МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет імені Петра Могили

Факультет комп'ютерних наук

Кафедра інженерії програмного забезпечення

ДОПУЩЕНО ДО ЗАХИСТУ

Завідувач кафедри_____ Є. О. Давиденко
підпис
«___» _____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Реалізація чату за допомогою вебсокетів

Спеціальність «Інженерія програмного забезпечення»

121 – КРБ.1 – 409.22010905

Здобувач _____І. В. Борецький
підпис
«__» _____ 2024 р.

Керівник канд. техн. наук, доцент _____Є. О. Давиденко
підпис
«__» _____ 2024 р.

Консультант канд. техн. наук, доцент _____А. О. Алексєєва
підпис
«__» _____ 2024 р.

Миколаїв – 2024

Кафедра інженерії програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри

інженерії програмного

забезпечення, канд.

техн. наук, доцент,

_____Є. О. Давиденко

«19» січня ____2024 р.

ЗАВДАННЯ

на виконання кваліфікаційної роботи бакалавра

Видано студенту групи 409 факультету комп'ютерних наук

Борецькому Івану Володимировичу

(прізвище, ім'я, по батькові студента)

1. Тема кваліфікаційної роботи

«Реалізація чату за допомогою вебсокетів»

Затверджена наказом по ЧНУ від «22» грудня 2023 р. № 269

2. Строк представлення кваліфікаційної роботи «___»_____2024 р.

3. Очікуваний результат роботи та початкові дані, якщо такі потрібні

Очікуваним результатом є працездатний чат-застосунок з реалізацією в реальному часі за допомогою вебсокетів.

4. Перелік питань, що підлягають розробці

Аналіз предметної області, порівняльний аналіз аналогів, визначення вимог та функціоналу системи, моделювання, прєктування програмного застосунку, кодування та тестування застосунку.

5. Перелік графічних матеріалів:

Презентація

6. Завдання до спеціальної частини

7. Консультанти:

Консультант	Кафедра (організація)	Частина роботи
Алєксєєва А. О.	Кафедра екології	Спеціальна частина з охорони праці

Керівник роботи
кандидат технічних наук, доцент Давиденко Євген Олександрович
(посада, прізвище, ім'я, по батькові)

Завдання прийнято до виконання _____
(підпис)

Борецький Іван Володимирович
(прізвище, ім'я, по батькові студента)

Дата видачі завдання «19» січня 2024 р. _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

Виконання кваліфікаційної роботи

Тема: «Реалізація чату за допомогою вебсокетів»

№	Найменування роботи	Початок	Закінчення	Примітки
1	Розробка та затвердження завдання на виконання КРБ	14.01.24	19.01.24	Виконано
2	Огляд літератури за темою роботи	20.01.24	22.01.24	Виконано
3	Складання календарного плану КРБ	23.01.24	24.01.24	Виконано
4	Аналіз системи, що розробляється	25.01.24	10.02.24	Виконано
5	Розробка вимог до системи	11.02.24	19.02.24	Виконано
6	Аналіз ринку наявних технологій і рішень	20.02.24	01.03.24	Виконано
7	Проєктування структури бази даних та архітектури системи	02.03.24	02.04.24	Виконано
8	Реалізація, тестування та налагодження системи	03.04.24	03.05.24	Виконано
9	Розробка спеціальної частини з охорони праці	04.05.24	05.05.24	Виконано
10	Відгук керівника КРБ	06.05.24	07.05.24	Виконано
11	Оформлення КРБ та презентації	08.05.24	09.05.24	Виконано
12	Попередній захист	05.06.24	05.06.24	Виконано
13	Рецензування	10.06.24	10.06.24	Виконано
14	Завершення оформлення КРБ та презентації	11.06.24	15.06.24	Виконано
15	Захист кваліфікаційної роботи	25.06.24	27.06.24	Виконано

Розробив здобувач Борецький Іван Володимирович

(прізвище, ім'я, по батькові студента)(*підпис*)

«19» січня 2024 р.

Керівник роботи кандидат технічних наук, доцент Давиденко Євген Олександрович

(посада, прізвище, ім'я, по батькові)(*підпис*)

«19» січня 2024 р.

