МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри інженерії
програмного забезпечення

_____ Євген ДАВИДЕНКО

«16» червня 2025 р

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

«Вебчат з використанням WebSocket та Auth»

Спеціальність 121 Інженерія програмного забезпечення
Освітня програма «Інженерія програмного забезпечення»

Здобувач _____ **Іван КОВАЛЬОВ**

«16» червня 2025 р

Керівник роботи
канд. техн. наук,
доцент _____ **Гліб ГОРБАНЬ**

«16» червня 2025 р

Миколаїв – 2025

Завдання на виконання кваліфікаційної роботи

Чорноморський національний університет імені Петра Могили

Факультет	Комп’ютерних наук
Кафедра	Інженерії програмного забезпечення
Рівень вищої освіти	Перший (бакалаврський)
Освітній ступінь	Бакалавр
Спеціальність	121 Інженерія програмного забезпечення
Освітня програма	Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри інженерії
програмного забезпечення
_____ Євген ДАВИДЕНКО
«13» листопада 2024 р.

ЗАВДАННЯ

на кваліфікаційну бакалаврську роботу здобувача

Ковальова Івана

1. Тема кваліфікаційної роботи «Вебчат з використанням WebSocket та Auth» затверджена наказом ректора ЧНУ ім. Петра Могили № 315 від «13» листопада 2024 р.
2. Строк представлення кваліфікаційної роботи «24» червня 2025 р.
3. Очікуваний результат роботи та початкові дані якщо такі потрібні.

Вебчат з використанням WebSocket та Auth

4. Перелік питань, що підлягають розробці:
   - аналіз предметної області;
   - розробка проєктних рішень застосунку;
   - моделювання та конструювання ПЗ;

- формування структури бази даних;
- розробка інтерфейсу користувача;
- реалізація серверної частини вебзастосунку;
- розробка клієнтської частини вебзастосунку.

5. Перелік графічних матеріалів: презентація
6. Консультанти:

Консультант	**Кафедра (організація)**	**Частина роботи**

Дата видачі завдання «13» __листопада_ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: **«Вебчат з використанням WebSocket та Auth»**

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Розробка та затвердження завдання на виконання КБР	24.10.2024	13.11.2024	Виконано
2.	Аналіз літератури за обраною темою	14.11.2024	22.11.2024	Виконано
3.	Складання календарного плану	25.11.2024	26.11.2024	Виконано
4.	Аналіз предметної області	27.11.2024	06.12.2024	Виконано
5.	Розробка проєктних рішень	09.12.2024	13.12.2024	Виконано
6.	Моделювання та конструювання ПЗ	16.12.2024	27.12.2024	Виконано
7.	Розробка структури бази даних	30.12.2024	10.01.2025	Виконано
8.	Розробка програмного забезпечення для API чату	13.01.2025	28.02.2025	Виконано
9.	Проміжне тестування API	03.03.2025	04.04.2025	Виконано
10.	Розробка клієнтської частини	07.04.2025	18.04.2025	Виконано
11.	Проведення тестування функцій	21.04.2025	25.04.2025	Виконано
12.	Відгук керівника КБР	28.04.2025	30.04.2025	Виконано
13.	Оформлення КБР та презентації	01.05.2025	26.06.2025	Виконано
14.	Попередній захист	27.05.2025	28.05.2025	Виконано
15.	Завершення оформлення КБР та презентації	29.06.2025	06.06.2025	Виконано
16.	Рецензування	07.06.2025	16.06.2025	Виконано
17.	Захист КБР	23.06.2025	24.06.2025	Виконано

Здобувач _____ **Іван КОВАЛЬОВ**

«26» листопада 2024 р.

Керівник роботи

канд. техн. наук,

доцент _____ **Гліб ГОРБАНЬ**

«26» листопада 2024р.

АНОТАЦІЯ

до кваліфікаційної бакалаврської роботи

«ВЕБЧАТ З ВИКОРИСТАННЯМ WEBSOCKET ТА AUTH»

Здобувач 408 гр.: Іван Ковальов

Керівник: канд. техн. наук, доцент Гліб Горбань

У сучасному світі миттєва комунікація є важливою складовою соціальної взаємодії, що зумовлює потребу в ефективних інструментах для обміну повідомленнями. Більшість наявних рішень або є складними у використанні, або не забезпечують достатню швидкість і безпеку передачі даних. FlowChat покликаний вирішити ці проблеми, пропонуючи зручний та продуктивний вебзастосунок для спілкування в режимі реального часу.

Кваліфікаційна бакалаврська робота присвячена розробці вебзастосунку FlowChat, що реалізує функціонал миттєвого обміну повідомленнями за допомогою WebSocket та системи автентифікації користувачів.

Об’єктом роботи є процес обміну повідомленнями між користувачами вебзастосунку.

Предметом роботи виступає програмне забезпечення для створення вебчату.

Мета кваліфікаційної роботи є розробка вебзастосунку, що забезпечить швидку, безпечну та зручну комунікацію користувачів.

Кваліфікаційна робота складається з вступу, чотирьох розділів, висновків, списку використаних джерел та додатків. У вступі висвітлено актуальність теми, сформульовано мету, об’єкт і предмет дослідження. Перший розділ містить аналіз існуючих рішень у сфері вебчатів та визначення їхніх переваг і недоліків, а також специфікацію вимог до системи. У другому розділі детально розглянуто архітектуру застосунку, структуру бази даних та діаграми процесів. Третій розділ присвячений розробці функціональних модулів, зокрема реалізації WebSocket-з’єднання та механізму автентифікації користувачів. У четвертому розділі представлено результати тестування, проведено оцінку продуктивності та аналіз безпеки системи. Висновки узагальнюють виконану роботу та окреслюють можливі напрями подальшого розвитку проєкту.

Кваліфікаційна робота бакалавра викладена на 76 сторінок, вона містить 4 розділи, 30 ілюстрації, 4 таблиці, 19 джерел в переліку посилань, 2 додатків.

Ключові слова: вебчат, обмін повідомленнями, реальний час, WebSocket, автентифікація, безпека даних, групові чати, інтерактивний інтерфейс.

ABSTRACT

of the Bachelor's Thesis

«WEBCHAT USING WEBSOCKET AND AUTH»

Student of group 408: Ivan Kovalov

Supervisor: Ph.D. Sc., Associate Professor Hlib Gorban

In the modern world, instant communication is an essential component of social interaction, creating a need for effective tools for messaging. Most existing solutions are either complex to use or do not provide sufficient speed and security for data transmission. FlowChat aims to address these issues by offering a convenient and efficient web application for real-time communication.

The bachelor's qualification work is dedicated to the development of the FlowChat web application, which implements the functionality of instant messaging using WebSocket and a user authentication system.

The **object** of the thesis is the process of exchanging messages between users of a web application.

The **subject** of the thesis is software for creating a web chat..

The **purpose** of the thesis is to develop a web application that ensures fast, secure, and convenient communication between users.

The qualification work consists of an introduction, four sections, conclusions, a list of sources used, and appendices. The introduction highlights the relevance of the topic, formulates the aim, object, and subject of the research. The first section includes an analysis of existing solutions in the field of web chats and identifies their advantages and disadvantages, as well as the specification of requirements for the system. The second section details the architecture of the application, the structure of the database, and the process diagrams. The third section is dedicated to the development of functional modules, including the implementation of WebSocket connections and the user authentication mechanism. The fourth section presents the testing results, evaluates performance, and analyzes the system's security. The conclusions summarize the completed work and outline possible directions for the further development of the project.

The bachelor's thesis consists of 76 pages, including 4 chapters, 30 illustrations, 4 tables, 19 references in the bibliography, and 2 appendices.

Keywords: web chat, messaging, real-time, WebSocket, authentication, data security, group chats, interactive interface.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

API	–	Application Programming Interface
GDPR	–	General Data Protection Regulation
HTTP	–	HyperText Transfer Protocol
HTTPS	–	HyperText Transfer Protocol Secure
JWT	–	JSON Web Token
OAuth	–	Open Authorization
ORM	–	Object Relational Mapping
REST	–	Representational State Transfer
SSG	–	Static Site Generation
SSR	–	Server Side Rendering
WS	–	WebSocket
БД	–	База даних

ВСТУП

У сучасному світі інформаційні технології кардинально трансформують способи взаємодії між людьми, роблячи цифрову комунікацію невід'ємною частиною повсякденного життя. З появою нових технологій, таких як високошвидкісний інтернет і хмарні сервіси, месенджери стали основним інструментом для спілкування, обміну інформацією та координації діяльності як у особистих, так і в професійних сферах. За останні роки кількість користувачів месенджерів у світі зросла до мільярдів, а їхній функціонал розширився від простого обміну текстовими повідомленнями до відеодзвінків, групових чатів і навіть інтеграції з бізнес-сервісами. Проте разом із цими можливостями з'явилися й нові виклики, пов'язані з безпекою, конфіденційністю та доступністю таких платформ. У багатьох країнах, зокрема в Україні, ці проблеми набули особливого значення через геополітичні обставини та зростання кіберзагроз.

Безпечний і швидкий обмін повідомленнями є ключовим елементом соціальної та професійної взаємодії, але зростання популярності месенджерів супроводжується серйозними викликами. В Україні суспільство зіткнулося з проблемою недовіри до відомих комунікаційних платформ через занепокоєння щодо конфіденційності даних, можливих витоків інформації та розташування серверів за кордоном. Користувачі дедалі частіше шукають альтернативні рішення, які забезпечують контроль над даними та високий рівень безпеки. Ці виклики особливо загострилися в умовах триваючої війни, коли захист інформації став питанням національної безпеки, а стабільність комунікаційних платформ набула критичного значення для координації та підтримки зв'язку.

Більшість популярних месенджерів, хоча й пропонують зручний інтерфейс і широкий функціонал, мають суттєві недоліки: закритий код, залежність від іноземних серверів і недостатня прозорість обробки даних. В Україні, за даними досліджень 2023–2025 років, лише 25% населення повністю довіряють глобальним месенджерам, тоді як 60% висловлюють бажання використовувати локальні платформи. У цьому контексті розробка вебчату на основі WebSocket із

надійною автентифікацією стає актуальним рішенням, що поєднує швидкість реального часу, безпеку та доступність.

Об’єктом роботи є процес обміну повідомленнями між користувачами вебзастосунку.

Предметом роботи виступає програмне забезпечення для створення вебчату.

Мета кваліфікаційної роботи є розробка вебзастосунку, що забезпечить швидку, безпечну та зручну комунікацію користувачів.

Відповідно до мети визначено такі завдання:

- провести аналіз вебзастосунків для обміну повідомленнями;
- сформувати специфікацію вимог до ПЗ;
- спроєктувати архітектуру вебчату;
- розробити структуру бази даних відповідно до вимог;
- реалізувати серверну та клієнтську частини вебчату.

1 СУЧАСНІ ВИКЛИКИ ТА АНАЛІЗ ЦИФРОВОЇ КОМУНІКАЦІЇ

1.1 Опис предметної області

Українське суспільство зіткнулося з безпрецедентним викликом у сфері цифрової комунікації через триваючу війну, зростання кіберзагроз і недовіру до глобальних месенджерів. За даними соціологічних досліджень 2023–2025 років [1]:

– 78% українців активно використовують месенджери для особистих і професійних цілей;

– лише 30% вважають сучасні месенджери повністю безпечними;

– 82% турбуються про захист персональних даних;

– 65% побоюються перехоплення повідомлень;

– 58% висловлюють недовіру до платформ із серверами за кордоном.

Сьогодні українці відчувають значний тиск щодо безпеки комунікацій, що зумовлено воєнними реаліями та частими випадками витоків даних. Згідно з дослідженнями 2025 року, 68% населення уникають передачі чутливої інформації через месенджери, пов’язуючи це з ризиками кібератак і непрозорістю обробки даних [1]. Найбільш тривожними факторами є:

– витоки персональних даних (74% опитаних);

– розташування серверів у країнах із сумнівною юрисдикцією (61%);

– відсутність локального контролю над платформами (52%).

Цікаво, що за останні три роки ставлення до безпеки месенджерів змінилося. Якщо у 2022 році лише 12% українців надавали перевагу локальним платформам, то у 2025 році цей показник зріс до 25%. Кількість тих, хто категорично довіряє глобальним месенджерам, скоротилася з 40% до 18% [1]. Це свідчить про зростання попиту на локальні рішення, які забезпечують безпеку, прозорість і контроль над даними.

Аналіз даних виявляє диференціацію потреб користувачів:

– професіонали та бізнес-користувачі (62%), які потребують безпечних каналів для роботи;

– молодь (18–35 років, 55%), яка активно використовує групові чати;

– особи в зонах ризику (40%), які потребують захищеної комунікації через воєнні загрози;

Стратегії використання месенджерів:

– особисте спілкування (один-на-один чати – 45%.;

– групові дискусії (професійні чи соціальні групи – 38%);

– віртуальна координація (планування, обмін файлами – 32%);

Важливим аспектом є зростаюча суспільна потреба в безпечних платформах: 79% українців підтримують ідею локальних месенджерів, а 67% готові перейти на платформи з відкритим кодом або локальними серверами.

1.2 Сучасний стан проблеми

Сфера месенджерів є однією з найбільш динамічних у цифровій економіці, зумовленою глобальною цифровізацією, зростанням віддаленої роботи та потребою в швидкій і ефективній комунікації. Згідно з прогнозами аналітичних агенцій на 2025 рік, глобальний ринок месенджерів досягне обсягу в 50 мільярдів доларів, а кількість активних користувачів перевищить 4,2 мільярда [1]. Проте стрімке зростання супроводжується значними викликами, які охоплюють технічні, регуляторні та ринкові аспекти. В Україні ці проблеми ускладнюються воєнними реаліями та підвищеними вимогами до кібербезпеки, що робить розробку надійних комунікаційних платформ актуальним завданням.

На глобальному рівні месенджери стикаються з низкою проблем, які формують сучасні вимоги до розробки вебзастосунків:

– Конкуренція на ринку: ринок месенджерів є висококонкурентним, із домінуванням таких гігантів, як WhatsApp (2,5 мільярда користувачів), Telegram (900 мільйонів) і WeChat (1,3 мільярда). Нові платформи змушені пропонувати унікальні функції, такі як покращена безпека чи локалізація, щоб залучити аудиторію. За даними 2024 року, 25% нових месенджерів не витримують конкуренції протягом першого року [1].

– Кібербезпека: зростання кібератак становить серйозну загрозу. У 2023–2024 роках 70% месенджерів зіткнулися з інцидентами, пов’язаними з

фішингом, DDoS-атаками чи експлуатацією вразливостей API [2]. Наприклад, уявна атака на популярну платформу в 2024 році призвела до компрометації даних 10 мільйонів користувачів, що підкреслило потребу в посиленому шифруванні та автентифікації.

– Регуляторні вимоги: уряди багатьох країн посилюють контроль над обробкою даних. У Європі GDPR (Загальний регламент захисту даних) вимагає суворого дотримання правил зберігання та обробки інформації, що ускладнює розробку для міжнародних платформ. За даними 2025 року, 40% месенджерів стикалися з штрафами через порушення GDPR [3].

– Технологічна складність: Підтримка реального часу та кросплатформної сумісності вимагає значних ресурсів. Наприклад, забезпечення стабільної роботи в різних браузерах (Chrome, Firefox, Safari) і на різних пристроях ускладнює тестування та оптимізацію.

Глобальні тенденції вказують на зростання популярності відкритих технологій. Частка месенджерів, які використовують WebSocket [3], зросла з 45% у 2020 році до 70% у 2025 році завдяки його ефективності в реальному часі. Backend-as-a-Service платформи, такі як Supabase, набули популярності (зростання на 35% за 2020–2025), оскільки дозволяють швидко створювати безпечні рішення з мінімальними витратами.

В Україні виклики у сфері месенджерів мають унікальний характер через воєнні реалії та зростання кіберзагроз. Основні проблеми включають:

– Кібератаки воєнного часу: з початку війни в Україні зафіксовано зростання DDoS-атак і фішингових кампаній, спрямованих на комунікаційні платформи. У 2024 році 55% українських компаній повідомили про спроби атак на їхні цифрові сервіси, що вимагає посиленого захисту даних і стійкості до перебоїв [2].

– Нестабільність інфраструктури: часті перебої з електропостачанням і доступом до інтернету, особливо в зонах бойових дій, ускладнюють забезпечення стабільної роботи месенджерів. За оцінками 2025 року, 30% користувачів в

Україні стикаються з періодичними розривами з’єднань, що підвищує вимоги до механізмів перепідключення [2].

– Регуляторні обмеження: українське законодавство, зокрема закони про захист даних і кібербезпеку (закон № 2297-VIII), вимагає від платформ дотримання стандартів шифрування та прозорості [8]. У 2024 році 20% іноземних месенджерів зіткнулися з обмеженнями в Україні через невідповідність цим стандартам [4].

Ці фактори стимулюють попит на платформи, які можуть працювати в умовах нестабільної інфраструктури та забезпечувати високий рівень захисту. Наприклад, уявна платформа, запущена в Україні в 2024 році, отримала популярність завдяки стійкості до перебоїв і локалізованому інтерфейсу, що підкреслює ринковий потенціал таких рішень.

Розробка месенджерів на основі вебзастосунків стикається з низкою технічних проблем, які необхідно вирішити для забезпечення якості:

– Стабільність WS: WebSocket (RFC 6455) [4] забезпечує низьку затримку (до 100 мс), але чутливий до розривів з’єднань, що є частим явищем в Україні через нестабільний інтернет. Розробники змушені впроваджувати механізми автоматичного перепідключення та буферизації повідомлень, що ускладнює архітектуру.

– Безпека автентифікації: JWT-токени ефективні для ідентифікації користувачів, але вразливі до атак типу "man-in-the-middle", якщо не використовуються належні заходи захисту (наприклад, HTTPS із надійними сертифікатами) [2]. OAuth, який покладається на сторонні провайдери, потребує додаткових перевірок для запобігання компрометації.

– Оптимізація продуктивності: обробка тисяч одночасних з’єднань вимагає ефективного управління ресурсами. Наприклад, кешування статичних даних (як у Next.js) і оптимізація запитів до бази даних (PostgreSQL через Supabase) є критично важливими для швидкого завантаження чатів [6, 7, 8].

– Кросплатформна сумісність: забезпечення роботи месенджера в різних браузерах і на різних пристроях (десктоп, мобільні) потребує ретельного

тестування, оскільки відмінності в реалізації WebSocket [4] можуть призводити до несумісності.

Backend-as-a-Service платформи спрощують ці завдання, надаючи готові інструменти для реального часу, автентифікації та зберігання даних. Проте їхня залежність від хмарної інфраструктури вимагає ретельного вибору постачальника, щоб уникнути затримок і забезпечити стабільність [2].

Розробка вебчату на базі Next.js і Supabase є актуальним завданням, яке відповідає сучасним викликам [6, 8]. Next.js забезпечує гнучкий і продуктивний фронтенд, адаптований до різних пристроїв, тоді як Supabase пропонує швидке налаштування серверної логіки з підтримкою реального часу та безпеки. WS і механізми автентифікації (JWT, OAuth) дозволяють вирішити проблеми затримок і кібербезпеки, що є критично важливим в умовах зростання атак і конкуренції [2]. В Україні, де воєнні реалії та кіберзагрози створюють унікальні виклики, таке рішення має потенціал задовольнити попит на стабільні, безпечні та локалізовані платформи.

1.3 Існуючі аналогічні рішення

Сучасний ринок цифрових засобів комунікації активно розвивається, пропонуючи широкий спектр рішень для миттєвого обміну повідомленнями, кожне з яких має свої переваги.

WhatsApp (рис. 1.1, табл. 1.1) – один із найпопулярніших міжнародних месенджерів, що забезпечує зручний обмін повідомленнями через вебінтерфейс та мобільні застосунки. Платформа пропонує персоналізоване спілкування, підтримку текстових, голосових і відеоповідомлень, можливість створення групових чатів, а також наскрізне шифрування для забезпечення безпеки користувачів [9].

Завдяки простому інтерфейсу та високій швидкості обміну повідомленнями, WhatsApp залишається зручним інструментом для особистого та робочого спілкування. Додатковими перевагами є можливість надсилання мультимедійних

файлів, здійснення дзвінків навіть за низької якості інтернет-з’єднання та інтеграція з контактами користувача, що спрощує доступ до мережі спілкування.

