Створення мікросервісів. 2nd ed. Sebastopol : O'Reilly Media, 2021.
- 42.Mozilla Developer Network. HTTP Overview. Огляд протоколу HTTP. URL:
<https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview> (дата звернення: 31.05.2026).
- 43.Vincent W. S. Django for Beginners: Build Websites with Python and Django. Django для початківців: створення вебсайтів за допомогою Python та Django. 5th ed. Self-published, 2025.
- 44.Vincent W. S. Django for APIs: Build Web APIs with Python and Django. Django для API: створення вебінтерфейсів програмування на Python та Django. 4th ed. Self-published, 2024.

45. Vincent W. S. Django for Professionals: Production Websites with Python and Django. Django для професіоналів: розробка промислових вебзастосунків на Python та Django. 5th ed. Self-published, 2025.
46. Sinha S. Django 5 By Example: Build Powerful and Reliable Python Web Applications from Scratch. Django 5 на практиці: створення потужних і надійних вебзастосунків з нуля. 5th ed. Birmingham : Packt Publishing, 2024.
47. Martin R. C. Agile Software Development: Principles, Patterns, and Practices. Гнучка розробка програмного забезпечення: принципи, шаблони та практики. Upper Saddle River : Prentice Hall, 2002.
48. Редич О. В., Савін Н. В. Ансамблеве машинне навчання та технології SHAP у задачах кредитного скорингу. Наука і техніка сьогодні. 2026. № 4(58). С. 4309–4321. DOI: 10.52058/2786-6025-2026-4(58)-4309-4321