АНОТАЦІЯ

до кваліфікаційної роботи бакалавра

«Реалізація чату за допомогою вебсокетів»

Здобувач 409 гр.: Борецький Іван Володимирович

Керівник: канд. техн. наук, доцент Давиденко Є. О.

Сучасне суспільство відзначається стрімким розвитком інформаційних технологій, які трансформують традиційні форми комунікації та сприяють швидкому доступу до інформації через різноманітні вебзастосунки. Серед передових технологій, що забезпечують миттєве спілкування через ці застосунки, вебсокети виступають як важливий інструмент для створення чатів та інших форм взаємодії в реальному часі.

Актуальність розробки програмного забезпечення «Реалізація чату за допомогою вебсокетів» посилюється у підтримці корпоративних комунікаційних систем, де потреба в надійному та ефективному обміні інформацією є критичною для оперативного управління проєктами і процесами. Чат на базі вебсокетів забезпечує миттєве доставлення повідомлень та можливість ефективної взаємодії в режимі реального часу, що є ключовим для сучасних комунікаційних сервісів.

Об’єктом кваліфікаційної роботи є процес взаємодії в реальному часі через вебзастосунок для миттєвого обміну повідомленнями на базі вебсокетів.

Предметом кваліфікаційної роботи є засоби розробки і впровадження вебсокет-технологій для швидкісної та безпечної онлайн-комунікації.

Метою кваліфікаційної роботи є розробка високоефективного, надійного чат-застосунку на базі вебсокетів для комунікації в реальному часі.

Для досягнення цієї мети визначено такі завдання:

1) дослідження предметної галузі;
2) розробка інтерфейсу для миттєвого обміну повідомленнями;
3) проєктування вебзастосунку;
4) реалізація вебзастосунку;
5) впровадження шифрування для забезпечення безпеки даних;

6) тестування розробленого вебзастосунку.

У першому розділі проведено детальний аналіз предметної області чат-застосунків. Вивчались вимоги користувачів та проведено порівняльний аналіз існуючих аналогів на ринку. Також розглянуто структурні особливості сучасних чат-застосунків та виявлено їхні сильні та слабкі сторони, зокрема архітектурні моделі, функціональність, інтерфейс користувача, аспекти безпеки та надійності.

У другому розділі описано процес проєктування та моделювання програмного забезпечення для вебзастосунку онлайн-комунікації користувачів. Це включало створення архітектурних діаграм, моделювання системи чат-застосунку, визначення ролей користувачів та адміністраторів, розробку діаграм варіантів використання та сценаріїв взаємодії, а також логічну та фізичну моделі бази даних для ефективного зберігання та управління даними.

У третьому розділі описано основні кроки проєктування системи чат-застосунку, включаючи створення діаграм для візуалізації структури коду та забезпечення чіткої комунікації між учасниками процесу розробки.

У четвертому розділі розглянуто програмну реалізацію системи чат-застосунку. Описано структуру проєкту на фронтенді та бекенді, налаштування клієнтської та серверної частин, розробку необхідних плагінів для роботи з API та забезпечення коректного функціонування системи. Процес тестування вебсокетів та запитів забезпечив правильність роботи API та взаємодії з сервером, що підтверджує стабільність і надійність системи.

Результатом кваліфікаційної роботи є функціональний чат-застосунок, розроблений у процесі комплексного аналізу, проєктування та розробки, який забезпечує онлайн-комунікацію в реальному часі.

КРБ викладена на 62 сторінки, вона містить 4 розділи, 32 ілюстрацій, 18 таблиці, 23 джерел в переліку посилань.

Ключові слова: вебсокети, групові чати, реальний час, RESTful API.

ABSTRACT

of the Bachelor's Thesis

"Implementation of chat using WebSockets"

Student of group 409: Boretskyi Ivan

Supervisor: Candidate of Technical Sciences (Ph. D.), Associate Professor,

Davydenko Y. O.

Contemporary society is characterized by rapid development of information technologies, transforming traditional forms of communication and facilitating quick access to information through various web applications. Among the advanced technologies enabling instant communication through these applications, websockets serve as an important tool for creating chats and other forms of real-time interaction.