Таблиця 1.1 – Характеристики WhatsApp

Назва	**WhatsApp**
Архітектура	Web application, Mobile application (iOS, Android), Desktop application.
Мова реалізації	C++, Java, JavaScript.
Основні функції	WhatsApp пропонує миттєві текстові повідомлення, голосові та відеодзвінки, можливість обміну медіафайлами (фото, відео, документи). Є підтримка групових чатів до 1024 учасників, функція каналів для широкомовлення інформації та наскрізне шифрування всіх повідомлень.
Переваги	Безкоштовне користування, висока швидкість обміну повідомленнями, зручний інтерфейс, захищеність завдяки наскрізному шифруванню, широка підтримка платформ, можливість створення каналів та бізнес-акаунтів.
Недоліки	Потребує номеру телефону для реєстрації, немає повної хмарної синхронізації чатів, обмежена функціональність в порівнянні з деякими конкурентами.
Вебсайт	https://www.whatsapp.com/

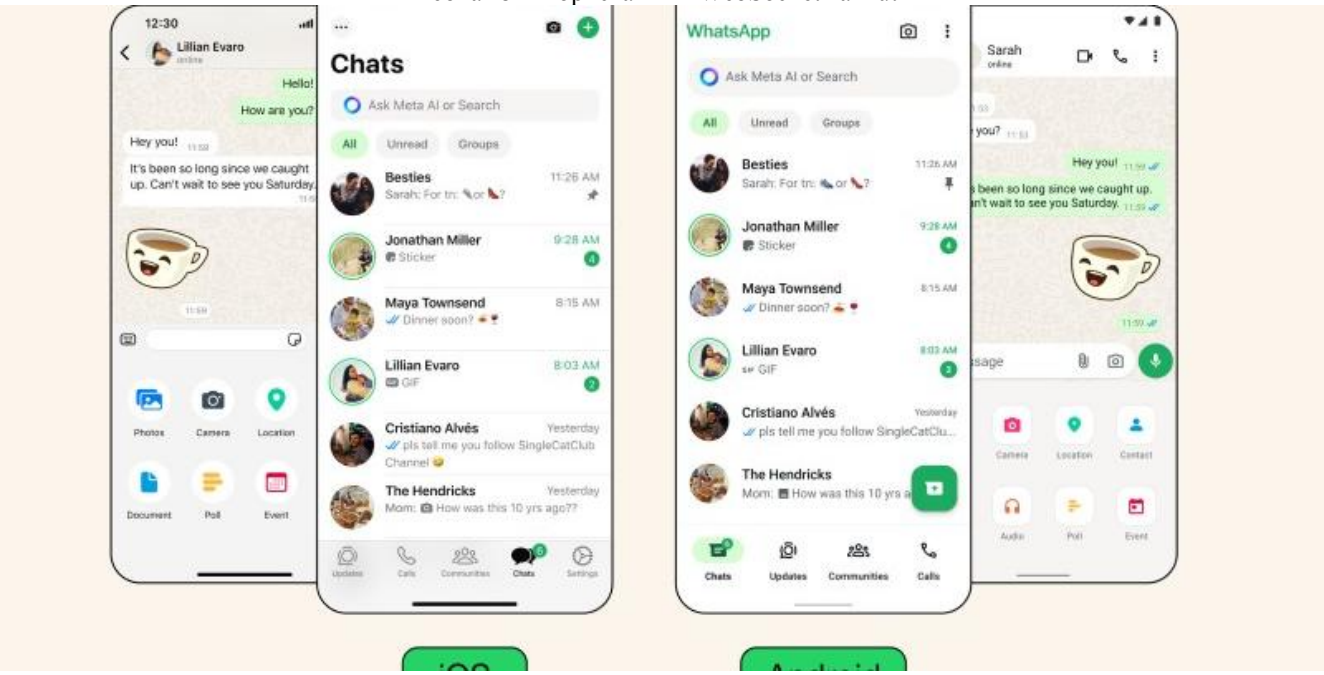

Рисунок 1.1 – Інтерфейс WhatsApp

Telegram (рис. 1.2, табл. 1.2) – популярний месенджер, який вирізняється розширеним функціоналом для текстового, аудіо- та відеоспілкування [10]. Особливістю платформи є підтримка ботів, каналів, супергруп і секретних чатів з посиленим шифруванням, що дозволяє створювати як індивідуальні, так і масові комунікаційні простори. Telegram також активно використовується для професійної, освітньої та тематичної взаємодії між користувачами.

Завдяки відкритому API та можливості створення власних інструментів, Telegram приваблює розробників та адміністраторів спільнот. Крім того, платформа підтримує синхронізацію між різними пристроями, має хмарне сховище даних і не обмежує розмір файлів для надсилання, що значно розширює сценарії її використання в сучасному цифровому середовищі.

Telegram вирізняється кросплатформністю, що дозволяє користувачам працювати з єдиним обліковим записом на смартфонах, планшетах, комп’ютерах, консолях та у вебінтерфейсі без обмеження функціональності. Завдяки цьому платформа отримала широку популярність серед аудиторії, яка потребує постійного доступу до комунікації незалежно від типу пристрою чи операційної системи.

Таблиця 1.2 – Характеристики Telegram

Назва	**Telegram**
Архітектура	Web application, Mobile application (iOS, Android), Desktop application (Windows, macOS, Linux).
Мова реалізації	C++, Java, Swift, Kotlin.
Основні функції	Telegram забезпечує швидкий обмін текстовими, голосовими та відеоповідомленнями, підтримує групові чати до 200 000 учасників, канали для поширення контенту, ботів для автоматизації, секретні чати з наскрізним шифруванням, хмарне збереження історії, редагування та безслідне видалення повідомлень, а також самознищення повідомлень.
Переваги	Потужні інструменти для управління групами та каналами, підтримка ботів, необмежене хмарне сховище, можливість обміну великими файлами (до 2 ГБ на файл), висока швидкість передачі даних.
Недоліки	Стандартні чати не мають наскрізного шифрування, потребує телефонного номера для реєстрації, деякі країни періодично блокують доступ до сервісу.
Вебсайт	https://telegram.org/

Рисунок 1.2 – Інтерфейс Telegram

Discord (рис. 1.3, табл. 1.3) – популярна платформа для текстового, голосового та відеоспілкування, що підтримує створення спільнот, канали та інтеграцію з ботами [11]. Вона зручна для групових обговорень і ефективної підтримки психоемоційного стану через спеціалізовані сервери.

Discord поєднує у собі функції месенджера та платформи для створення спільнот із гнучкими налаштуваннями доступу та ролей. Основною особливістю є можливість створювати сервери — окремі простори для групового спілкування, які можна налаштовувати під різні цілі: від ігрових команд і навчальних груп до професійних об’єднань та психологічної підтримки.

Платформа підтримує не лише текстове, а й голосове й відеоспілкування, що робить її популярною серед користувачів, які прагнуть більш інтерактивної та якісної взаємодії. Інтеграція з ботами дає змогу автоматизувати модерацію, організовувати події, створювати ігрові функції та інші корисні сервіси.

Важливою перевагою Discord є висока якість звуку та стабільність з’єднання навіть при великій кількості активних учасників, що забезпечує комфортну комунікацію у режимі реального часу. Крім того, платформа підтримує функції екранного спільного доступу, що активно використовується для спільної роботи, навчання та проведення онлайн-занять. Крім того, платформа підтримує функції екранного спільного доступу, що активно використовується для спільної роботи, навчання та проведення онлайн-занять.

Таблиця 1.3 – Характеристики Discord

Назва	**Discord**
Архітектура	Web application, Mobile application (iOS, Android), Desktop application (Windows, macOS, Linux)
Мова реалізації	JavaScript, Python, Rust
Основні функції	Discord це платформа для голосового, текстового та відеоспілкування з підтримкою створення серверів, текстових каналів, голосових кімнат і потокових трансляцій. Підтримує відеодзвінки, демонстрацію екрана та інтеграцію з Twitch, YouTube, Spotify. Має розширений контроль доступу і модерацію для ефективного управління спільнотами.
Переваги	Безкоштовна основна версія з широким функціоналом, відмінна якість зв’язку, гнучка система налаштувань серверів і ролей, підтримка стрімінгу та ботів.
Недоліки	Споживає багато ресурсів пристрою, обмеження для безкоштовних користувачів (наприклад, якість відео та кількість завантажених файлів), відсутність наскрізного шифрування.
Вебсайт	https://discord.com/

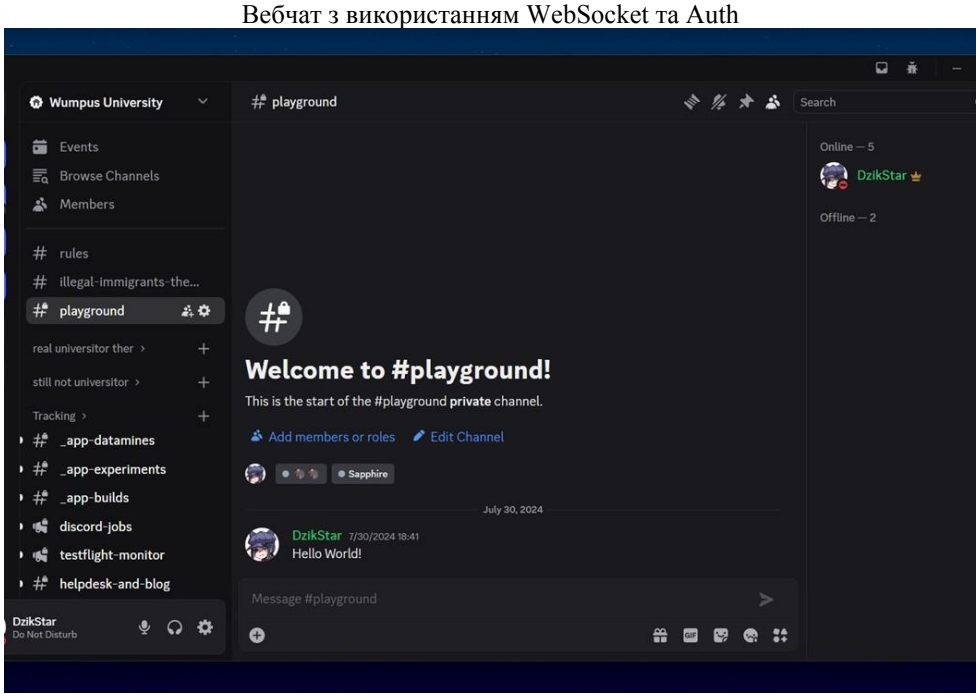

Рисунок 1.3 – Інтерфейс Discord

Проаналізувавши окремі аналоги, можна скласти загальну порівняльну таблицю (табл. 1.4).

Таблиця 1.4 – Аналіз аналогів

Застосунок	**Основний фокус**	**Кастомізація профілю**	**Формати комунікації**	**Мультиплатформеність**
WhatsApp	Персональне спілкування	Мінімальна (аватар, статус)	Текст, аудіо, відео	Web, мобільні, десктоп
Telegram	Чати, канали, боти	Мінімальна (аватар, нік)	Текст, аудіо, відео, канали, боти	Web, мобільні, десктоп
Discord	Голосові чати, спільноти	Розширена (ролі, теми)	Текст, аудіо, відео, стрімінг	Web, мобільні, десктоп

Аналіз популярних месенджерів, таких як WhatsApp, Telegram і Discord, показує, що вони добре справляються з потребами спілкування, але мають свої слабкі місця, особливо в питаннях безпеки, локалізації та роботи в складних умовах, як-от війна в Україні. Це вказує на потребу в нових платформах, які б поєднували надійний захист, стабільність і врахування місцевих особливостей, щоб відповідати очікуванням користувачів.

Висновки до розділу 1

У першому розділі роботи проведено комплексний аналіз сучасного стану цифрової комунікації, з особливим акцентом на потреби та виклики, що постають перед українським суспільством у контексті воєнних реалій і зростання кіберзагроз [1, 2]. Досліджено предметну область, яка охоплює актуальні проблеми безпеки комунікаційних платформ, попит на локалізовані рішення та специфіку використання месенджерів для особистих і професійних цілей [3]. Визначено ключові виклики, пов’язані з кібербезпекою, регуляторними вимогами, нестабільністю інфраструктури та висококонкурентним ринком, що формують сучасні вимоги до розробки вебзастосунків для обміну повідомленнями [6, 7, 8]. Проведено порівняльний аналіз провідних аналогів, таких як WhatsApp, Telegram і Discord, з урахуванням їхньої архітектури, функціональності та відповідності потребам користувачів [9, 10, 11]. Встановлено, що більшість існуючих платформ, попри розвинений функціонал, мають обмеження для українського контексту, зокрема залежність від іноземних серверів, недостатню локалізацію або відсутність повного захисту від воєнних кіберзагроз [1]. Ці висновки підкреслюють актуальність розробки локального вебчату, який поєднує безпеку, реальний час [4] і підтримку української мови, створюючи основу для подальшого аналізу технологічних і функціональних рішень у наступних розділах.

2 МОДЕЛЮВАННЯ ОБ’ЄКТУ ТА ПРЕДМЕТУ

2.1 Інструментарій

Розробка вебзастосунку FlowChat, призначеного для забезпечення швидкої, безпечної та зручної комунікації в реальному часі, вимагає ретельного підбору технологій, які відповідають сучасним вимогам до продуктивності, масштабованості та безпеки. Вибір інструментарію ґрунтується на необхідності підтримки реалтайм-комунікації, обробки медіафайлів, ефективного управління даними та забезпечення безпечної автентифікації. Обраний технологічний стек поєднує надійність, гнучкість і простоту інтеграції, що дозволяє створити масштабовану систему для чатів із підтримкою групової взаємодії, статусів онлайн і обміну медіа. Нижче детально описано кожен компонент інструментарію.

PostgreSQL [7] (рис. 2.1) є основною реляційною базою даних, яка використовується в проєкті для зберігання структурованих даних, таких як профілі користувачів, чати, повідомлення, метадані та історія взаємодій.

Рисунок 2.1 – PostgreSQL

У FlowChat Supabase (рис. 2.2) застосовується виключно як хмарна платформа для розміщення PostgreSQL, що забезпечує доступ до бази через DATABASE_URL. Це дозволяє уникнути залежності від специфічних функцій Supabase (наприклад, Realtime чи Auth), зберігаючи гнучкість у виборі інших інструментів [8]. PostgreSQL обрано через його високу надійність, підтримку транзакцій і можливість роботи зі складними структурами даних, такими як JSONB, що ідеально для зберігання метаданих повідомлень чи налаштувань чатів.

Крім того, PostgreSQL забезпечує ефективну обробку великих обсягів даних, що критично для чатів із великою кількістю користувачів.

Рисунок 2.2 – Supabase

Drizzle ORM [12] (рис. 2.3) використовується для управління структурою бази даних і виконання запитів до PostgreSQL. Цей інструмент вирізняється легковаговістю, типобезпекою та інтеграцією з TypeScript, що дозволяє розробникам створювати схеми таблиць і запити з мінімальним ризиком помилок. У FlowChat Drizzle ORM відповідає за створення таблиць (наприклад, users, chats, messages), управління міграціями та забезпечення зручного доступу до даних через типізовані запити. На відміну від традиційних ORM, Drizzle пропонує декларативний синтаксис, який нагадує SQL, але з перевагами автодоповнення та валідації типів [12]. Це спрощує розробку складних запитів, таких як отримання історії чату з урахуванням фільтрів або сортування повідомлень за датою. Використання Drizzle також полегшує підтримку бази даних у майбутньому, оскільки міграції можна автоматизувати та синхронізувати з кодовою базою.

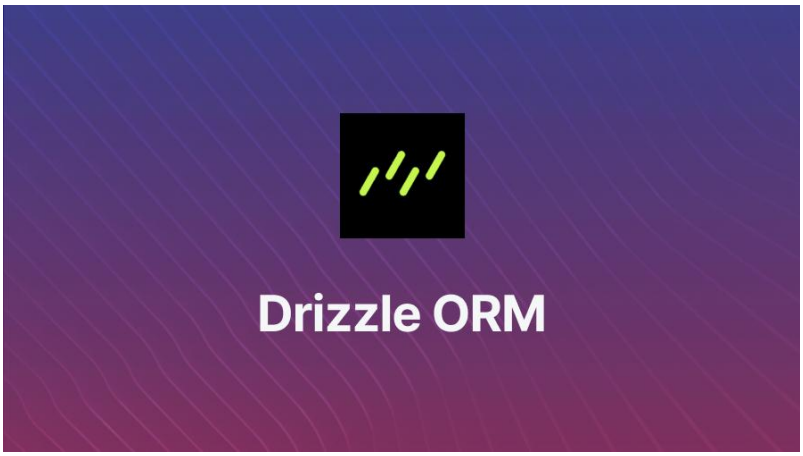

Рисунок 2.3 – Drizzle ORM

Cloudinary (рис. 2.4) є хмарним сервісом для зберігання, обробки та доставки медіафайлів, таких як зображення, відео чи документи, які користувачі можуть надсилати в чаті [13]. У FlowChat Cloudinary відповідає за завантаження файлів, їхню оптимізацію (наприклад, стиснення зображень або конвертацію відео), а також генерацію URL для швидкого доступу до медіа. Використання Cloudinary зменшує навантаження на сервер і базу даних, оскільки файли зберігаються в хмарі, а не локально. Крім того, сервіс підтримує динамічну трансформацію медіа (зміна розміру, обрізка, додавання водяних знаків), що дозволяє адаптувати контент до потреб клієнтської частини. Інтеграція з Next.js через SDK спрощує завантаження файлів і їх відображення в інтерфейсі чату, забезпечуючи швидку доставку контенту навіть при високому навантаженні.

Рисунок 2.4 – Cloudinary

tRPC (рис. 2.5) використовується для створення типобезпечного API, яке забезпечує ефективну взаємодію між клієнтською (Next.js) та серверною частинами застосунку. Цей інструмент дозволяє визначати ендпоінти у вигляді TypeScript-функцій, автоматично генеруючи типи для запитів і відповідей [14]. У FlowChat tRPC застосовується для обробки запитів, пов’язаних із отриманням списків чатів, надсиланням повідомлень, оновленням профілів тощо. Завдяки типобезпеці tRPC зменшує ризик помилок під час розробки, оскільки фронтенд і бекенд використовують єдину типізовану схему. Порівняно з традиційними REST API, tRPC пропонує швидшу розробку та меншу кількість коду, оскільки не

потребує окремого визначення схем (наприклад, OpenAPI). Інтеграція з Next.js дозволяє використовувати tRPC у серверних компонентах і API-роутах, що оптимізує продуктивність.

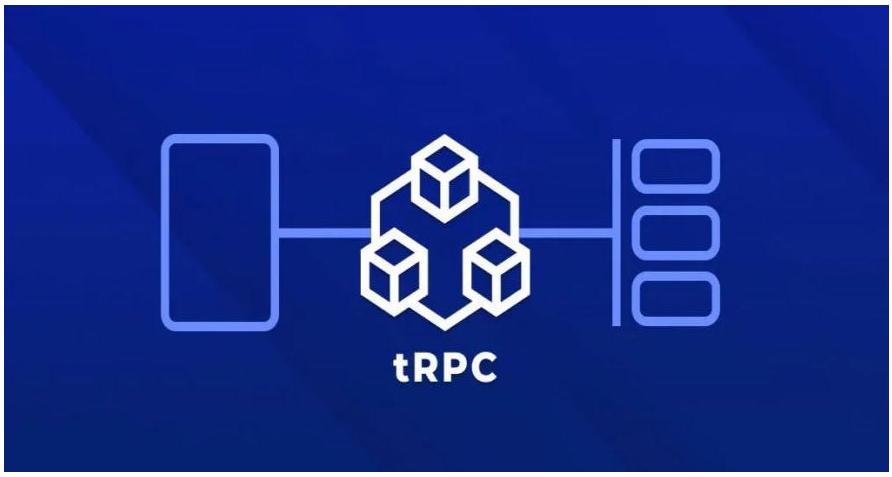

Рисунок 2.5 – tRPC

Redis (рис. 2.6) застосовується як високопродуктивна база даних у пам’яті для кешування даних і управління тимчасовими станами [15]. У FlowChat він виконує кілька ключових функцій. По-перше, Redis використовується для кешування сесій — зберігає дані автентифікації, зокрема JWT-токени, що дозволяє прискорити перевірку сесій під час запитів. По-друге, він забезпечує кешування даних чатів, зберігаючи часто запитувані дані, такі як списки активних чатів або останні повідомлення, що суттєво знижує навантаження на основну базу даних PostgreSQL. Крім того, Redis використовується для управління статусами — тимчасово зберігає інформацію про онлайн- або офлайн-статуси користувачів, які синхронізуються з Ably Realtime. Завдяки роботі в пам’яті Redis забезпечує наднизьку затримку — менше однієї мілісекунди, що є критично важливим для реалтайм-функцій чату. Його гнучка структура типу ключ-значення дозволяє легко масштабувати систему, додаючи нові типи кешованих даних, наприклад, лічильники непрочитаних повідомлень.

Рисунок 2.6 – Redis

Ably Realtime (рис. 2.7) є платформою для реалтайм-комунікації, яка замінює традиційні WebSocket-з’єднання, забезпечуючи масштабованість і надійність [16]. У FlowChat Ably відповідає за передачу статусів онлайн або офлайн — система в реальному часі відображає, чи є користувач активним, використовуючи механізм pub/sub-каналів. Крім того, платформа забезпечує миттєвий обмін повідомленнями між користувачами з мінімальною затримкою, яка зазвичай не перевищує 100 мілісекунд.