The relevance of developing software «Chat Implementation Using Websockets» is heightened in support of corporate communication systems, where the need for reliable and efficient information exchange is critical for managing projects and processes in a timely manner. A websocket-based chat provides instant message delivery and enables effective real-time interaction, which is crucial for modern communication services.

The object of the qualification work is real-time interaction through a web application for instant messaging based on websockets. The subject of the qualification work is the processes of developing and implementing websocket technologies for fast and secure online communication. The aim of the qualification work is to develop a highly efficient, reliable chat application based on websockets for real-time communication.

To achieve this goal, the following tasks were defined:

1) research of the subject area;
2) development of an interface for instant messaging;
3) designing the web application;
4) implementing the web application;
5) encryption implementation for data security;
6) testing of the developed web application.

Chapter 1 provides a detailed analysis of the chat application domain. User requirements were studied, and a comparative analysis of existing analogs on the market was conducted, identifying their strengths and weaknesses. This overview helped determine the key functions the product should contain, as well as technical aspects to focus on during development.

Chapter 2 describes the process of designing and modeling software for the web application for online user communication. This included creating architectural diagrams, modeling the chat application system, defining user and administrator roles, developing use case diagrams and interaction scenarios, and developing logical and physical database models for effective data storage and management.

Chapter 3 outlines the main steps in designing the chat application system, including creating diagrams to visualize code structure and ensuring clear communication among development process participants.

Chapter 4 discusses the software implementation of the chat application system. It describes the project structure on the frontend and backend, configuration of client and server parts, development of necessary plugins for working with the API, and ensuring correct system functioning. Testing of websockets and requests ensured the API's correct operation and interaction with the server, confirming the system's stability and reliability.

The result of the qualification work is a functional chat application developed through comprehensive analysis, design, and development processes, providing online communication in real-time.

The thesis is presented on 62_pages, including _4_chapters, 32_illustrations, 18_tables, and 23_sources in the reference list.

Keywords: websockets, group chats, real-time, RESTful API.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ЗВО – Заклад вищої освіти

ПЗ – Програмне забезпечення

БД – База Даних

СКБД – Система Керування Базами Даних

UML – Unified Modeling Language

API – Application Programming Interface

ORM – Object-Relational Mapping

JS – JavaScript

ВСТУП

Сучасне суспільство відзначається стрімким розвитком інформаційних технологій, які трансформують традиційні форми комунікації та сприяють швидкому доступу до інформації через різноманітні вебзастосунки. Серед передових технологій, що забезпечують миттєве спілкування через ці застосунки, вебсокети виступають як важливий інструмент для створення чатів та інших форм взаємодії в реальному часі.

Основна перевага вебзастосунків полягає у їх універсальності та доступності: користувачам не потрібно встановлювати додаткове програмне забезпечення. Вони можуть виконуватися у будь-якому сучасному браузері, незалежно від операційної системи – чи то iOS, Android, MacOS, Windows, чи Linux. Це забезпечує користувачам високу ступінь гнучкості та миттєвий доступ до застосунків з будь-якого пристрою.

Крім того, вебзастосунки здатні пропонувати постійні оновлення та покращення без необхідності користувачам перевстановлювати застосунок. Це означає, що вони завжди можуть користуватися найновішою версією застосунку. Завдяки централізованому оновленню, вебзастосунки також забезпечують кращу безпеку, оскільки всі користувачі мають однаковий рівень захисту і останні патчі безпеки.

Така гнучкість та доступність особливо актуалізується завдяки ролі, яку відіграє JS у розробці цих застосунків. JS є основою для більшості сучасних вебзастосунків, включаючи ті, що використовують вебсокети для реалізації двостороннього зв'язку в режимі реального часу. Використання JS дозволяє розробникам створювати взаємодії, які здаються безперервними та інтуїтивно зрозумілими, подібно до реального спілкування.

Таким чином, Вебсокети відіграють ключову роль у сучасних вебтехнологіях, дозволяючи створювати складні та відповідальні вебзастосунки, зокрема в електронній комерції, ігровій індустрії та соціальних медіа. Вони

забезпечують високу швидкість взаємодії, підвищену безпеку та стабільність в обміні даними, що критично важливо для захисту користувацької інформації.