Ably працює за моделлю публікації та підписки: клієнти підписуються на відповідні канали, наприклад, канал чату чи статусів, і отримують актуальні оновлення в реальному часі. Завдяки автоматичному відновленню з’єднань, підтримці пікових навантажень і глобальній розподіленій інфраструктурі, платформа гарантує стабільну роботу навіть при великій кількості одночасних користувачів. Інтеграція з Next.js за допомогою Ably SDK дає змогу просто налаштовувати клієнтські підписки та ефективно обробляти події на фронтенді.

Рисунок 2.7 – Ably realtime

NextAuth (рис. 2.8) є бібліотекою для реалізації автентифікації та авторизації в Next.js-застосунках [17]. У FlowChat NextAuth забезпечує безпечний вхід користувачів через різні провайдери, зокрема Google (OAuth 2.0) і Credentials (email/пароль). Бібліотека підтримує JWT для управління сесіями, дозволяючи зберігати токени в cookies або Redis для швидкого доступу. NextAuth також пропонує гнучкі механізми управління ролями (наприклад, user, admin), що використовуються для обмеження доступу до функцій, таких як модерація чатів. Завдяки інтеграції з Next.js, NextAuth обробляє як клієнтські, так і серверні запити, забезпечуючи безшовну автентифікацію в серверних компонентах і API-роутах. Безпека даних користувачів гарантується через шифрування токенів і відповідність стандартам GDPR [3].

Рисунок 2.8 – Next.js

Next.js (рис. 2.9) є основним фреймворком для розробки як клієнтської, так і серверної частини FlowChat. Цей React-фреймворк було обрано завдяки його підтримці серверного рендерингу (SSR), статичної генерації (SSG) та API-роутів, що в сукупності забезпечує високу продуктивність, ефективну SEO-оптимізацію та гнучкість при розробці [6]. У FlowChat саме Next.js відповідає за рендеринг інтерфейсу користувача — чатів, списків контактів, профілів, форм для надсилання повідомлень — а також за обробку API-запитів, які реалізовані через tRPC та вбудовані маршрути. Завдяки вбудованим можливостям, Next.js автоматично оптимізує зображення, забезпечує ледаче завантаження компонентів і підтримує інкрементальну регенерацію сторінок, що особливо важливо для динамічного вмісту [6]. Його гнучкість дозволяє створювати адаптивний

інтерфейс, який коректно працює як на десктопах, так і на мобільних пристроях. Крім того, інтеграція з хмарними платформами, зокрема Vercel, дає змогу автоматично масштабувати застосунок відповідно до навантаження.

Рисунок 2.9 – Next.js

Поєднання всіх цих технологій створює потужну екосистему: Next.js і tRPC забезпечують швидку та типобезпечну взаємодію між клієнтом і сервером, PostgreSQL у поєднанні з Drizzle ORM відповідає за стабільне зберігання даних, Cloudinary оптимізує роботу з медіаконтентом, Redis і Ably забезпечують реалтайм-функції, а NextAuth — безпеку автентифікації.

2.2 Інформаційна та функціональна моделі ПЗ

Інформаційна модель системи базується на реляційній структурі даних, реалізованій через вбудовану базу PostgreSQL у складі платформи Supabase [7, 8]. Основу моделі складають ключові сутності, які забезпечують функціонування вебчату. Сутність "Користувачі" включає унікальні ідентифікатори, профільні дані, такі як email і налаштування, а також ролі, такі як користувач або адміністратор, що дозволяють персоналізувати доступ до функцій. Сутність "Чати" охоплює особисті та групові чати, містить метадані, включаючи назви груп і списки учасників. Сутність "Повідомлення" зберігає текст повідомлень, мітки часу, ідентифікатори відправника та чату, забезпечуючи зв’язок із іншими сутностями. Сутність "Статуси" відображає стан користувача, такий як онлайн

або офлайн, і синхронізується через WebSocket (WS) для оновлення в реальному часі [4].

Функціональна модель описує ключові процеси, які забезпечують роботу вебчату. Користувацький функціонал охоплює реєстрацію та авторизацію через email/пароль або OAuth 2.0, зокрема через Google, надсилання текстових повідомлень у реальному часі з підтримкою Markdown-форматування, створення групових чатів із можливістю додавання чи видалення учасників, відображення статусів користувачів, таких як онлайн чи офлайн, а також доступ до історії повідомлень для збереження контексту спілкування [4]. Адміністративний функціонал включає модерацію групових чатів, управління користувачами, наприклад блокування чи зміну ролей, і аналіз статистики активності, такої як кількість створених чатів чи надісланих повідомлень. Системні процеси забезпечують синхронізацію даних у реальному часі через WS, збереження повідомлень і метаданих у базі даних, оновлення статусів користувачів, а також налаштування інтерфейсу, включаючи вибір української мови чи темної або світлої теми [3].

Модель підтримує обробку даних у реальному часі з автоматичним оновленням чатів і статусів. Система гарантує конфіденційність через анонімізацію чутливих даних і контроль доступу на основі ролей. Архітектура дозволяє розширювати функціонал, наприклад, додаванням підтримки обміну файлами чи створення ботів, і ефективно обробляти велику кількість одночасних з’єднань, що відповідає потребам активних користувачів вебчату.

Модуль чатів є центральним елементом фронтенду, відповідальним за відображення індивідуальних і групових чатів, надсилання текстових повідомлень і збереження історії спілкування. Він реалізований за допомогою фреймворку Next.js, з використанням tRPC для типобезпечної взаємодії між клієнтом і сервером [6, 14]. Дані отримуються з бази PostgreSQL через Drizzle ORM, що забезпечує надійний доступ до інформації про чати, повідомлення та користувачів.

Для підвищення продуктивності і швидкості відгуку інтерфейсу, списки чатів і метадані повідомлень кешуються у Redis [15]. Це дозволяє зменшити

навантаження на базу даних і прискорити завантаження інтерфейсу. Реалтайм-доставка повідомлень та оновлення статусів користувачів (онлайн/офлайн) реалізовані через Ably Realtime. Завдяки механізму pub/sub, клієнти можуть підписуватись на відповідні канали, забезпечуючи доставку нових повідомлень і зміну статусів із затримкою до 100 мс. Інтеграція Ably SDK із Next.js дозволяє синхронізувати ці дані з фронтендом у режимі реального часу.

Модуль медіа забезпечує обробку файлів, таких як зображення та відео, які надсилаються в чатах. Він інтегрований із сервісом Cloudinary, що дозволяє здійснювати завантаження, стискання, зміну формату медіафайлів та генерацію URL для їхнього відображення у вебінтерфейсі [13]. Метадані про медіафайли зберігаються у PostgreSQL, тоді як самі файли зберігаються в хмарному сховищі, що зменшує серверне навантаження та прискорює доступ до контенту.

Кешуючий модуль на базі Redis також зберігає тимчасові дані, зокрема сесії користувачів, лічильники непрочитаних повідомлень і статуси онлайн, забезпечуючи наднизьку затримку (менше 1 мс) і мінімізацію запитів до реляційної бази даних [15]. Усі модулі взаємодіють між собою через внутрішнє API, реалізоване у Next.js, що відповідає за маршрутизацію запитів і обробку даних на серверній стороні.

Завдяки інтеграції зазначених технологій і принципів управління архітектурою, описаних у [18], система підтримує ефективну обробку великої кількості одночасних з’єднань, швидку передачу повідомлень, гнучку роботу з мультимедійним контентом і масштабованість, необхідну для сучасного вебчату.

2.3 Специфікація вимог до програмного забезпечення

1) **Призначення системи:**

1.1) вебзастосунок для обміну повідомленнями (вебчат) створений для забезпечення швидкої, безпечної та локально контрольованої комунікації між користувачами через вебінтерфейс. Система дозволяє користувачам надсилати текстові повідомлення в реальному часі, створювати групові чати, відстежувати статуси (онлайн/офлайн) і зберігати історію спілкування. Головна мета — надати

доступ до зручної та захищеної платформи, яка враховує потреби українців, зокрема високий рівень занепокоєння щодо конфіденційності (68%) і захисту даних (82%) [1];

1.2) погодження та стандарти: відповідність вимогам GDPR для захисту персональних даних, підтримка української мови інтерфейсу, використання сучасних вебтехнологій (HTML5, CSS3, WebSocket через Ably) для кросплатформної роботи., OAuth 2.0), сумісність із основними браузерами (Chrome, Firefox, Safari) [4, 16];

1.3) межі проекту: розробка вебверсії з адаптацією під мобільні пристрої через адаптивний дизайн, впровадження базового функціоналу: особисті та групові чати, автентифікація, статуси, історія повідомлень, відсутність функціоналу відеодзвінків, інтеграції з зовнішніми месенджерами, не включає розробку окремого мобільного застосунку;

2) **Сфера застосування:**

Система розрахована на користувачів віком від 16 років, які:

– потребують швидкої та безпечної комунікації для особистих чи професійних цілей;

– шукають локалізовану платформу з українським інтерфейсом;

– бажають зберігати конфіденційність даних і уникати іноземних серверів;

– активно використовують групові чати для соціальної чи робочої взаємодії.

3) **Архітектура системи** (рис. 2.10):

3.1) клієнтська частина: Реалізована на Next.js з використанням React-компонентів. Забезпечує адаптивний інтерфейс для відображення чатів, профілів, форм надсилання повідомлень і статусів онлайн. Використовує tRPC для запитів до сервера та Ably Realtime для реалтайм-оновлень [14, 16]. NextAuth обробляє автентифікацію на клієнтському рівні через cookies [17];

3.2) серверна частина: Побудована на Next.js із підтримкою API-роутів і tRPC для обробки запитів. Drizzle ORM забезпечує доступ до PostgreSQL,

а Redis використовується для кешування та управління сесіями. Ably Realtime обробляє реалтайм-події, а Cloudinary інтегрується для роботи з медіа. NextAuth управляє серверною логікою автентифікації;

3.3) база даних: PostgreSQL (через Supabase) зберігає структуровані дані, Cloudinary – медіафайли, Redis – кеш і тимчасові стани [7, 8, 13, 15];

3.4) додаткові сервіси: Ably Realtime для реального часу, NextAuth для безпечної автентифікації [16, 17];

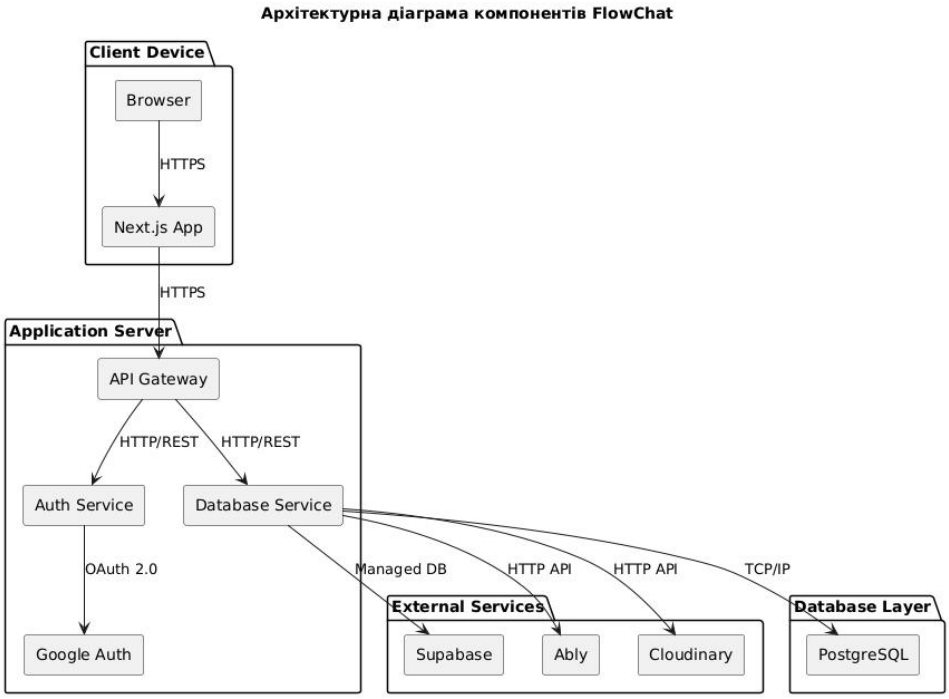

Рисунок 2.10 – Архітектура системи

4) **Характеристики користувачів:**

4.1) основні користувачі: дорослі та молодь (16+ років) із базовими навичками роботи з вебінтерфейсами, користувачі, які потребують безпечної комунікації, зокрема в умовах воєнних реалій, не вимагається спеціальних технічних чи професійних знань;

4.2) адміністратори/модератори: технічні спеціалісти з правами управління користувачами та контентом, мають доступ до аналітики (статистика чатів, активність користувачів) і модерації групових чатів.

5) **Загальні обмеження:**

5.1) технічні обмеження: потребує стабільного інтернет-з’єднання, підтримує лише сучасні браузери, обмеження на розмір медіафайлів (залежить від Cloudinary);

5.2) функціональні обмеження: не підтримує відеодзвінки чи голосові повідомлення, не замінює повноцінні месенджери з розширеним функціоналом (наприклад, Telegram), Обмежений набір функцій на стартовому етапі (без ботів чи автоматизації).

6) **Функціональні можливості:**

6.1) опис функції «Особистий кабінет та автентифікація»: реєстрація, авторизація та управління профілем користувача, авторизація та управління профілем користувача. Вхідна інформація: Email/пароль або дані Google (OAuth) [2]. Вихідна інформація: JWT-токен, персональна сторінка з історією чатів. Функціональні вимоги: підтримка OAuth 2.0, захист паролів через хешування, опція відновлення пароля;

6.2) опис функції «Обмін повідомленнями»: надсилання та отримання текстових повідомлень у реальному часі.. Вхідна інформація: текст повідомлення, ідентифікатор чату. Вихідна інформація: відображення повідомлення, статус "доставлено". Функціональні вимоги: WS для миттєвої передачі, підтримка Markdown-форматування [4];

6.3) опис функції «Групові чати»: Створення та управління груповими чатами. Вхідна інформація. Назва чату, список учасників. Вихідна інформація: Груповий чат із списком учасників, історія повідомлень. Функціональні вимоги: Можливість додавання/видалення учасників, модерація адміністратором;

6.4) опис функції «Статуси користувачів»: відображення статусу онлайн/офлайн. Вхідна інформація: стан з’єднання користувача. Вихідна інформація: оновлення статусу в інтерфейсі. Функціональні вимоги: синхронізація між клієнтами;

6.5) опис функції «Історія повідомлень»: Зберігання та відображення історії чатів. Вхідна інформація: Ідентифікатор чату, період запиту. Вихідна

інформація: Список повідомлень із мітками часу. Функціональні вимоги: Зберігання в PostgreSQL, завантаження при вході [7];

6.6) опис функції «Системні налаштування»: Керування інтерфейсом і параметрами користувача. Вхідна інформація: вибір мови, теми (світла/темна). Вихідна інформація: оновлений інтерфейс. Функціональні вимоги: Локалізація українською, адаптивний дизайн.

7) **Технічні вимоги:**

7.1) Використання HTTPS-з’єдання;

7.2) клієнтська частина: підтримка останніх версій Chrome, Firefox, Safari;

8) **Вимоги до безпеки:**

Використання HTTPS-з’єдання та шифрування повідомлень у БД, Захист від XSS і SQL-ін’єкцій через валідацію даних.

9) **Продуктивність та надійність:**

Час відгуку системи: не більше 300 мс, Обробка повідомлення: до 100 мс через Ably Realtime, Підтримка до 1000 одночасних з’єднань

10) **Перспективи розвитку:**

Розширення функціоналу: підтримка файлів, стікери, боти; Інтеграція з мобільним застосунком у майбутньому; Впровадження аналітики використання чатів

11) **Додаткові вимоги:**

Можливість адаптації до зростаючого навантаження.

12) **Властивості програмного забезпечення:**

12.1) **гнучкість.** Налаштування рівнів доступу (користувач, адміністратор), кастомізація інтерфейсу (теми, локалізація), можливість додавання нових функцій, таких як обмін файлами;

12.2) **сумісність**. Підтримка вебстандартів, інтеграція з Cloudinary і Ably;

12.3) **масштабованість.** Модульна архітектура Next.js, гнучкі механізми обробки з’єднань, підтримка розширення функціоналу.;

12.4) **продуктивність.** Середній час відгуку API 200–300 мс, обробка повідомлення до 100 мс, кешування через Next.js SSG, підтримка до 1000 користувачів;

12.5) **безпека.** хешування паролів, підтримка HTTPS;

12.6) **надійність.** Автоматичне резервне копіювання, стійкість до розривів WebSocket-з’єднань;

12.7) **доступність.** Адаптивний інтерфейс для всіх пристроїв, українська локалізація;

Висновки до розділу 2

У розділі 2 проведено комплексний аналіз моделювання вебзастосунку FlowChat, призначеного для швидкої, безпечної та зручної реалтайм-комунікації, з урахуванням потреб української аудиторії. Інформаційна модель визначила ключові сутності системи — користувачі, чати, повідомлення, статуси онлайн/офлайн і медіафайли, — які формують реляційну структуру бази даних на PostgreSQL через Supabase [7, 8]. Використання Drizzle ORM забезпечує типобезпечне управління схемами, автоматизацію міграцій і ефективну обробку складних запитів, що підтримує надійність і гнучкість при роботі з великими обсягами даних [12]. Функціональна модель детально описала основні процеси: автентифікацію через NextAuth, реалтайм-надсилання повідомлень за допомогою Ably Realtime, створення й модерацію групових чатів, обробку медіафайлів через Cloudinary і відображення статусів користувачів, що разом гарантують швидкість, зручність і безпеку комунікації [4, 13, 16, 17].

Технологічний стек ретельно підібрано для забезпечення продуктивності, масштабованості та безпеки. Next.js, як основний фреймворк, підтримує серверний рендеринг (SSR), статичну генерацію (SSG) і API-роути, створюючи адаптивний інтерфейс із локалізацією українською мовою для цільової аудиторії [1, 6]. tRPC забезпечує типобезпечну взаємодію між фронтендом і бекендом, мінімізуючи помилки та спрощуючи розробку API [14]. Ably Realtime, використовуючи модель pub/sub, доставляє повідомлення та статуси з затримкою

до 100 мс, замінюючи традиційні WebSocket-з’єднання і забезпечуючи стабільність при високих навантаженнях [16]. Redis підвищує продуктивність шляхом кешування сесій, статусів і метаданих, таких як лічильники непрочитаних повідомлень, знижуючи навантаження на PostgreSQL [15]. Cloudinary оптимізує медіафайли, зберігаючи їх у хмарі та надаючи URL для швидкого відображення [13]. NextAuth реалізує безпечну автентифікацію з підтримкою OAuth 2.0 (Google) і Credentials, відповідаючи стандартам GDPR і нормам українського законодавства щодо захисту даних [3, 17].

Специфікація вимог враховує сучасні стандарти безпеки, локалізацію українською мовою, адаптивний дизайн і кросбраузерність, забезпечуючи доступність для користувачів віком від 16 років, які потребують безпечної комунікації [1, 3]. Трирівнева архітектура (клієнт, сервер, база даних) гарантує модульність, масштабованість і здатність обробляти до 1000 одночасних з’єднань із середнім часом відгуку API 200–300 мс, уникаючи ерозії архітектури завдяки чіткому визначенню вимог [18]. Обмеження включають залежність від хмарних сервісів (Supabase, Cloudinary, Ably), що може впливати на операційні витрати, і потребу в стабільному інтернет-з’єднанні для реалтайм-функцій. Перспективи розвитку охоплюють впровадження підтримки файлів, стікерів, ботів, аналітики чатів і можливої інтеграції з мобільними застосунками, що посилить конкурентоспроможність FlowChat як сучасного, локалізованого месенджера з потенціалом для розширення функціоналу [4].

3 ПРОЄКТУВАННЯ ВЕБЧАТУ

3.1 Архітектура ПЗ

Вебзастосунок FlowChat, призначений для швидкої та зручної реалтайм-комунікації, потребує архітектури, яка забезпечує ефективну роботу користувацького інтерфейсу та його інтеграцію з серверними сервісами, зберігаючи простоту й продуктивність. Для реалізації обрано компонентно-орієнтовану архітектуру, що дозволяє ізолювати функціональні модулі фронтенду та їхню взаємодію з бекендом через чітко визначені інтерфейси. Архітектура FlowChat складається з модулів, відповідальних за автентифікацію, відображення чатів, обробку медіафайлів, реалтайм-комунікацію та кешування даних, які інтегруються за допомогою tRPC та Ably Realtime [14, 16]. Такий підхід підкреслює фронтенд-орієнтовану розробку, забезпечуючи адаптивний і локалізований інтерфейс для користувачів.