Технологія вебсокетів також сприяє розвитку дистанційної освіти та телекомунікацій. Навчальні платформи та віртуальні класні кімнати, використовуючи цю технологію, можуть проводити інтерактивні заняття з різноманітними медіаресурсами без затримок, що підвищує ефективність освітнього процесу.

Актуальність розробки програмного забезпечення «Реалізація чату за допомогою вебсокетів» посилюється у підтримці корпоративних комунікаційних систем, де потреба в надійному та ефективному обміні інформацією є критичною для оперативного управління проєктами і процесами. Чат на базі вебсокетів забезпечує миттєве доставлення повідомлень та можливість ефективної взаємодії в режимі реального часу, що є ключовим для сучасних комунікаційних сервісів.

Об’єктом кваліфікаційної роботи є процес взаємодії в реальному часі через вебзастосунок для миттєвого обміну повідомленнями на базі вебсокетів.

Предметом кваліфікаційної роботи є засоби розробки і впровадження вебсокет-технологій для швидкісної та безпечної онлайн-комунікації.

Метою кваліфікаційної роботи є розробка високоефективного, надійного чат-застосунку на базі вебсокетів для комунікації в реальному часі. Для досягнення цієї мети визначено наступні завдання:

1) дослідження предметної галузі;
2) розробка інтерфейсу для миттєвого обміну повідомленнями;
3) проєктування вебзастосунку;
4) реалізація вебзастосунку;
5) впровадження шифрування для забезпечення безпеки даних;
6) тестування розробленого вебзастосунку.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Головна мета чат-застосунку полягає у створенні високоефективного та надійного рішення для комунікації в реальному часі. Розділ присвячений глибокому аналізу предметної області, що дозволяє визначити основні вимоги та очікування користувачів від подібних систем, з метою виявлення їхніх сильних та слабких сторін. Цей аналіз допомагає визначити ключові функції, які має містити продукт, а також технічні аспекти, на які звертається увага при розробці.

1.1 Структурні особливості

Трирівнева архітектура є одним з найбільш поширених підходів до розробки програмного забезпечення, і вона ідеально підходить для написання систем обміну повідомленнями, таких як чат-застосунки. Ця архітектурна модель дозволяє логічно розділити систему на три рівні, кожен з яких відповідає за свої функціональність та взаємодію з іншими рівнями.

Клієнтська частина – це та частина системи, що забезпечує інтерфейс користувача та взаємодію з ним. Вона представлена вебсторінками або мобільним застосунком, через які користувачі взаємодіють з системою.

Серверна частина – це середовище, яке відповідає за обробку запитів користувачів, управління даними та взаємодію з базою даних. Вона забезпечує обробку бізнес-логіки, а також розподілення повідомлень між користувачами.

База даних – це централізоване сховище, де зберігаються дані про користувачів, чати та повідомлення. Вона використовується серверною частиною для зберігання та отримання інформації.

Ці три рівні взаємодіють між собою, утворюючи структуру, яка дозволяє ефективно та надійно забезпечувати функціональність та продуктивність чат-застосунків.

1.2 Аналіз аналогів

У даному аналізі предметної області були взяті для порівняння три популярні месенджери: Telegram, Signal та Facebook Messenger. Кожен з цих застосунків має свої унікальні особливості та переваги, які варто врахувати при розробці власного чат-застосунку.

Згідно з [1], Телеграм – це хмарний месенджер, який працює на декількох платформах і доступний користувачам по всьому світу. Він відомий також своєю можливістю шифрування чатів «кінець-до-кінця» та відеодзвінків. Детальніший опис можна знайти у таблиці 1.1.