Основна увага в архітектурі FlowChat приділяється фронтенду, реалізованому на Next.js, який забезпечує адаптивний і локалізований українською мовою інтерфейс для комфортної взаємодії користувачів із чатами. Фронтенд складається з React-компонентів, що відповідають за автентифікацію, відображення списків чатів, надсилання повідомлень і роботу з медіа, стилізованих для зручного користувацького досвіду. Модуль автентифікації, інтегрований із NextAuth, дозволяє користувачам входити в систему через Google або email/пароль, отримуючи JWT-токени для доступу до функцій. Дані автентифікації передаються через tRPC, сесії зберігаються в Redis для швидкої перевірки, а профілі користувачів записуються в PostgreSQL за допомогою Drizzle ORM [12, 14, 15, 16]. Цей модуль забезпечує безпечний доступ до чатів і персоналізованих функцій, таких як редагування імені чи аватарки.

Модуль чатів є центральним для фронтенду, відповідаючи за відображення індивідуальних і групових чатів, надсилання текстових повідомлень і збереження історії спілкування. Реалізований на Next.js, він використовує tRPC для запитів до сервера, отримуючи дані з PostgreSQL через Drizzle ORM [7, 14]. Для оптимізації

списки чатів і метадані повідомлень кешуються в Redis, що прискорює завантаження інтерфейсу [15]. Реалтайм-доставка повідомлень забезпечується через Ably Realtime, що дозволяє користувачам миттєво бачити нові повідомлення та статуси онлайн/офлайн. Модуль медіа обробляє файли, такі як зображення чи відео, що надсилаються в чатах. Інтегрований із Cloudinary, він забезпечує завантаження, оптимізацію файлів (стиснення, зміна формату) і генерацію URL для їх відображення. Метадані медіа зберігаються в PostgreSQL, тоді як файли розміщуються в хмарі, що зменшує навантаження на сервер. Фронтенд отримує доступ до медіа через tRPC, швидко відображаючи контент у чатах.

Модуль реалтайм-комунікації, побудований на Ably Realtime, відповідає за миттєву доставку повідомлень і оновлення статусів онлайн/офлайн. Використовуючи pub/sub-механіку, Ably дозволяє клієнтам підписуватися на канали чатів або статусів, забезпечуючи затримку до 100 мс [16]. Інтеграція з Next.js через Ably SDK синхронізує реалтайм-дані з фронтенд-компонентами чатів. Модуль кешування, реалізований на Redis, зберігає тимчасові дані, такі як сесії користувачів, лічильники непрочитаних повідомлень і статуси онлайн, що дозволяє досягти наднизької затримки (менше 1 мс) і зменшує кількість запитів до PostgreSQL. Усі компоненти взаємодіють через API, реалізоване в Next.js, яке обробляє запити до відповідних модулів.

Процес надсилання повідомлення ілюструється діаграмою послідовності: користувач через фронтенд надсилає запит за допомогою tRPC до модуля чатів, який перевіряє автентифікацію через NextAuth, отримуючи сесію з Redis. Повідомлення записується в PostgreSQL через Drizzle ORM, а Ably Realtime доставляє його до отримувача в реальному часі [7, 12, 16]. Якщо повідомлення містить медіа, Cloudinary обробляє файл і повертає URL, який зберігається в базі. Для фонових задач, таких як оновлення лічильників повідомлень, використовується Redis як внутрішня черга, що дозволяє обробляти їх без затримок для користувачів. Дані про активність чатів зберігаються в PostgreSQL для можливості перегляду історії [7].

База даних PostgreSQL, розміщена через Supabase, гарантує надійне зберігання даних із підтримкою транзакцій. Drizzle ORM спрощує роботу з базою, забезпечуючи типобезпеку для запитів і схем. Cloudinary оптимізує медіафайли, а Ably і Redis підтримують реалтайм-функції та кешування. Компонентно-орієнтована архітектура FlowChat забезпечує модульність, що спрощує розробку фронтенду, швидке завантаження інтерфейсу завдяки Redis і Ably, а також безпеку через NextAuth і GDPR-сумісне зберігання даних [3, 15, 16, 17]. Локалізація українською мовою в Next.js відповідає потребам аудиторії [19]. Обмеження включають залежність від хмарних сервісів (Supabase, Cloudinary, Ably), що може впливати на витрати, і потребу в стабільному інтернет-з’єднанні для реалтайм-функцій. У майбутньому планується додавання функцій, таких як підтримка ботів або аналіз повідомлень за допомогою AI.

Користувач взаємодіє з FlowChat через Next.js-фронтенд із локалізованим інтерфейсом. Уніфікований сценарій (рис. 3.1) використання ілюструється діаграмою варіантів використання, що включає авторизацію через Google або email/пароль, створення індивідуальних і групових чатів, надсилання текстових повідомлень і медіа, перегляд статусів онлайн/офлайн, редагування профілю (ім’я, аватарка, налаштування), пошук користувачів за іменем для початку чату, а також модерацію для користувачів із роллю модератора, зокрема видалення повідомлень чи блокування учасників. Користувач автентифікується через фронтенд, який надсилає запит до API. Модуль автентифікації перевіряє сесію в Redis, після чого користувач може створювати чати чи надсилати повідомлення. Дані записуються в PostgreSQL, медіа обробляються Cloudinary, а повідомлення доставляються через Ably Realtime. Результати повертаються через tRPC, забезпечуючи швидку та безпечну взаємодію.

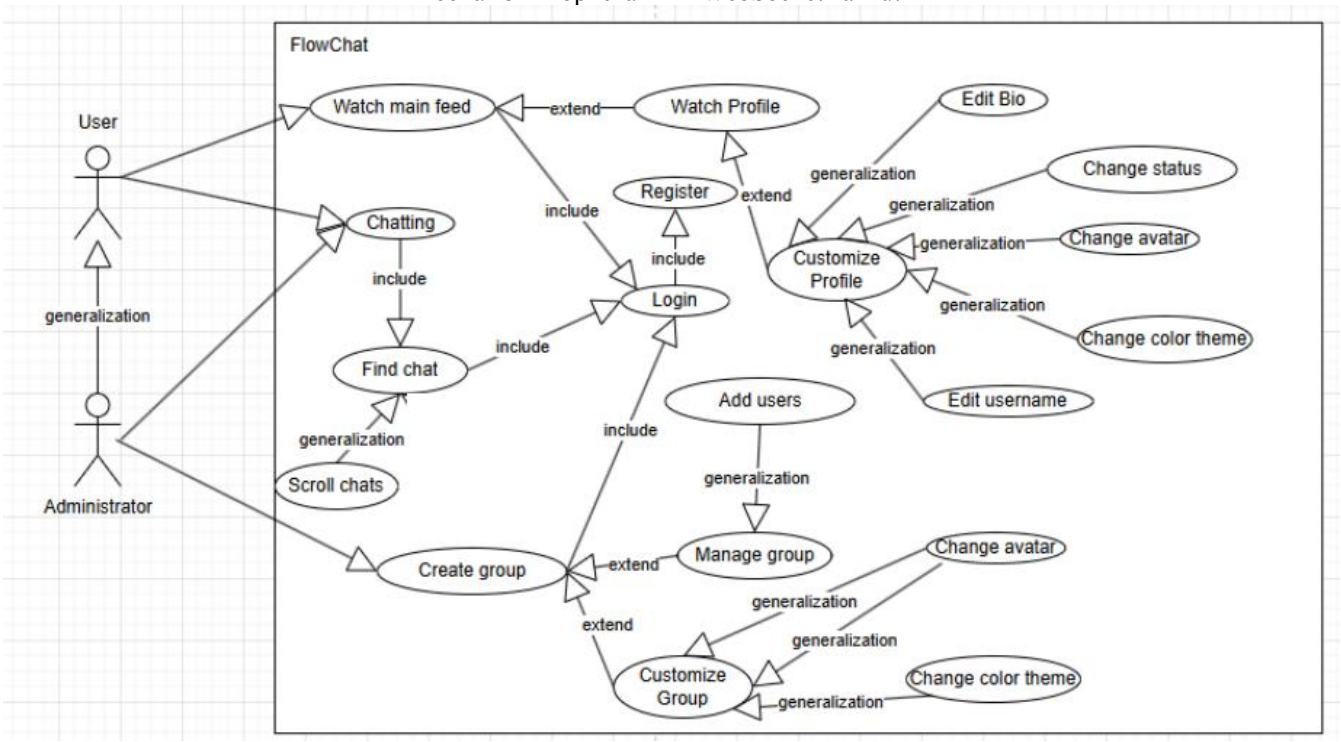

Рисунок 3.1 – Уніфікований сценарій використання

Ця архітектура забезпечує створення зручного, швидкого та безпечного месенджера, орієнтованого на фронтенд, із підтримкою реалтайм-комунікації та потенціалом для подальшого розвитку.

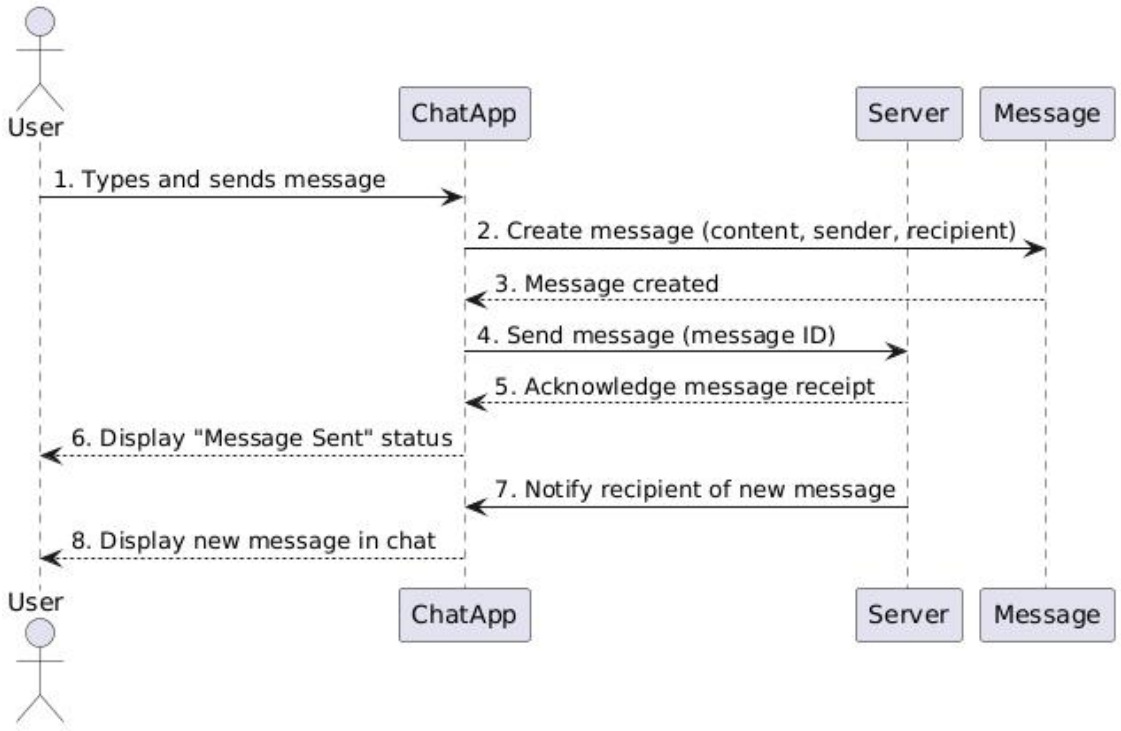

Рисунок 3.2 – Діаграма послідовності надсилання повідомлення

Ця діаграма (рис. 3.2) демонструє етапи відправлення повідомлення від користувача через додаток FlowChat до сервера, включаючи створення та передачу.

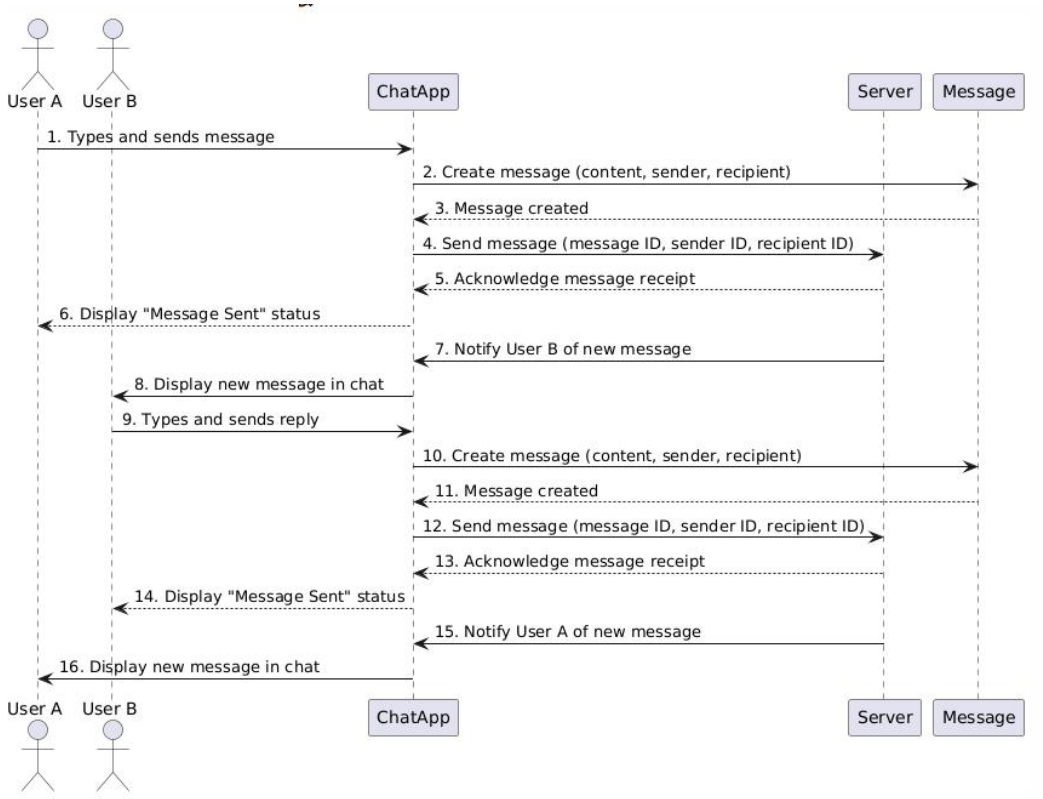

Рисунок 3.3 – Діаграма послідовності спілкування

Діаграма (рис. 3.3) відображає цикл взаємодії між користувачем, додатком FlowChat і сервером, акцентуючи на сповіщенні отримувача.

3.2 Конструювання ПЗ

Конструювання програмного забезпечення вебзастосунку FlowChat передбачає детальну розробку логіки реалізації фронтенд-компонентів системи, їхньої взаємодії, структури та поведінки, з акцентом на користувацький інтерфейс і реалтайм-комунікацію. Для ілюстрації структури й динаміки застосунку створено набір UML-діаграм, що відображають ключові сутності, стани, розгортання та організацію коду, забезпечуючи чітке розуміння функціонування чат-системи.

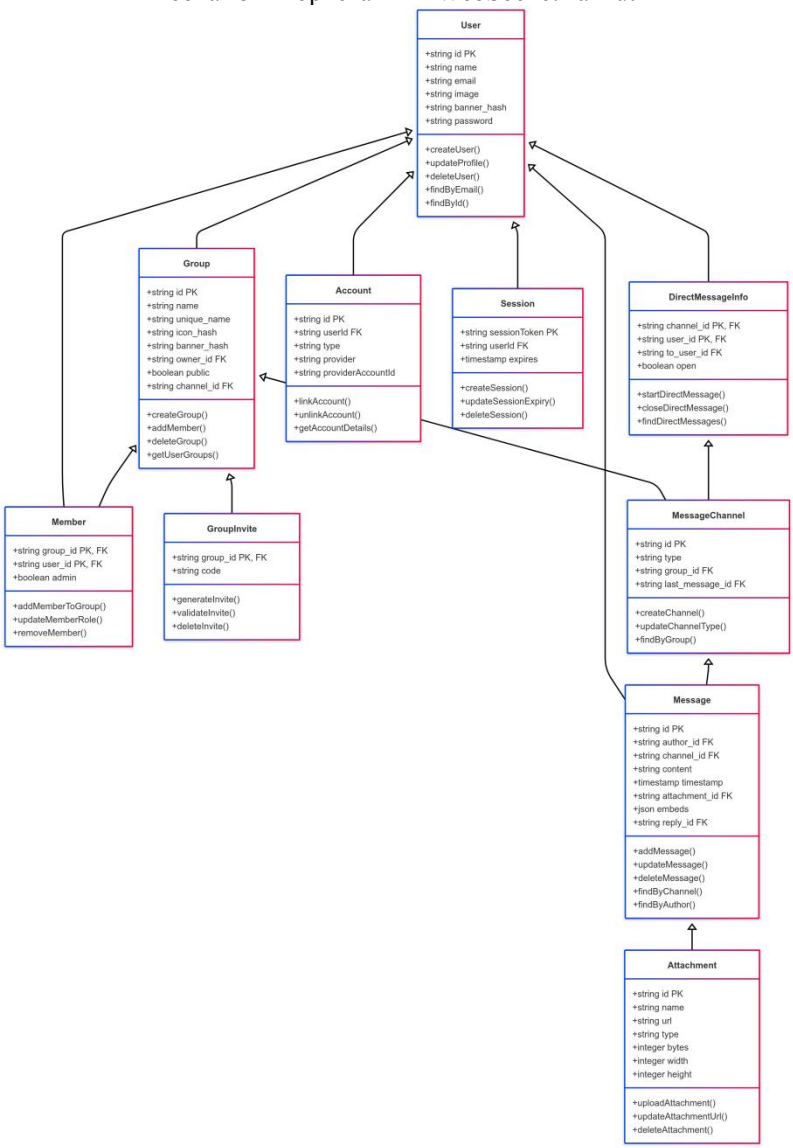

Рисунок 3.4 – Діаграма класів

На рисунку 3.4 представлена діаграма класів, що описує основні сутності системи, які реалізують функціонал створення чатів, надсилання повідомлень, обробки медіафайлів і управління профілями користувачів. Діаграма демонструє класи, їхні атрибути, методи та зв’язки між об’єктами, відповідні структурі бази даних. Основні класи включають:

– User – клас, що моделює користувача системи. Містить атрибути, такі як id, name, email, image, banner_hash, і методи для автентифікації та оновлення профілю. Пов’язаний із чатами через клас Member і повідомленнями через Message;

– Group – клас, що представляє груповий чат. Включає id, name, unique_name, icon_hash, banner_hash, owner_id, public. Має зв’язок із MessageChannel через channel_id і з учасниками через Member;

– Message – клас для повідомлень у чаті. Містить id, author_id, channel_id, content, timestamp, attachment_id, embeds, reply_id, а також методи для створення та видалення. Пов’язаний із User і MessageChannel;

– MessageChannel – клас, що моделює канал комунікації (DM або груповий). Включає id, type, group_id, last_message_id. Забезпечує зв’язок між чатами та повідомленнями;

– Attachment – клас для медіафайлів, що надсилаються в повідомленнях. Містить id, name, url, type, bytes, width, height. Інтегрується з Cloudinary для зберігання файлів;

– DirectMessageInfo – клас для прямих повідомлень між двома користувачами. Включає channel_id, user_id, to_user_id, open;

– GroupInvite – клас для запрошень до груп. Містить group_id, code;

– Account – клас для управління автентифікацією та сесіями. Містить id, userId, type, provider, providerAccountId, refresh_token, access_token, expires_at, token_type, scope, id_token, session_state. Використовується для OAuth-автентифікації та зберігання даних сесій, кешованих у Redis;

Ці класи відображають структуру таблиць бази даних, де логіка сесій реалізована через Account, і забезпечують логіку роботи фронтенду для відображення чатів, повідомлень і профілів.