Таблиця 1.1 – Опис Telegram

Назва	**Telegram**
Розробник	Telegram Messenger LLP
Архітектура	Client-server
Мова реалізації	Java (Android), Swift (iOS), JavaScript (десктопна версія)
Функції	1) групові чати; 2) Telegram Bot API для створення ботів; 3) можливість відправки фотографій, відео та аудіофайлів; 4) функції конфіденційності, такі як самознищувані повідомлення.
Переваги	1) наявність версій для різних операційних систем; 2) різноманітні можливості комунікації (групові чати, відеодзвінки); 3) Telegram Wallet для зберігання і обробки криптовалюти; 4) можливість створювати ботів для автоматизації завдань.
Недоліки	1) шифрування «кінець-до-кінця» доступне лише у секретних чатах; 2) протокол шифрування MTProto викликає дискусії щодо надійності; 3) централізована інфраструктура може створювати вразливості безпеки; 4) проблеми із синхронізацією повідомлень між різними пристроями.
Вебсайт	https://telegram.org/

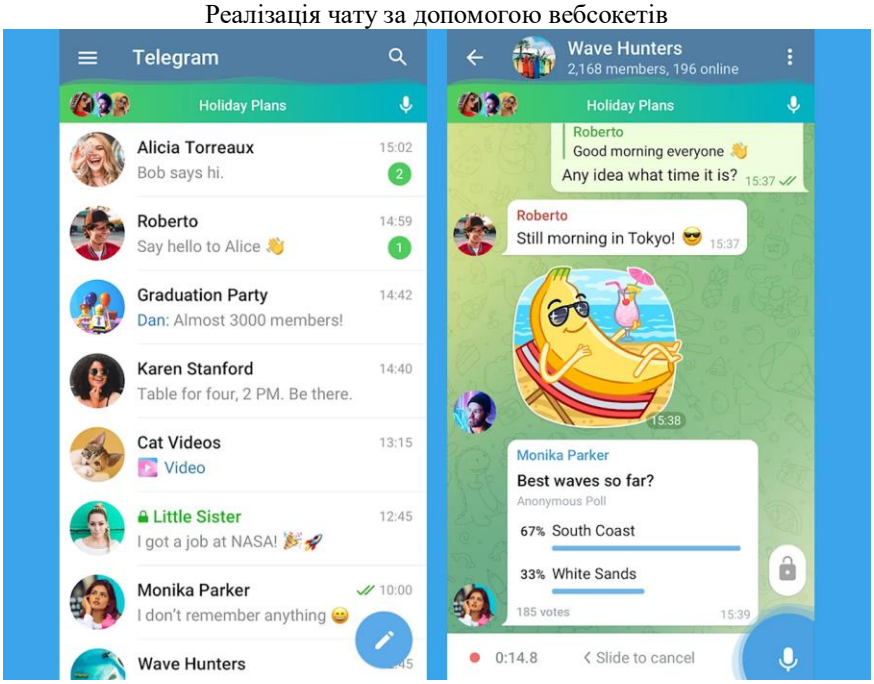

Рисунок 1.1 – Інтерфейс Telegram

Інтерфейс Телеграму (див. рисунок 1.1) вирізняється мінімалістичним дизайном, що забезпечує простоту та зручність використання. Його чистий і інтуїтивно зрозумілий макет дозволяє користувачам легко орієнтуватися між чатами, налаштуваннями та функціями застосунку, забезпечуючи приємний користувацький досвід.

Опираючись на [2], Signal є безкоштовним месенджером, доступним для обміну повідомленнями на мобільних та комп'ютерних платформах. За останній час, застосунок набуває все більшої популярності, особливо завдяки його фокусу на конфіденційності спілкування. Детальніший опис можна знайти у таблиці 1.2.

Таблиця 1.2 – Опис Signal

Назва	**Signal**
Розробник	Signal Foundation
Архітектура	Client-server
Мова реалізації	Java (Android), Swift (iOS), JavaScript (десктопна версія)
Функції	1) наскрізне шифрування повідомлень та дзвінків; 2) групові чати; 3) секретні чати з самознищенням повідомлень; 4) приватність даних без збору та зберігання інформації користувачів.