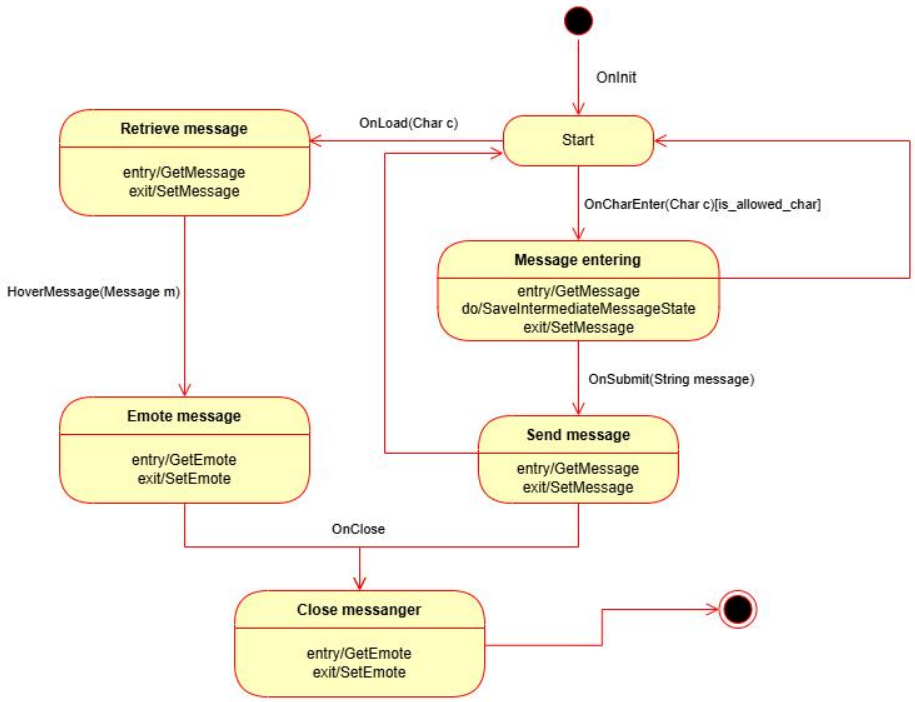

Рисунок 3.5 – Діаграма станів месенджера

На рисунку 3.5 представлена діаграма станів, що моделює поведінку користувача під час взаємодії з чатом у FlowChat. Діаграма відображає стани фронтенд-компонентів і переходи між ними залежно від дій користувача. Користувач може перебувати в стані перегляду списку чатів, вибору чату, написання повідомлення, очікування доставки або перегляду медіа. Наприклад, при виборі чату система переходить у стан відображення історії повідомлень, отримуючи дані через tRPC із таблиці messages. Під час написання повідомлення активується стан введення тексту, а після натискання «Надіслати» система переходить у стан відправки, викликаючи Ably Realtime для доставки до MessageChannel. Якщо додається медіа, система переходить у стан завантаження через Cloudinary, зберігаючи метадані в attachments. Користувач може повернутися до списку чатів або вийти з чату, завершуючи взаємодію. Діаграма ілюструє реакцію фронтенду на дії, забезпечуючи плавний користувацький досвід.

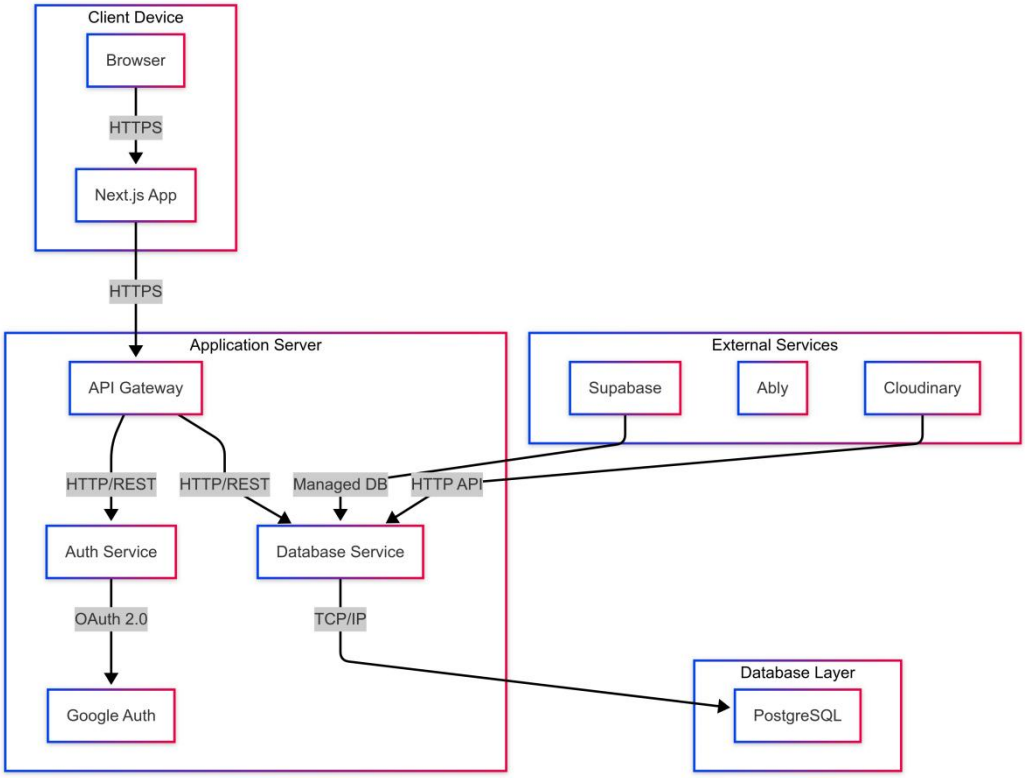

Рисунок 3.6 – Діаграма розгортання

На рисунку 3.6 представлена діаграма розгортання, що ілюструє організацію компонентів FlowChat у середовищі виконання. Фронтенд, побудований на Next.js, розгортається як клієнтський застосунок у браузері та взаємодіє з серверною частиною через tRPC. Серверна частина, також на Next.js, обробляє запити, використовуючи Drizzle ORM для доступу до PostgreSQL (розміщеного через Supabase), де зберігаються дані таблиць users, groups, messages тощо. Redis кешує дані автентифікації з accounts і статуси онлайн, а Ably Realtime забезпечує доставку повідомлень. Cloudinary обробляє медіафайли для таблиці attachments, надаючи URL для відображення на фронтенді. Діаграма показує інтеграцію фронтенду з хмарними сервісами для реалтайм-функцій.

Рисунок 3.7 – Діаграма пакетів

На рисунку 3.7 представлена діаграма пакетів, що відображає організацію коду FlowChat. Пакет frontend містить React-компоненти для автентифікації (з NextAuth), чатів, профілів і медіа, а також хуки для роботи з tRPC і Ably Realtime. Пакет api включає tRPC-ендпоінти для обробки запитів, таких як створення чатів із groups чи надсилання повідомлень у messages. Пакет database містить схеми Drizzle ORM, що відповідають таблицям users, groups, messages, accounts, attachments тощо. Пакет services інтегрує зовнішні сервіси, такі як Cloudinary для медіа та Redis для кешування даних accounts. Ця структура забезпечує модульність, спрощуючи розробку фронтенду.

Фронтенд-логіка реалізована через React-компоненти, що взаємодіють із сервером через tRPC. Компонент чату отримує повідомлення з messages і messageChannels, відображаючи їх у реальному часі завдяки Ably Realtime. Компонент профілю дозволяє редагувати дані з users, викликаючи tRPC-запити. Медіа з attachments відображаються через URL від Cloudinary. NextAuth управляє автентифікацією, використовуючи accounts для OAuth і кешування сесій у Redis. PostgreSQL зберігає дані, а Drizzle ORM забезпечує типобезпечні запити. Ця організація створює швидкий, локалізований українською мовою інтерфейс із реалтайм-комунікацією. Обмеження включають залежність від хмарних сервісів (Supabase, Cloudinary, Ably), що може впливати на витрати, і потребу в стабільному інтернет-з’єднанні. У майбутньому планується додавання функцій, таких як підтримка ботів [2, 4, 8].

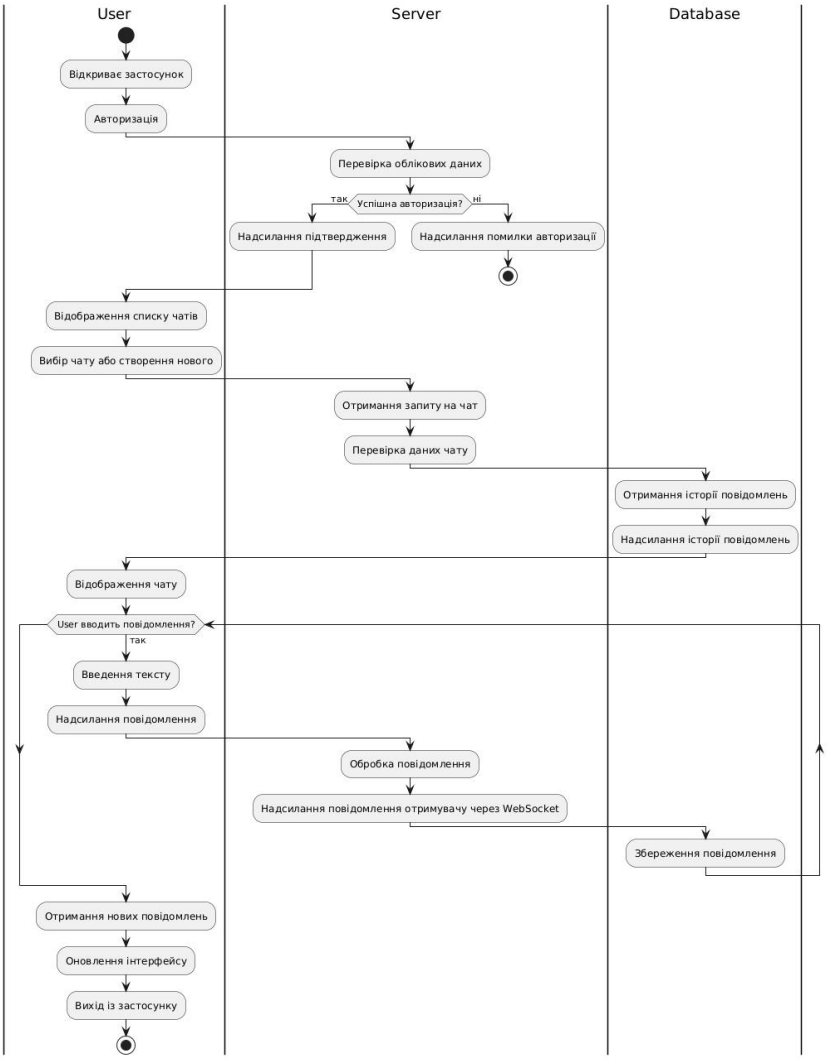

Рисунок 3.8 – Swimlane діаграма діяльності застосунку

На рисунку 3.8 показано swimlane-діаграму діяльності вебзастосунку FlowChat, що ілюструє взаємодію між користувачем, сервером і базою даних під час використання чату. Діаграма починається з авторизації користувача, після чого сервер перевіряє облікові дані. У разі успіху користувач отримує доступ до списку чатів.

Після вибору або створення чату надсилається запит на сервер, який звертається до бази даних за історією повідомлень. Далі користувач може ввести повідомлення, яке передається на сервер, зберігається у базі й надсилається іншим учасникам через WebSocket. Нові повідомлення відображаються в інтерфейсі, і сесія завершується виходом із застосунку.

Діаграма відображає основні етапи роботи чату та розподіл відповідальності між компонентами системи.

3.3 Зберігання даних

Підрозділ описує побудову та використання моделі бази даних вебзастосунку FlowChat, призначеного для реалтайм-комунікації, а також використання міграцій, формування ER-діаграми («сутність-зв’язок»). Додатково представлено опис технології повнотекстового пошуку та індексації для оптимізації запитів.

3.3.1 ER-діаграма даних

ER-діаграма (рис. 3.9) описує структуру реляційної бази даних FlowChat, яка забезпечує функціонування реалтайм-чату. База даних, побудована на PostgreSQL через сервіс Supabase, використовує Drizzle ORM для типобезпечного управління схемами. Вона охоплює кілька функціональних зон: автентифікацію, організацію чатів, обробку повідомлень, управління групами та зберігання медіафайлів.

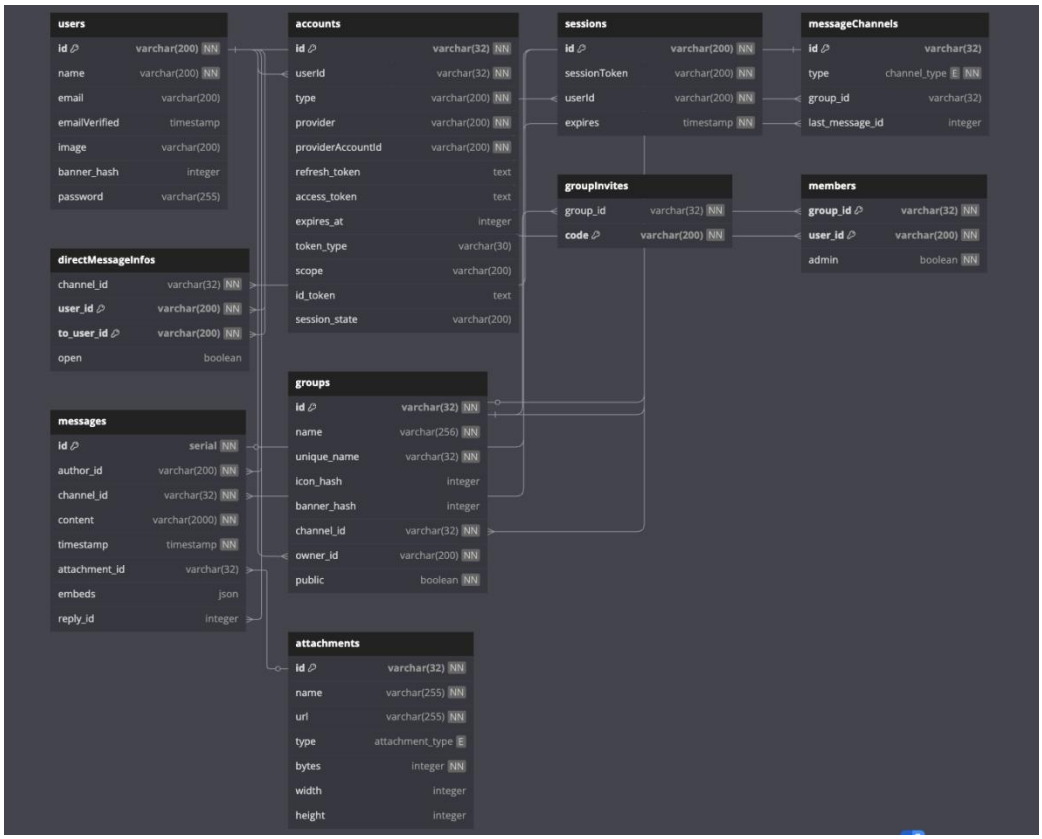

Рисунок 3.9 – ER-діаграма

У центрі логіки бази даних розташована таблиця users, яка зберігає облікову інформацію про користувачів: унікальний ідентифікатор (id), ім’я (name), електронну пошту (email), аватарку (image), хеш банера (banner_hash) і пароль (password) для автентифікації через email/пароль. З цією таблицею пов’язані інші сутності: автентифікаційні дані через accounts, сесії через sessions, повідомлення через messages, учасники груп через members і прямі повідомлення через directMessageInfos. Таблиця accounts містить дані для OAuth-автентифікації, такі як provider, providerAccountId, access_token, а sessions управляє сесіями з полями sessionToken і expires, кешованими в Redis для швидкого доступу.

Для організації чатів застосовуються таблиці messageChannels, directMessageInfos і groups. Таблиця messageChannels визначає канали комунікації з полями id, type (DM або GROUP), group_id і last_message_id, що посилається на останнє повідомлення з messages. Таблиця directMessageInfos моделює прямі повідомлення між двома користувачами через user_id, to_user_id і channel_id з полем open для позначення активності. Таблиця groups зберігає дані групових чатів: id, name, unique_name, icon_hash, banner_hash, owner_id і public, а також посилається на канал через channel_id.

Повідомлення зберігаються в таблиці messages з полями id, author_id, channel_id, content, timestamp, attachment_id, embeds (у форматі JSON) і reply_id для відповідей на інші повідомлення. Таблиця attachments містить метадані медіафайлів: id, name, url, type (image, video, raw), bytes, width, height, інтегруючись із Cloudinary для зберігання файлів. Зв’язок між messages і attachments дозволяє швидко отримувати медіа для відображення.

Для управління групами створені таблиці members і groupInvites. Таблиця members визначає учасників груп через group_id, user_id і admin (роль адміністратора), а groupInvites містить коди запрошень (code) для приєднання до груп через group_id. Усі таблиці пов’язані через зовнішні ключі, наприклад, messages.author_id посилається на users.id, а groups.owner_id – також на users.id, що забезпечує цілісність даних.

3.3.2 Застосування міграцій

Для налаштування та автоматичного розгортання структури бази даних FlowChat використано механізм міграцій Drizzle ORM. Цей підхід дозволяє централізовано створювати, оновлювати та відкочувати зміни до схеми бази даних без ручного втручання, забезпечуючи узгодженість між кодом і схемою PostgreSQL (через Supabase).

У файлі схеми описуються сутності, які відображають структуру таблиць бази даних. Кожна таблиця визначається через функцію pgTable, а поля конфігуруються з використанням типів, таких як varchar, integer, boolean, із зазначенням первинних ключів (primaryKey), унікальних індексів (uniqueIndex) та індексів (index). Генерація міграцій виконується за допомогою інструменту Drizzle Kit, який аналізує зміни в схемах і автоматично створює SQL-скрипти для оновлення бази даних. Команда drizzle-kit generate генерує міграційні файли, а drizzle-kit push застосовує їх до бази. Лістинг коду User Entity:

```
export const users = pgTable(
  "User",
  {
    id: varchar200("id").notNull().primaryKey(),
    name: varchar200("name").notNull().default("User"),
    banner_hash: integer("banner_hash"),
    emailVerified: timestamp("emailVerified"),
    image: varchar200("image"),
    email: varchar200("email"),
    password: varchar255("password"),
  },
  (table) => ({
    User_email_key: uniqueIndex("User_email_key").on(table.email),
  }),
);
```

Цей код описує таблицю users із полями id, name, email, emailVerified, image, banner_hash, password. Після оновлення схеми команда drizzle-kit generate створює міграцію, наприклад, 20250527T234500_add_password_to_users.sql, яка додає колонку password до таблиці. Виконання міграції через Supabase CLI або прямий доступ до бази гарантує синхронізацію структури з кодом, зберігаючи дані.

Міграції також підтримують індексацію, наприклад, унікальний індекс на email для users чи індекс на channel_id для messages, що оптимізує запити. Цей

підхід спрощує розробку та супровід бази даних, адаптовуючи її до реалтайм-функцій FlowChat.

3.4 Реалізація інтерактивних функцій реалтайм-чату

Вебзастосунок FlowChat, розроблений для забезпечення швидкої та зручної реалтайм-комунікації, передбачає реалізацію інтерактивних функцій, які підвищують залученість користувачів і забезпечують інтуїтивну взаємодію з чатами. Ці функції, реалізовані у фронтенд-компонентах на базі Next.js, включають відповіді на повідомлення, відображення статусу набору тексту, модерацію групових чатів і пошук користувачів, що підтримуються через інтеграцію з tRPC і Ably Realtime. Використання даних із таблиць бази даних, таких як users, groups, messages, messageChannels, directMessageInfos, groupInvites, members, accounts і attachments, дозволяє створювати динамічний і персоналізований інтерфейс, адаптований для української аудиторії.

Інтерактивні функції FlowChat зосереджені на React-компонентах, які відповідають за ключові аспекти комунікації. Однією з основних функцій є можливість відповідати на конкретні повідомлення, реалізована через поле reply_id у таблиці messages. У фронтенді користувач може вибрати повідомлення через контекстне меню, після чого компонент чату відображає цитату оригінального повідомлення разом із полем для введення відповіді. При відправленні tRPC-запит записує нове повідомлення в messages, пов’язуючи його з оригінальним через reply_id, а Ably Realtime доставляє його до учасників чату в реальному часі. Ця функція забезпечує структуровану комунікацію, особливо в групових чатах, де зберігається контекст розмови.

Статус набору тексту є ще однією інтерактивною функцією, яка підвищує динамічність спілкування. Коли користувач починає вводити повідомлення, фронтенд-компонент надсилає подію через Ably Realtime до відповідного каналу (messageChannels), що дозволяє відобразити індикатор «користувач пише» для інших учасників. Ця подія обробляється з затримкою до 100 мс, забезпечуючи миттєве оновлення інтерфейсу без додаткових запитів до PostgreSQL. Дані про

користувача, такі як name із таблиці users, відображаються поруч із індикатором, що додає персоналізації. Функція активується лише для активних чатів, визначених полем open у directMessageInfos для прямих повідомлень або статусом учасників у members для груп.

Модерація групових чатів, реалізована для користувачів із роллю адміністратора (admin у members), дозволяє видаляти повідомлення або блокувати учасників. У фронтенді адміністратор через контекстне меню може вибрати повідомлення з messages для видалення, викликаючи tRPC-запит, який видаляє запис із бази даних. Для блокування учасника компонент надсилає запит на видалення запису з members, після чого Ably Realtime оновлює список учасників у групі (groups). Ці дії супроводжуються сповіщеннями в чаті, що відображаються як системні повідомлення, створені через messages із спеціальним форматом content. Такий підхід забезпечує прозорість модерації та зручність управління групами.

Функція пошуку користувачів дозволяє знаходити співрозмовників для створення прямих чатів. Компонент пошуку викликає tRPC-запит до таблиці users, фільтруючи записи за полем name. Результати відображаються в реальному часі у вигляді списку з аватарками (image) і статусами онлайн, кешованими в Redis. При виборі користувача створюється новий канал у messageChannels і запис у directMessageInfos, після чого відкривається чат. Ця функція оптимізує ініціацію спілкування, особливо для нових користувачів.

Інтерфейс підтримує додаткові інтерактивні елементи, такі як попередній перегляд медіафайлів із attachments, реалізований через Cloudinary. Користувач може натиснути на зображення чи відео (url, type), щоб відкрити його в модальному вікні, що покращує взаємодію з контентом. Локалізація українською мовою, реалізована через Next.js, охоплює всі інтерактивні елементи, включаючи підказки, кнопки та сповіщення. Адаптивний дизайн, побудований із CSS-модулями, забезпечує коректне відображення функцій на різних пристроях. Анімації, такі як плавне з’явлення повідомлень чи індикатора набору тексту, додають зручності.

Реалізація інтерактивних функцій інтегрована з серверною частиною через tRPC, що забезпечує типобезпечність, і Ably Realtime для реалтайм-оновлень. Drizzle ORM спрощує роботу з PostgreSQL (Supabase), дозволяючи ефективно обробляти запити до messages, users, groups. Redis кешує тимчасові дані, такі як статуси набору тексту чи автентифікаційні токени з accounts. Обмеженнями є залежність від хмарних сервісів (Supabase, Cloudinary, Ably), що може впливати на витрати, і потреба в стабільному інтернет-з'єднанні для реалтайм-функцій. У майбутньому планується розширення інтерактивності, наприклад додавання реакцій на повідомлення чи інтеграція AI для автодоповнення тексту [2, 4, 8].

3.5 Локалізація та мультимовність

Підтримка кількох мов у FlowChat реалізована для розширення аудиторії та покращення доступності, що дозволяє застосунку адаптуватися до потреб користувачів із різних культурних і лінгвістичних контекстів. Локалізація базується на бібліотеці next-i18next, яка інтегрується з Next.js для ефективного управління перекладами, забезпечуючи гнучкість і масштабованість. Файли перекладів зберігаються в директорії locales, наприклад, locales/en.json для англійської та locales/uk.json для української, що дозволяє легко додавати нові мови в міру розширення проєкту. Користувач може змінити мову через компонент ThemeSwitcher, який розширено для перемикання мов, як показано в прикладі з (chat)/(app)/settings/page.tsx, де додано селектор мови з опціями English і Українська, що інтегрується з логікою перемикання через setLocale.

Переклади динамічно завантажуються на основі вибраної мови, що забезпечує миттєву зміну інтерфейсу без необхідності перезавантаження сторінки, наприклад, текст "No groups yet" на сторінці / змінюється на "Немає груп" для української локалізації, що реалізовано через useTranslations із next-intl, як видно з прикладу в (chat)/(app)/page.tsx, де текст noGroups береться з відповідного файлу перекладів. Файли перекладів структуровані для легкої підтримки, наприклад, locales/en.json містить ключі для сторінки Home з текстами "No groups yet" і описом, що полегшує оновлення вмісту без порушення цілісності коду. Ця

система також дозволяє адаптувати не лише текст, але й форматування дат, чисел і валют, що є важливим для міжнародних користувачів, наприклад, формат часу в чатах може змінюватися залежно від локалі.

Для подальшого вдосконалення планується додати підтримку RTL (right-to-left) мов, таких як арабська, що вимагатиме додаткових налаштувань у Tailwind CSS і логіці компонентів, щоб забезпечити правильне відображення тексту та елементів інтерфейсу. Це розширить аудиторію до регіонів із високим попитом на подібні рішення, зберігаючи при цьому інтуїтивність і зручність. Такий підхід до локалізації не лише підвищує доступність, але й сприяє залученню нових користувачів, роблячи FlowChat більш конкурентоспроможним на глобальному ринку.

Висновки до розділу 3

Розробка вебзастосунку FlowChat, призначеного для швидкої, безпечної та зручної реалтайм-комунікації, ґрунтується на компонентно-орієнтованій архітектурі, яка забезпечує модульність, високу продуктивність і гнучкість для подальшого масштабування. Фронтенд, побудований на Next.js, пропонує адаптивний і локалізований українською мовою інтерфейс, що інтегрується з серверною частиною через tRPC та Ably Realtime. Ця інтеграція підтримує ключові модулі: автентифікацію (на базі NextAuth із кешуванням сесій у Redis), управління чатами (з підтримкою індивідуальних і групових каналів), обробку медіафайлів (через Cloudinary з оптимізацією та зберіганням метаданих у PostgreSQL) і реалтайм-комунікацію з затримкою до 100 мс. База даних PostgreSQL, розгорнута через Supabase і керована Drizzle ORM, забезпечує надійне зберігання даних із типобезпечними запитами, тоді як Redis оптимізує продуктивність шляхом кешування сесій, статусів онлайн і лічильників повідомлень.

Для деталізації структури та поведінки застосунку використано набір UML-діаграм. Діаграма класів описує основні сутності (users, groups, messages, attachments, messageChannels, directMessageInfos, groupInvites, accounts), що

відображають структуру бази даних і логіку взаємодії. Діаграма станів ілюструє плавні переходи між станами фронтенд-компонентів, такими як перегляд чатів, написання повідомлень чи обробка медіа. Діаграма розгортання демонструє інтеграцію фронтенду з хмарними сервісами (Supabase, Cloudinary, Ably, Redis) забезпечуючи стабільність і ізоляцію компонентів. Діаграма пакетів структурує код, виділяючи модулі frontend, api, database і services, що спрощує розробку та супровід. Swimlane-діаграма діяльності відображає взаємодію між користувачем, фронтендом, сервером і базою даних, підкреслюючи чіткий розподіл відповідальності.

Інтерактивні функції FlowChat, реалізовані через React-компоненти, включають відповіді на повідомлення (з використанням reply_id у messages), відображення статусу набору тексту (через Ably Realtime), модерацію груп (видалення повідомлень і блокування учасників через members) та пошук користувачів (за name у users). Ці функції підвищують залученість і забезпечують інтуїтивний досвід, підкріплений реалтайм-оновленнями та типобезпечними tRPC-запитами. Обробка медіа через Cloudinary з метаданими в attachments дозволяє швидко відображати контент, а адаптивний дизайн із CSS-модулями та анімаціями забезпечує зручність на різних пристроях.

Локалізація, реалізована за допомогою next-i18next, підтримує українську мову з можливістю додавання нових мов через структуровані файли перекладів (locales/uk.json). Це охоплює тексти, формати дат і підказки, адаптуючи інтерфейс до потреб української аудиторії. Планується підтримка RTL-мов для розширення аудиторії. Обмеженнями залишаються залежність від хмарних сервісів (Supabase, Cloudinary, Ably), що може впливати на операційні витрати, та потреба в стабільному інтернет-з’єднанні для реалтайм-функцій. У майбутньому передбачається впровадження додаткових можливостей, таких як підтримка ботів, реакції на повідомлення чи AI-аналітика для автодоповнення тексту, що зміцнить позиції FlowChat як сучасного, безпечного та конкурентоспроможного месенджера.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЧАТУ

4.1 Серверна частина застосунку

Після створення структури бази даних необхідно реалізувати серверну логіку FlowChat, яка забезпечує стабільну обробку запитів через API, високу швидкість виконання операцій і підтримку реалтайм-комунікації.

4.1.1 Організація серверного коду в Next.js

Серверна частина FlowChat розроблена на базі Next.js, використовуючи API Routes для створення ендпоінтів і tRPC для типобезпечної взаємодії між клієнтом і сервером. Цей підхід гарантує модульність, полегшує масштабування та інтеграцію з реалтайм-сервісами, зокрема Ably Realtime. Код структуровано за принципами поділу відповідальностей, що сприяє його зрозумілості та зручності в супроводі.

Основною одиницею є tRPC-роутери, які групують логіку за доменами:

– chatRouter – відповідає за роботу з повідомленнями (надсилання, оновлення, видалення, статус набору тексту);

– groupRouter – управляє групами (створення, оновлення, приєднання, видалення);

– dmRouter – обробляє прямі повідомлення (відкриття, закриття, отримання каналів);

– accountRouter – забезпечує управління профілем користувача (отримання даних, оновлення, видалення);

– uploadRouter – відповідає за завантаження медіафайлів через Cloudinary.

Роутери містять процедури, поділені на публічні та захищені (protectedProcedure). Захищені процедури використовують middleware для перевірки автентифікації через NextAuth, дозволяючи доступ лише авторизованим користувачам. Валідація вхідних даних здійснюється через Zod-схеми, наприклад, messageSchema для надсилання повідомлень або updateProfileSchema для оновлення профілю. Лістинг коду схеми повідомлень:

```
export const users = pgTable(
  "User",
  {
    id: varchar200("id").notNull().primaryKey(),
    name: varchar200("name").notNull().default("User"),
    banner_hash: integer("banner_hash"),
    emailVerified: timestamp("emailVerified"),
    image: varchar200("image"),
    email: varchar200("email"),
    password: varchar255("password"),
  },
  (table) => ({
    User_email_key: uniqueIndex("User_email_key").on(table.email),
  }),
);
```

Інтеграція з зовнішніми сервісами, такими як Redis для кешування (наприклад, last_read для позначення прочитаних повідомлень), Ably Realtime для доставки повідомлень і Cloudinary для управління медіафайлами, забезпечує додаткову функціональність. Файл ably/index.ts ініціалізує підключення до Ably, а schema.ts визначає канали та типи подій для реалтайм-оновлень (повідомлення, статуси, оновлення груп).

Центральний appRouter об’єднує всі роутери та експортує типи для типобезпеки на клієнті, що дозволяє фронтенду взаємодіяти з сервером через типізовані запити, мінімізуючи помилки.

4.1.2 Обробка запитів через tRPC

У FlowChat tRPC-роутери забезпечують маршрутизацію та виконання бізнес-логіки, поєднуючи ці функції в єдиній структурі, інтегрованій із Next.js API Routes.

Роутери відповідають за окремі доменні області:

– chatRouter обробляє надсилання повідомлень (send), отримання історії чатів (messages), оновлення (update), видалення (delete), а також реалтайм-функції, такі як статус набору тексту (type) і перевірка статусу онлайн (status);

– groupRouter управляє створенням груп (create), приєднанням до груп (join, joinByUniqueName), оновленням (update), видаленням (delete) і виходом із груп (leave);

– dmRouter відповідає за прямі повідомлення, включаючи відкриття (open), закриття (close) і отримання списку каналів (channels);

– accountRouter обробляє запити на отримання профілю (get, profile), оновлення даних (updateProfile) і видалення акаунта (delete);

– uploadRouter генерує підписи для завантаження медіа через Cloudinary (signAvatar, signGroupIcon, signAttachment).

Запити обробляються асинхронно через процедури tRPC, які використовують Drizzle ORM для роботи з PostgreSQL. Наприклад, у chatRouter.send після перевірки прав доступу створюється повідомлення, оновлюється статус каналу в directMessageInfos (для прямих повідомлень), а подія надсилається через Ably Realtime для миттєвого оновлення.

4.1.3 Механізми автентифікації та авторизації

Автентифікація та авторизація у FlowChat реалізовані через NextAuth із підтримкою OAuth 2.0 через Google (GoogleProvider) та входу за email/паролем (CredentialsProvider) (рис. 4.4). Конфігурація NextAuth визначена в authOptions, де адаптер (authAdapter) інтегрується з базою даних через Drizzle ORM.

OAuth-автентифікація через Google використовує змінні середовища OAUTH_GOOGLE_CLIENT_ID і OAUTH_GOOGLE_CLIENT_SECRET. Для входу через email/пароль пароль хешується за допомогою SHA-256. Адаптер authAdapter підтримує операції з користувачами: створення (createUser), оновлення (updateUser), видалення (deleteUser), а також управління сесіями (createSession, updateSession) і зв’язками з акаунтами (linkAccount, unlinkAccount).

Сесії зберігаються через стратегію JWT (session: { strategy: "jwt" }), а дані користувача передаються через токени та callbacks (session, jwt). Redis кешує верифікаційні токени (createVerificationToken, useVerificationToken), прискорюючи перевірку. Авторизація реалізується через protectedProcedure у tRPC для перевірки сесії, а також через функції перевірки прав (checkChannelPermissions, getMembership), що обмежують доступ до груп чи

повідомлень (наприклад, лише адміністратори можуть видаляти повідомлення інших). Лістинг коду authOptions:

```
export const authOptions: AuthOptions = {
  adapter: authAdapter,
  providers: [
    GoogleProvider({
      clientId: process.env.OAUTH_GOOGLE_CLIENT_ID!,
      clientSecret: process.env.OAUTH_GOOGLE_CLIENT_SECRET!,
    }),
    CredentialsProvider({
      id: "credentials",
      name: "Credentials",
      credentials: {
        email: { label: "Email", type: "email", placeholder: "your-email@gmail.com" },
        password: { label: "Password", type: "password" },
      },
      async authorize(credentials) {
        if (!credentials?.email || !credentials?.password) {
          return null;
        }

        const [user] = await db
          .select()
          .from(users)
          .where(eq(users.email, credentials.email));

        if (!user || !user.password) {
          return null;
        }

        // Verify password
        const hashedPassword = hashPassword(credentials.password);
        if (hashedPassword !== user.password) {
          return null;
        }

        return user;
      },
    }),
  ],
  pages: {
    signIn: "/auth/signin",
    newUser: "/?modal=new",
  },
  callbacks: {
    session: async ({ session, token }) => {
      if (session.user) {
        session.user.id = token.uid;
      }
      return session;
    },
    jwt: async ({ user, token }) => {
      if (user) {
```

```
        token.uid = user.id;
      }
      return token;
    },
  },
  session: {
    strategy: "jwt",
  },
};
```

Таким чином застосунок підтримує авторизацію з використанням credentials(пошта та пароль) та авторизація за допомогою сторонніх сервісів

4.1.4 Інтеграція з Ably Realtime

Реалтайм-комунікація у FlowChat забезпечується через Ably Realtime для миттєвої доставки повідомлень і оновлення статусів. Модуль ably/index.ts ініціалізує підключення до Ably, використовуючи ключ із змінної REALTIME_ABLY_KEY. У режимі розробки підключення кешується глобально для оптимізації. Модуль ably/schema.ts визначає канали (private, group, chat) та типи подій, такі як message_sent, group_updated, typing, із валідацією через Zod.

Події публікуються через функцію publish, яка надсилає повідомлення до відповідних каналів. Наприклад, у chatRouter.send після створення повідомлення подія message_sent відправляється через Ably до каналу chat:${channelId}, що дозволяє клієнтам оновлювати інтерфейс у реальному часі. Оновлення груп (group_updated) чи видалення повідомлень (message_deleted) також синхронізуються миттєво. Ably використовується для перевірки статусу онлайн (chatRouter.status) через API присутності (presence.get).

Redis зберігає тимчасові дані, такі як last_read для відстеження прочитаних повідомлень, зменшуючи навантаження на PostgreSQL. Асинхронна обробка та реалтайм-можливості Ably забезпечують швидку й надійну комунікацію, адаптовану для україномовних користувачів через локалізацію.

4.2 Клієнтська частина застосунку

Розробка серверної частини FlowChat супроводжується створенням інтуїтивного інтерфейсу та клієнтських сторінок, які інтегрують можливості бекенду. Клієнтська частина побудована на Next.js із використанням Server Components і Client Components для забезпечення продуктивності та масштабованості.

4.2.1 Структура Next.js

Клієнтська частина FlowChat структурована за принципом модульності, із чітким поділом на домени: (chat) для чатів, (app) для основного інтерфейсу, auth для автентифікації та settings для налаштувань. Компоненти, такі як MessageList, ChatInput і ChatViewport, реалізують логіку відображення та взаємодії, а стилізація виконана з використанням Tailwind CSS для адаптивності.

Основна взаємодія з API здійснюється через tRPC-клієнт (utils/trpc.ts), який забезпечує типобезпечні запити до серверних роутерів. Наприклад, у useProfile (hooks/use-profile.ts) використовується trpc.account.get.useQuery для отримання даних профілю.

4.2.2 Реалізація маршрутизації

Маршрутизація у FlowChat базується на App Router у Next.js, де структура маршрутів визначається через файлову систему. Групування в папці (chat)/(app) організує логіку чатів, а кожен маршрут асоціюється з відповідною сторінкою чи компонентом.

Основні маршрути:

– / – головна сторінка ((chat)/(app)/page.tsx), де відображаються списки прямих повідомлень (dmQuery) і груп (groups) з кнопками для створення (create-group) чи приєднання (join-group) через модальні вікна;

– /auth/signin – сторінка входу (auth/signin/page.tsx), що підтримує автентифікацію через форму (SignInForm) або зовнішні провайдери (наприклад, Google) за допомогою NextAuth;

– /auth/register – сторінка реєстрації (auth/register/page.tsx), із формою для створення користувача, яка взаємодіє з API-роутом /api/sign-up;

– /chat/[group] – динамічний маршрут для групового чату ((chat)/(app)/chat/[group]/page.tsx), де group є ідентифікатором групи; відображає повідомлення через MessageList і дозволяє надсилати нові через ChatInput;

– /dm/[channel] – маршрут для прямих повідомлень ((chat)/(app)/dm/[channel]/page.tsx), де channel є ідентифікатором каналу; відображає чат із користувачем через trpc.dm.info.useQuery;

– /settings – сторінка налаштувань ((chat)/(app)/settings/page.tsx), де користувач може редагувати профіль (UpdateProfile), змінювати тему (ThemeSwitcher) або видалити акаунт.

Динамічні маршрути використовують useParams із next/navigation. Наприклад, у /chat/[group] (рис. 4.5) параметр group передається в useGroupContext для отримання channel_id, а в /dm/[channel] параметр channel використовується для запиту даних. Лістинг коду /chat/[group]/page:

```
export default function Page() {
  const { data: session } = useSession();
  const { channel_id, member, owner_id } = useGroupContext();

  return (
    <>
      <div className="relative flex-1">
        <ChatArea
          deleteMessage={member.admin || owner_id === session?.user.id}
        >
          <MessageList channelId={channel_id} header={<Welcome />} />
        </ChatArea>
      </div>
      <ChatInput channelId={channel_id} />
    </>
  );
}
```

Компонент Nav у (chat)/(app)/layout.client.tsx генерує хлібні крихти (breadcrumb) залежно від поточного шляху. Наприклад, для /chat/[group] відображається назва групи з аватаром, а для /dm/[channel] – ім’я користувача.

Перевірка сесії виконується в Provider ((chat)/(app)/layout.client.tsx), який перенаправляє неавтентифікованих користувачів на /auth/signin через signOut({ redirect: true }). Серверна логіка в /auth/signin/page.tsx перенаправляє автентифікованих користувачів.

Обробка подій, таких як group_deleted, реалізується через Ably у utils функціях. Лістинг коду GroupEventManager:

```
export const usePageStore = create<PageStore>((set) => ({
  isSidebarOpen: false,
  setSidebarOpen: (value: boolean) => set({ isSidebarOpen: value }),
  messages: [],
  addMessage(info) {
    const message: ToastMessage = {
      id: Date.now(),
      variant: "red",
      ...info,
    };

    set((prev) => ({
      messages: [...prev.messages, message],
    }));
  },
  removeMessage(id) {
    set((state) => ({
      messages: state.messages.filter((msg) => msg.id !== id),
    }));
  },
  setModal: (type) => set({ modal: type }),
}));
```

Для невалідних шляхів Next.js повертає сторінку 404. API-роути /api/ably/auth і /api/trpc/[trpc] підтримують реалтайм-оновлення через Ably і tRPC-запити.

4.2.3 Реалтайм-інтеграція та Zustand

Реалтайм-функціональність забезпечується через Ably Realtime (utils/ably/client.ts), який обробляє події, такі як message_sent і typing. Компоненти

MessageEventManager, GroupEventManager і PrivateEventManager підписуються на канали Ably для оновлення стану. Стан застосунку управляється через Zustand.

Лістинг коду usePageStore:

```
export const usePageStore = create<PageStore>((set) => ({
  isSidebarOpen: false,
  setSidebarOpen: (value: boolean) => set({ isSidebarOpen: value }),
  messages: [],
  addMessage(info) {
    const message: ToastMessage = {
      id: Date.now(),
      variant: "red",
      ...info,
    };

    set((prev) => ({
      messages: [...prev.messages, message],
    }));
  },
  removeMessage(id) {
    set((state) => ({
      messages: state.messages.filter((msg) => msg.id !== id),
    }));
  },
  setModal: (type) => set({ modal: type }),
}));
```

4.3 Інтерфейс застосунку

Вебзастосунок FlowChat, розроблений для реалтайм-комунікації, пропонує користувачам зручні інтерфейси, які забезпечують доступ до чатів, профілів та групових функцій. Інтерфейси, реалізовані на Next.js із React-компонентами, стилізовані за допомогою Tailwind CSS і локалізовані українською мовою, адаптовані для різних пристроїв. Вони інтегруються з базою даних (users, groups, messages, messageChannels, directMessageInfos, groupInvites, members, accounts, attachments) через tRPC, Ably Realtime і Cloudinary, забезпечуючи швидке оновлення даних і реалтайм-взаємодію.

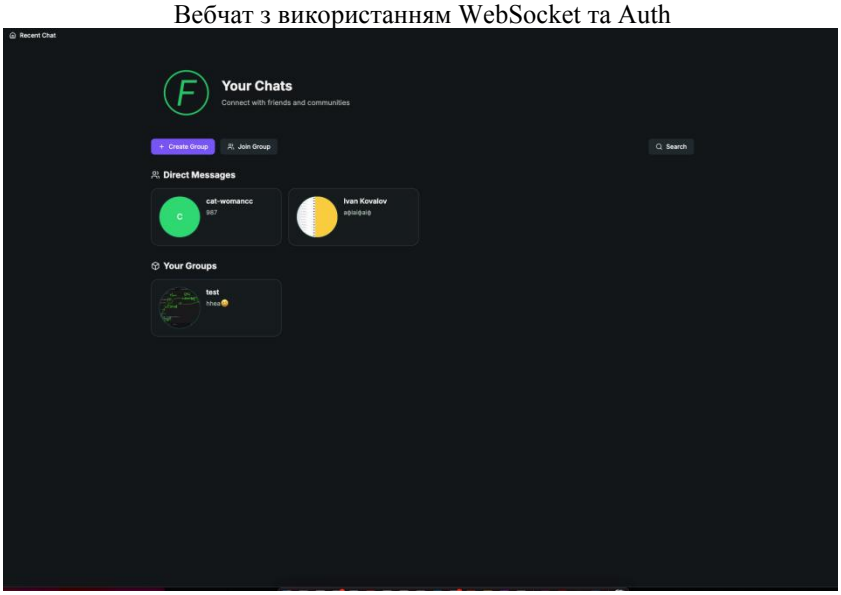

Рисунок 4.1 – Домашня сторінка

Лендінг FlowChat (рис. 4.1) зустрічає користувачів темним фоном із логотипом «F» у зеленому колі, що символізує бренд. У верхній частині розміщено навігаційне меню з кнопками «Create Group», «Join Group» і «Search», які ведуть до відповідних функцій. Центральна частина містить заголовок «Your Chats» із підзаголовком «Connect with friends and communities», а також блоки «Direct Messages» і «Your Groups», що відображають список чатів із messageChannels і groups. Дизайн підтримує адаптивність і локалізацію українською.

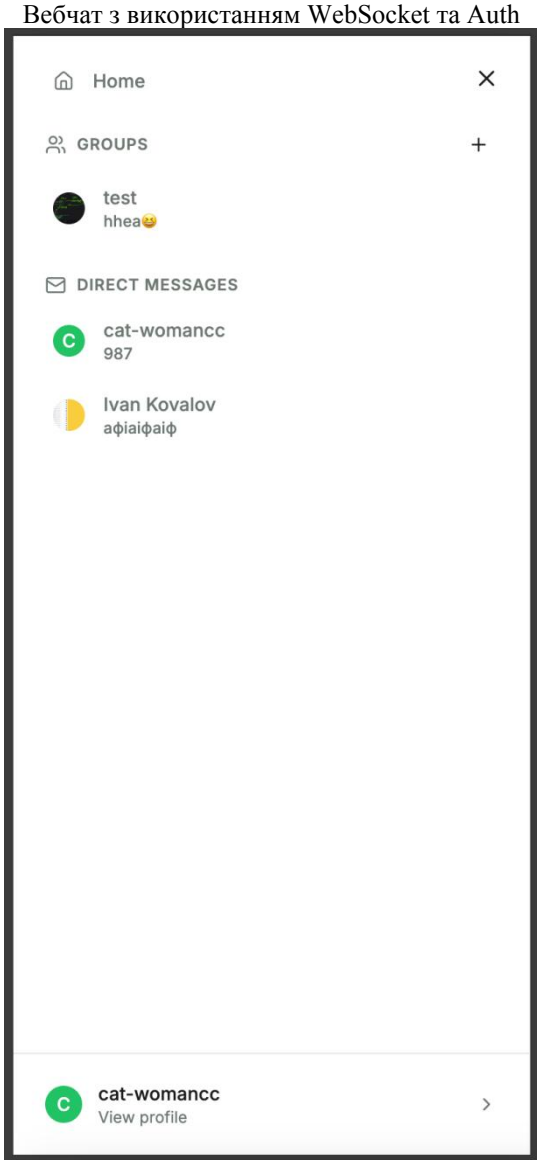

Рисунок 4.2 – Список чатів

Інтерфейс списку (рис. 4.2) чатів відображає перелік активних каналів, отриманих із messageChannels. У верхній частині розміщено заголовок із логотипом і кнопками «Create Group» та «Join Group». Розділи «Direct Messages» і «Your Groups» показують картки з іменами користувачів (users.name) або груп (groups.name), аватарками (users.image) і кількістю повідомлень (messages.count). Користувачі можуть переглядати статус онлайн, кешований у Redis, а кнопка «Search» активує пошук співрозмовників.

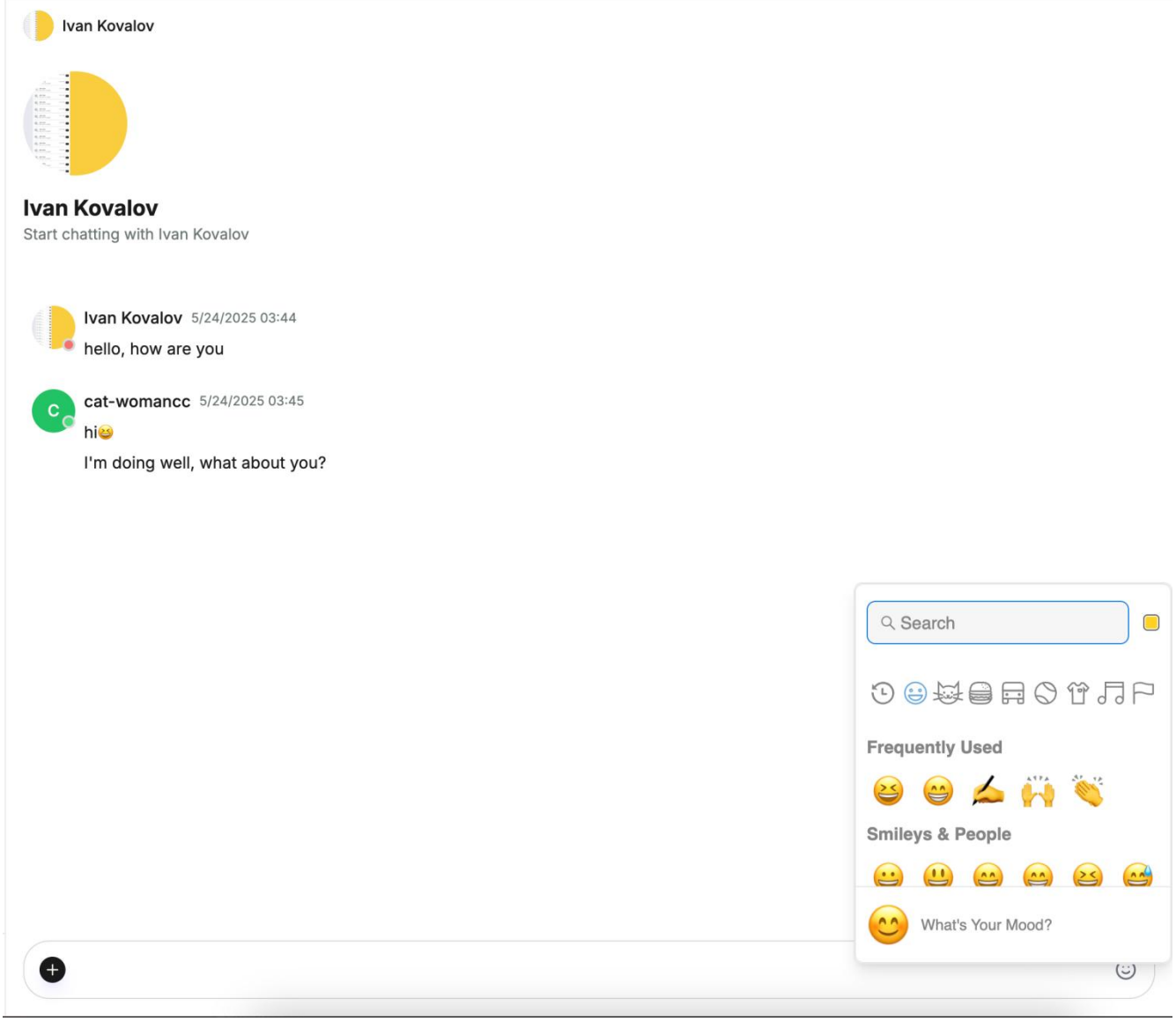

Рисунок 4.3 – Вікно чату

Вікно чату (рис. 4.3) відображає історію повідомлень із таблиці messages, де кожне повідомлення супроводжується аватаркою (users.image), ім’ям (users.name) і часом (messages.timestamp). У верхній частині показано назву чату та учасника, а внизу – поле введення тексту з кнопкою відправлення, інтегрованою через tRPC і Ably Realtime. Інтерфейс підтримує реалтайм-оновлення, а анімації додають інтерактивності.

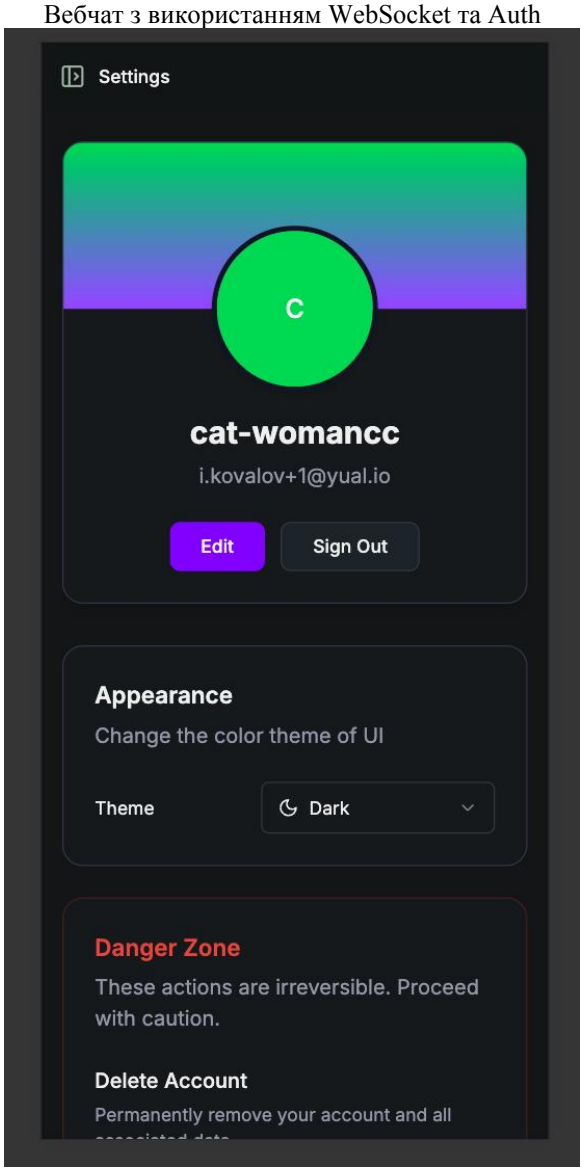

Рисунок 4.4 – Профіль користувача

Сторінка профілю (рис. 4.4) дозволяє користувачам редагувати дані з users. Інтерфейс «Profile customization» включає поля для зміни банера (banner_hash) і аватарки (image) через Cloudinary, а також введення нового username. Кнопки «Undo» і «Save» забезпечують управління змінами, а зелений градієнт у заголовку додає візуальної привабливості. Локалізація українською полегшує сприйняття.

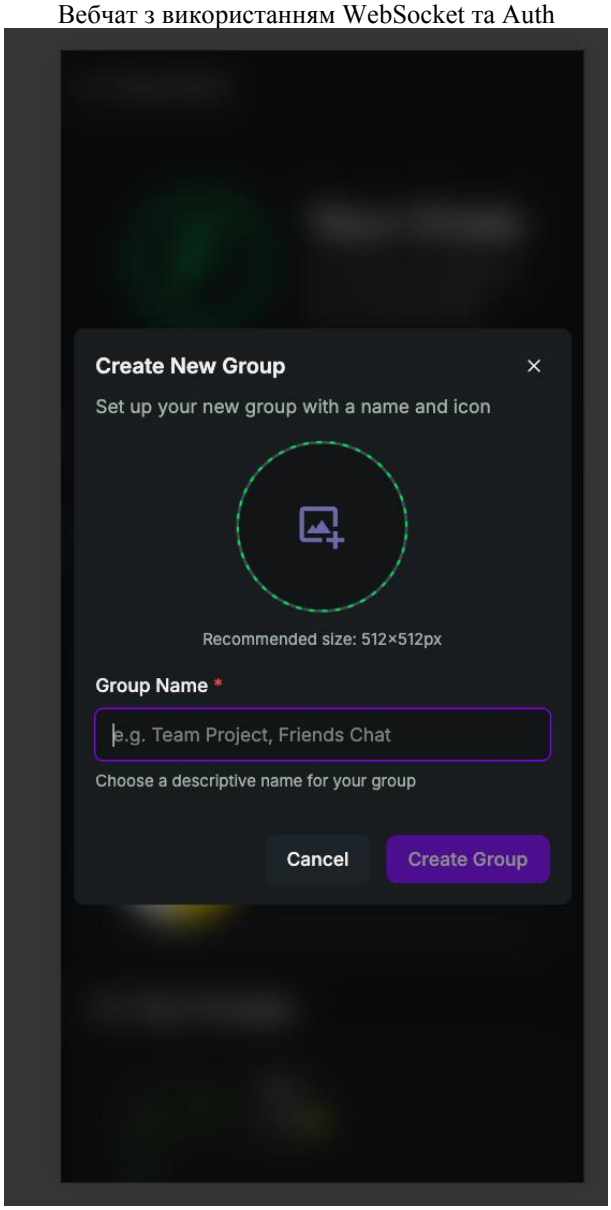

Рисунок 4.5 – Створення групового чату

Інтерфейс створення групи (рис. 4.5) пропонує користувачам форму для введення даних у groups. Поле «Group Name» дозволяє задати name, а кругла область із іконкою (icon_hash) підтримує завантаження зображення через Cloudinary з рекомендованим розміром 512x512 пікселів. Кнопка «Create Group» зберігає дані, а «Cancel» скасовує дію, забезпечуючи зручний процес.

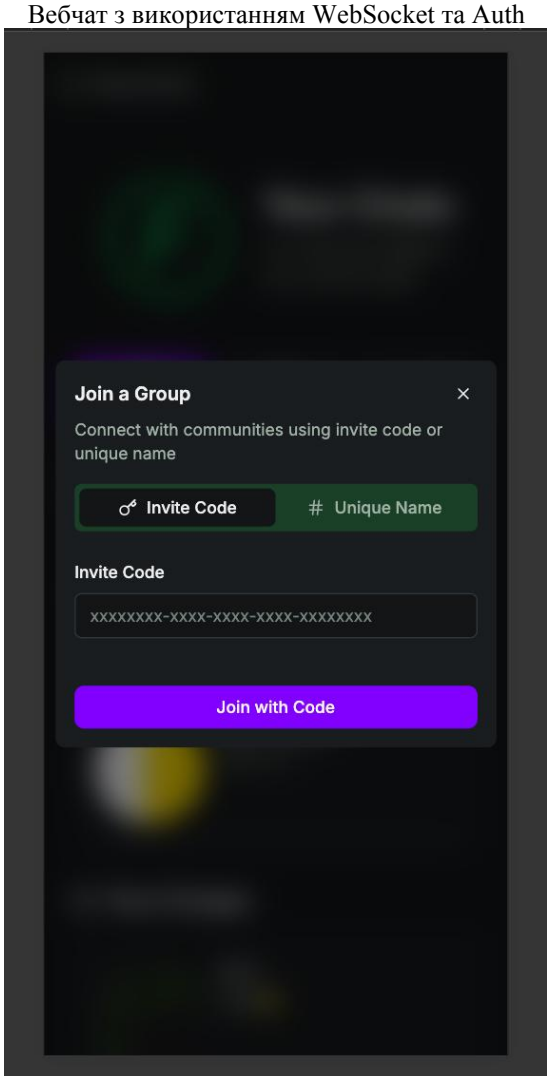

Рисунок 4.6 – Приєднання до групи

Інтерфейс приєднання до групи (рис. 4.6) відображає модальне вікно з полем для введення коду запрошення (groupInvites.code) або унікального імені (groups.unique_name). Кнопка «Join with Code» активує процес через tRPC, а перемикачі «Invite Code» і «Unique Name» дозволяють обрати метод підключення. Темний фон із контрастними акцентами полегшує читання.

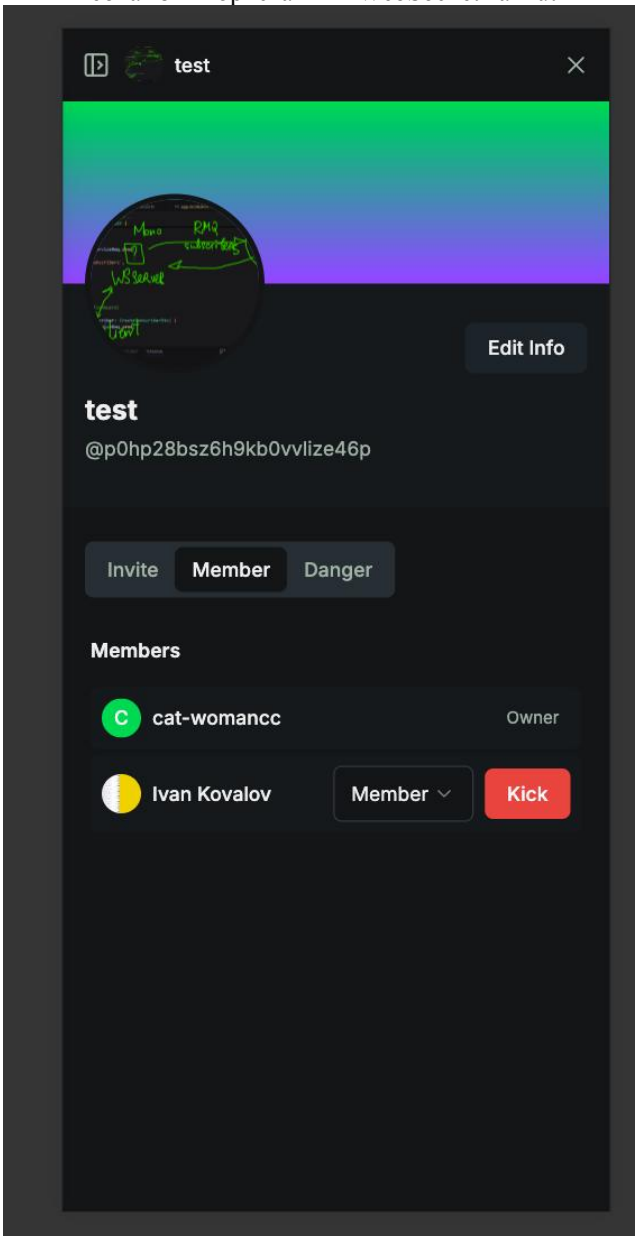

Рисунок 4.7 – Налаштування модерації

Інтерфейс налаштувань групи (рис. 4.7) відображає список членів (members) із ролями, такими як «Owner» і «Member». Користувачі можуть запрошувати нових учасників («Invite»), змінювати ролі («Member») або видаляти їх («Kick») через members.admin. Кнопка «Edit Info» дозволяє редагувати groups.name, а градієнтний фон додає візуальної глибини. Інтеграція з Ably забезпечує реалтайм-оновлення.

Ці інтерфейси створюють цілісний досвід реалтайм-комунікації, інтегруючись із tRPC, Ably Realtime і Cloudinary. Локалізація українською підвищує доступність. Обмеження включають залежність від хмарних сервісів і

потребу в стабільному інтернеті. У майбутньому планується додавання функцій, таких як реакції на повідомлення чи AI-підтримка.

4.4 Якість ПЗ

Оцінка якості FlowChat проводиться через аналіз продуктивності та відповідність стандартам UI/UX. Для оцінки продуктивності використовується Lighthouse, який аналізує швидкість завантаження, доступність, PWA та SEO. Наприклад, головна сторінка (/) завдяки ледачому завантаженню (Spinner) і оптимізованому рендерингу (Avatar) має високі оцінки за швидкість.

Додатково, якість перевіряється через:

– Зручність навігації: хлібні крихти в Nav (layout.client.tsx) допомагають орієнтуватися між маршрутами.;

– Адаптивність: Tailwind CSS забезпечує коректне відображення на різних пристроях, наприклад, grid-cols-1 sm:grid-cols-2 у page.tsx;

– Час відгуку: реалтайм-оновлення через Ably (наприклад, message_sent) забезпечують миттєве відображення повідомлень;

Lighthouse (рис. 4.8) використовується для комплексного аудиту вебсторінки, що дозволяє оцінити її швидкодію, виявити бар’єри для користувачів з інвалідністю, перевірити відповідність принципам прогресивних вебдодатків (PWA), а також знайти недоліки у сфері пошукової оптимізації та надати поради щодо підвищення загальної ефективності сайту.

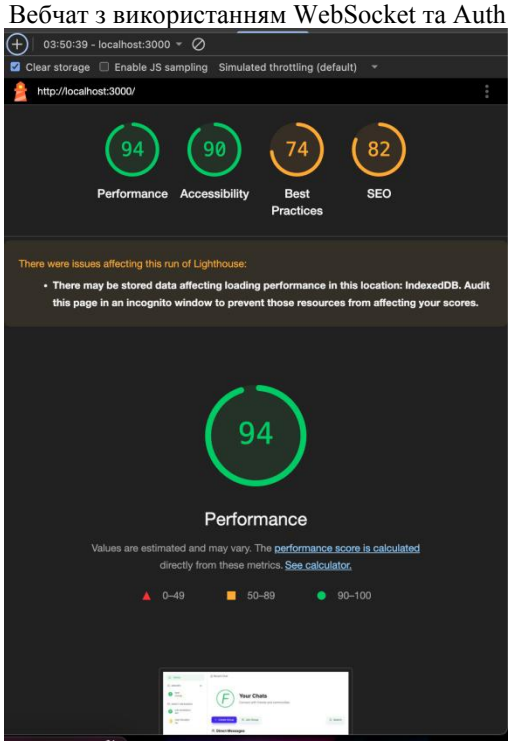

Рисунок 4.8 – Результати продуктивності

4.5 Оптимізація продуктивності

Оптимізація продуктивності FlowChat спрямована на підвищення швидкості завантаження та зменшення навантаження на клієнт, що є ключовим для забезпечення плавного досвіду користувачів, особливо в умовах активного використання чатів із великою кількістю повідомлень. Одним із основних підходів є ледаче завантаження компонентів, яке дозволяє відкладати рендеринг ресурсоємних елементів до моменту їхньої потреби. Наприклад, на сторінці /((chat)/(app)/page.tsx) компоненти Spinner і Avatar завантажуються лише при необхідності, що видно з умовного рендерингу, де Spinner відображається під час завантаження даних dmQuery.isLoading, а списки чатів і груп з’являються лише після завершення цього процесу, що значно скорочує початковий час завантаження сторінки. Ця стратегія особливо ефективна для користувачів із повільним інтернет-з’єднанням, дозволяючи їм одразу бачити базовий вміст, а не чекати повного рендерингу.

Іншим важливим аспектом є кешування запитів через tRPC-клієнт, яке зменшує кількість повторних звернень до серверу, підвищуючи швидкість відгуку

застосунку. Наприклад, у /dm/[channel] дані про користувача кешуються після першого успішного запиту query.isSuccess, що дозволяє уникнути зайвих мережевих запитів при перемиканні між прямими чатами, зберігаючи при цьому актуальність інформації. Ця оптимізація особливо корисна для реалтайм-систем, де швидке оновлення даних критично важливе. Крім того, оптимізація зображень відіграє значну роль у підвищенні продуктивності, адже компоненти Avatar і HeroImage використовують оптимізовані зображення через userBanners.url і groupIcon.url, що зменшує розмір файлів і прискорює їх завантаження, як видно в (chat)/(app)/settings/page.tsx для банера профілю, де зображення адаптується до розміру h-32 object-cover.

Мінімізація рендерів також є важливим елементом оптимізації, і тут використовується Zustand у usePageStore, який дозволяє уникати зайвих оновлень компонентів, що позитивно впливає на продуктивність, особливо на слабших пристроях. Наприклад, у (chat)/(app)/page.tsx модальні вікна відкриваються лише при зміні стану modal, що виключає непотрібні перерисовки інтерфейсу під час взаємодії з кнопками, такими як Create Group. Цей підхід забезпечує економію ресурсів і покращує плавність анімацій. Загалом, ці заходи створюють основу для швидкого й ефективного застосунку, який здатен обробляти великі обсяги даних без втрати якості користувацького досвіду, адаптуючись до різних сценаріїв використання, від одночасного перегляду кількох чатів до роботи з медіафайлами в групах.

Висновки до розділу 4

Розробка та тестування вебзастосунку FlowChat для реалтайм-комунікації забезпечили створення високопродуктивної, безпечної та зручної системи, адаптованої до потреб користувачів, зокрема української аудиторії. Інтерфейс застосунку, побудований на Next.js із використанням React-компонентів і стилізований через Tailwind CSS, пропонує адаптивний і локалізований українською мовою користувацький досвід. Інтерфейси, включаючи лендінг, список чатів, вікно чату, профіль користувача, створення та приєднання до груп, а

також налаштування модерації, інтегруються з базою даних (таблиці users, groups, messages, attachments, messageChannels, directMessageInfos, groupInvites, members, accounts) через tRPC, Ably Realtime і Cloudinary. Це забезпечує швидке оновлення даних, реалтайм-взаємодію, плавні анімації та інтуїтивну навігацію, що підвищує залученість користувачів.

Серверна частина, реалізована на Next.js із API Routes і tRPC-роутерами (chatRouter, groupRouter, dmRouter, accountRouter, uploadRouter), забезпечує типобезпечну обробку запитів із чітким розподілом логіки за доменами, такими як управління повідомленнями, групами, прямими чатами, профілями та медіафайлами. Інтеграція з Ably Realtime дозволяє доставляти повідомлення й оновлення статусів (наприклад, message_sent, typing, group_updated) із затримкою до 100 мс, а Redis кешує тимчасові дані, такі як сесії, статуси онлайн і лічильники прочитаних повідомлень (last_read), зменшуючи навантаження на PostgreSQL. Автентифікація через NextAuth із підтримкою OAuth 2.0 (GoogleProvider) і CredentialsProvider, разом із protectedProcedure та middleware для перевірки прав, гарантує безпеку доступу. Cloudinary оптимізує обробку медіафайлів, зберігаючи метадані в attachments і надаючи URL для швидкого відображення.

Клієнтська частина структурована модульно з використанням App Router у Next.js, що підтримує динамічні маршрути (/chat/[group], /dm/[channel], /settings) і комбінацію Server та Client Components для оптимізації продуктивності. Компоненти, такі як MessageList, ChatInput і ChatViewport, забезпечують відображення чатів і взаємодію, а Zustand (usePageStore) мінімізує рендери, зберігаючи стан інтерфейсу. tRPC-клієнт кешує запити, наприклад, для профілів чи списків чатів, прискорюючи відгук. Реалтайм-функції, реалізовані через Ably (MessageEventManager, GroupEventManager, PrivateEventManager), забезпечують миттєве оновлення повідомлень і статусів. Оптимізація продуктивності включає ледаче завантаження (Spinner, Avatar), оптимізацію зображень через Cloudinary (userBanners.url, groupIcon.url) і зменшення рендерів, що гарантує швидкість навіть на слабких пристроях.

Оцінка якості за допомогою Lighthouse підтвердила високу продуктивність, відповідність стандартам PWA, SEO та доступності. Адаптивний дизайн через Tailwind CSS забезпечує коректне відображення на різних пристроях, а хлібні крихти (Nav) полегшують навігацію. Локалізація українською мовою, реалізована через next-i18next із файлами перекладів (locales/uk.json), охоплює тексти, формати дат і підказки, підвищуючи доступність для цільової аудиторії. Планується підтримка RTL-мов для розширення охоплення. Обмеженнями є залежність від хмарних сервісів (Supabase, Cloudinary, Ably), що впливає на операційні витрати, і потреба в стабільному інтернет-з’єднанні для реалтайм-функцій. У майбутньому передбачається впровадження додаткових можливостей, таких як реакції на повідомлення, підтримка ботів, AI-аналітика для автодоповнення тексту чи аналізу чатів, що зміцнить позиції FlowChat як сучасного, масштабованого та конкурентоспроможного месенджера.

ВИСНОВКИ

У кваліфікаційній бакалаврській роботі здійснено повний цикл створення вебзастосунку FlowChat, орієнтованого на забезпечення безпечної та швидкої реалтайм-комунікації з урахуванням потреб українського суспільства в умовах воєнних реалій і кіберзагроз. На першому етапі проведено аналіз предметної області, що виявив недоліки існуючих платформ, таких як WhatsApp, Telegram і Discord, зокрема їхню обмежену адаптацію до локального контексту. Це стало основою для формування вимог до системи, розробки її функціональної та інформаційної моделей.

У проєктній частині визначено архітектуру на базі Next.js, Supabase і Ably Realtime, створено прототип інтерфейсу з українською локалізацією та реалізовано механізми реалтайм-обміну повідомленнями. Кодування включало побудову бази даних із підтримкою індексації та кешування, серверну частину з автентифікацією через NextAuth, інтеграцію з хмарними сервісами та клієнтський інтерфейс з адаптивним дизайном. Якість застосунку підтверджена тестуванням, а контейнеризація забезпечила зручність розгортання.

Розроблений FlowChat відповідає сучасним вимогам до безпеки, продуктивності та зручності, пропонуючи локалізоване рішення для комунікації. У рамках поставленої мети виконано завдання:

– проаналізовано ринок месенджерів, визначивши потреби користувачів;

– спроєктовано архітектуру з урахуванням вимог до безпеки та реального часу;

– створено базу даних із підтримкою масштабування;

– реалізовано серверну логіку з автентифікацією та реалтайм-функціями;

– розроблено зручний інтерфейс із локалізацією.

Проєкт став важливим кроком у професійному розвитку, заклавши основу для подальших досліджень і вдосконалень, таких як додавання нових функцій чи інтеграція передових технологій..

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Ivanova O., Petrova L. Impact of War on Digital Communication in Ukraine: Sociological Insights 2023–2025. Kyiv: Ukrainian Sociological Institute. 2025. P. 78–85.

2. Cybersecurity and Infrastructure Security Agency (CISA). 2024 Report on DDoS Attacks and Cyber Threats. URL: https://www.cisa.gov/reports, (last accessed: 02.05.2025).

3. European Union. General Data Protection Regulation (GDPR). Official Journal of the European Union, L 119. 2016. URL: https://eur-lex.europa.eu/eli/reg/2016/679/oj, (last accessed: 01.05.2025).

4. IETF. RFC 6455 - The WebSocket Protocol. URL: https://datatracker.ietf.org/doc/html/rfc6455, (last accessed: 01.05.2025).

5. Verkhovna Rada of Ukraine. Law No. 2297-VIII on Cybersecurity. 2017. URL: https://zakon.rada.gov.ua/laws/show/2163-19, (last accessed: 01.05.2025).

6. Official Next.js Documentation. URL: https://nextjs.org/docs, (last accessed: 03.05.2025).

7. PostgreSQL Documentation. URL: https://www.postgresql.org/docs/, (last accessed: 02.05.2025).

8. Supabase Documentation. URL: https://supabase.com/docs, (last accessed: 02.05.2025).

9. WhatsApp Security Whitepaper. URL: https://www.whatsapp.com/security, (last accessed: 03.05.2025).

10. Telegram. FAQ. URL: https://telegram.org/faq, (last accessed: 03.05.2025).

11. Discord. Beginner's Guide to Discord. URL: https://support.discord.com/hc/en-us/articles/360045138571-Beginner-s-Guide-to-Discord, (last accessed: 03.05.2025).

12. DrizzleORM Documentation. URL: https://orm.drizzle.team/docs/overview, (last accessed: 20.05.2025).

13. Cloudinary Documentation. URL: https://cloudinary.com/documentation, (last accessed: 22.05.2025).

14. tRPC Documentation. URL: https://trpc.io/docs/, (last accessed: 16.05.2025).

15. Redis Documentation. URL: https://redis.io/docs/latest/, (last accessed: 12.05.2025).

16. Ably Documentation. URL: https://ably.com/docs, (last accessed: 03.05.2025).

17. NextAuth Documentation. URL: https://next-auth.js.org/getting-started/introduction, (last accessed: 09.05.2025).

18. Ю. Форкун, В. Мартинюк, О. Яшина. Метод розробки та проектування архітектурної складової програмного застосунку. 2023. № 4. с. 87-93. doi : 10.31891/2219-9365-2023-76-11.

19. NextI18n App Router. URL: https://www.locize.com/blog/i18n-next-app-router, (last accessed: 18.05.2025).

ДОДАТОК А

Схема бази данних

```
const len32 = { length: 32 };
const len200 = { length: 200 };
const len255 = { length: 255 };
const len256 = { length: 256 };
const varchar32 = (name: string) => varchar(name, len32);
const varchar200 = (name: string) => varchar(name, len200);
const varchar255 = (name: string) => varchar(name, len255);
const varchar256 = (name: string) => varchar(name, len256);

export type Embed = {
  url: string;
  title?: string;
  description?: string;
  image?: {
    url: string;
    width: number;
    height: number;
  };
};

export const attachmentType = pgEnum("attachment_type", [
  "image",
  "video",
  "raw",
]);

export const accounts = pgTable(
  "Account",
  {
    id: varchar32("id").notNull().primaryKey(),
    type: varchar200("type").notNull(),
    userId: varchar32("userId").notNull(),
    providerAccountId: varchar200("providerAccountId").notNull(),
    provider: varchar200("provider").notNull(),
    access_token: text("access_token"),
    refresh_token: text("refresh_token"),
    token_type: varchar("token_type", { length: 30 }),
    expires_at: integer("expires_at"),
    id_token: text("id_token"),
    scope: varchar200("scope"),
    session_state: varchar200("session_state"),
  },
  (table) => ({
    Account_provider_providerAccountId_key: uniqueIndex(
      `Account_provider_providerAccountId_key`,
    ).on(table.provider, table.providerAccountId),
    Account_userId_idx: index("Account_userId_idx").on(table.userId),
  }),
);
```

```
export const directMessageInfos = pgTable(
  "DirectMessageInfo",
  {
    user_id: varchar200("user_id").notNull(),
    channel_id: varchar32("channel_id").notNull(),
    open: boolean("open").default(true),
    to_user_id: varchar200("to_user_id").notNull(),
  },
  (table) => ({
    cpk: primaryKey({ columns: [table.user_id, table.to_user_id] }),
    channel_id_key: index("channel_id_key").on(table.channel_id),
  }),
);

export const ChannelType = pgEnum("channel_type", ["DM", "GROUP"]);

export const messageChannels = pgTable("MessageChannel", {
  id: varchar32("id").primaryKey(),
  group_id: varchar32("group_id"),
  type: ChannelType("type").notNull(),
  last_message_id: integer("last_message_id"),
});

export const groups = pgTable(
  "Group",
  {
    id: varchar32("id").notNull().primaryKey(),
    icon_hash: integer("icon_hash"),
    unique_name: varchar32("unique_name").notNull(),
    public: boolean("public").notNull().default(false),
    banner_hash: integer("banner_hash"),
    owner_id: varchar200("owner_id").notNull(),
    name: varchar256("name").notNull(),
    channel_id: varchar32("channel_id").notNull().default(""),
  },
  (table) => ({
    Group_unique_name_key: uniqueIndex("Group_unique_name_key").on(table.unique_name),
    Group_channel_idx: index("Group_channel_idx").on(table.channel_id),
  }),
);

export const groupInvites = pgTable(
  "GroupInvite",
  {
    code: varchar200("code").notNull().primaryKey(),
    group_id: varchar32("group_id").notNull(),
  },
  (table) => ({
    group_idx: index("GroupInvite_group_id_idx").on(table.group_id),
  }),
);

export const members = pgTable(
  "Member",
  {
```

```
    user_id: varchar200("user_id").notNull(),
    admin: boolean("is_admin").notNull().default(false),
    group_id: varchar32("group_id").notNull(),
  },
  (table) => ({
    cpk: primaryKey({ columns: [table.group_id, table.user_id] }),
    Member_group_id_idx: index("Member_group_id_idx").on(table.group_id),
    Member_user_id_idx: index("Member_user_id_idx").on(table.user_id),
  }),
);

export const messages = pgTable(
  "Message",
  {
    id: serial("id").notNull().primaryKey(),
    reply_id: integer("reply_id"),
    attachment_id: varchar32("attachment_id"),
    author_id: varchar200("author_id").notNull(),
    content: varchar("content", { length: 2000 }).notNull(),
    embeds: json("embeds").$type<Embed[]>(),
    timestamp: timestamp("timestamp", { withTimezone: true })
      .notNull()
      .defaultNow(),
    channel_id: varchar32("channel_id").notNull(),
  },
  (table) => ({
    Message_channel_id_idx: index("Message_channel_id_idx").on(table.channel_id),
    Message_timestamp_idx: index("Message_timestamp_idx").on(table.timestamp),
  }),
);

export const sessions = pgTable(
  "Session",
  {
    id: varchar200("id").notNull().primaryKey(),
    expires: timestamp("expires").notNull(),
    sessionToken: varchar200("sessionToken").notNull(),
    userId: varchar200("userId").notNull(),
  },
  (table) => ({
    Session_sessionToken_key:
uniqueIndex("Session_sessionToken_key").on(table.sessionToken),
    Session_userId_idx: index("Session_userId_idx").on(table.userId),
  }),
);

export const users = pgTable(
  "User",
  {
    id: varchar200("id").notNull().primaryKey(),
    name: varchar200("name").notNull().default("User"),
    banner_hash: integer("banner_hash"),
    emailVerified: timestamp("emailVerified"),
    image: varchar200("image"),
    email: varchar200("email"),
```

```
    password: varchar255("password"),
  },
  (table) => ({
    User_email_key: uniqueIndex("User_email_key").on(table.email),
  }),
);

export const attachments = pgTable("Attachment", {
  id: varchar32("id").notNull().primaryKey(),
  url: varchar255("url").notNull(),
  name: varchar255("name").notNull(),
  bytes: integer("bytes").notNull(),
  width: integer("width"),
  height: integer("height"),
  type: attachmentType("type"),
});
```

ДОДАТОК Б

Лістинг коду useMessageStore

```
export type MessagePlaceholder = {
  data: SendData;
  reply?: MessageType;
  error?: string;
  nonce: number;
};

export interface TypingUser {
  timestamp: number;
  user: UserProfile;
}

export type ChatStore = {
  ably?: Realtime;
  sending: {
    [id: string]: MessagePlaceholder[];
  };
  editing: {
    [id: string]: { messageId?: number };
  };
  messages: {
    [id: string]: MessageType[];
  };
  status: Record<
    string,
    {
      type: "online" | "offline";
    }
  >;
  typing: Map<string, TypingUser[]>;
  reply: Map<string, MessageType>;
  pointer: Map<string, number>;
  setEditing(channelId: string, messageId?: number): void;
  updatePointer(channelId: string): void;
  updateReply(channelId: string, data: MessageType | null): void;
  setUserTyping(channelId: string, user: UserProfile): void;
  addSending: (
    id: string,
    data: SendData,
    reply?: MessageType,
  ) => MessagePlaceholder;
  errorSending: (id: string, nonce: number, message: string) => void;
  removeSending: (id: string, nonce: number) => void;
};

export const useMessageStore = create<ChatStore>((set) => ({
  sending: {},
  editing: {},
  messages: {},
  pointer: new Map(),
```

```
reply: new Map(),
typing: new Map(),
status: {},
updatePointer(channelId) {
  set((prev) => {
    const next = new Map(prev.pointer);
    const messages = prev.messages[channelId];
    if (messages && messages.length > 0)
      next.set(channelId, new Date(messages[0].timestamp).getTime());

    return {
      pointer: next,
    };
  });
},
setUserTyping(channelId: string, user: UserProfile) {
  set((prev) => {
    const next = new Map(prev.typing);
    let channel = next.get(channelId) ?? [];
    channel = channel.filter((item) => item.user.id !== user.id);
    channel.push({ user, timestamp: Date.now() });
    next.set(channelId, channel);

    return {
      typing: next,
    };
  });
},
setEditing(channelId: string, messageId?: number) {
  set((prev) => ({
    editing: {
      ...prev.editing,
      [channelId]: { messageId },
    },
  }));
},
addSending: (group, data, reply) => {
  const item: MessagePlaceholder = {
    data,
    reply,
    nonce: Date.now(),
  };

  addNonce(item.nonce);
  set((prev) => {
    return {
      sending: {
        ...prev.sending,
        [group]: [...(prev.sending[group] ?? []), item],
      },
    };
  });

  return item;
},
```

```
  updateReply(channelId: string, data: MessageType | null) {
    set((prev) => {
      const next = new Map(prev.reply);
      if (!data) next.delete(channelId);
      else next.set(channelId, data);

      return {
        reply: next,
      };
    });
  },
  errorSending(group, nonce, message) {
    set((prev) => {
      const updated = prev.sending[group]?.map((item) =>
        item.nonce === nonce ? { ...item, error: message } : item,
      );

      return {
        sending: {
          ...prev.sending,
          [group]: updated,
        },
      };
    });
  },
  removeSending(group, nonce) {
    removeNonce(nonce);
    set((prev) => {
      const filtered = prev.sending[group]?.filter(
        (item) => item.nonce !== nonce,
      );

      return {
        sending: {
          ...prev.sending,
          [group]: filtered,
        },
      };
    });
  },
}));
```