Кінець таблиці 1.2

Функції	1) наскрізне шифрування повідомлень та дзвінків; 2) групові чати; 3) секретні чати з самознищенням повідомлень; 4) приватність даних без збору та зберігання інформації користувачів.
Переваги	1) наскрізне шифрування для повної конфіденційності; 2) швидка доставка повідомлень навіть у повільних мережах; 3) відсутність реклами; 4) встановлення на до п'яти пристроях одночасно.
Недоліки	1) втрата даних під час зміни пристрою або номера телефону; 2) обмеження на розмір файлів до 100 МБ; 3) обмежена можливість адаптації інтерфейсу та налаштувань.
Веб Сайт	https://signal.org/

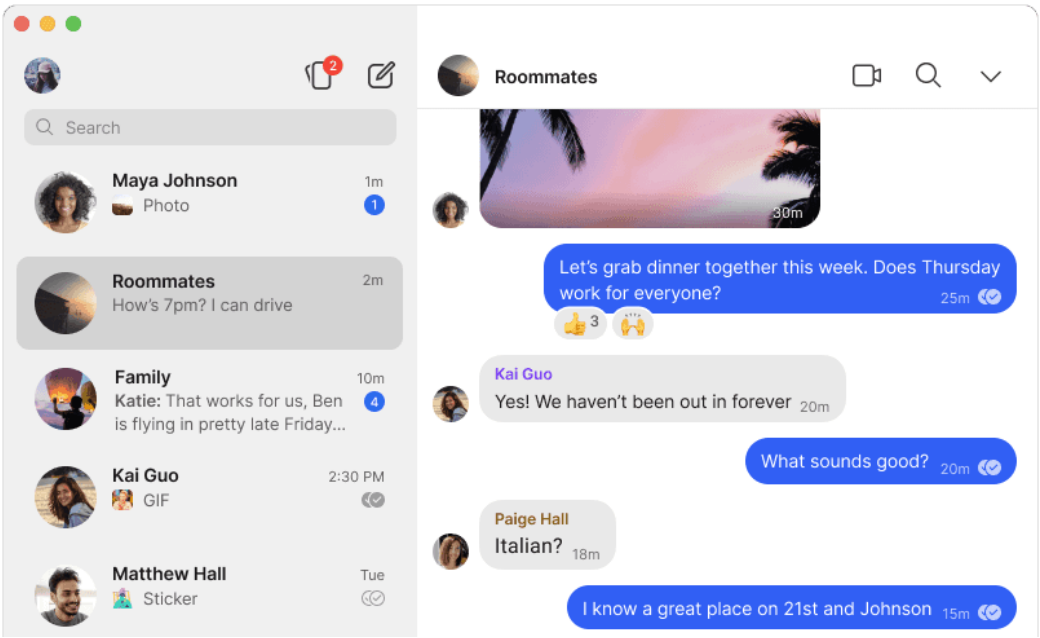

Рисунок 1.2 – Інтерфейс Signal

Інтерфейс Signal (див. рисунок 1.2) має стриманий та функціональний дизайн, зосереджений на зручності користувача. Він використовує чисті лінії та просту кольорову гамму, що сприяє легкій навігації. Інтерфейс забезпечує чіткий і невимушений досвід спілкування, підкреслюючи важливість конфіденційності та безпеки.

Базуючись на [3], Facebook Messenger є безкоштовним мобільним застосунком для миттєвого обміну повідомленнями, обміну фотографіями, відео, аудіозаписами та для групових чатів. Застосунок, який можна завантажити безкоштовно, може бути використаний для спілкування з друзями на Facebook та з контактами на вашому телефоні. Більше деталей про це застосунок можна знайти у таблиці 1.3.

Таблиця 1.3 – Опис Facebook Messenger

Назва	**Facebook Messenger**
Розробник	Meta Platforms (раніше Facebook, Inc.)
Архітектура	Client-server
Мова реалізації	Java (Android), Swift (iOS), JavaScript (десктоп)
Функції	1) групові чати з декількома учасниками; 2) швидке надсилання та отримання повідомлень через систему Facebook; 3) відправлення грошей та здійснення платежів за товари та послуги.
Переваги	1) інтеграція з Facebook та Instagram для зручного зв'язку; 2) обмін мультимедійним контентом (фотографії, відео, файли).
Недоліки	1) відображення цільових рекламних оголошень; 2) постійна робота у фоновому режимі, що споживає заряд батареї; 3) переповнений інтерфейс через велику кількість інформації та функцій.
Вебсайт	https://www.messenger.com/

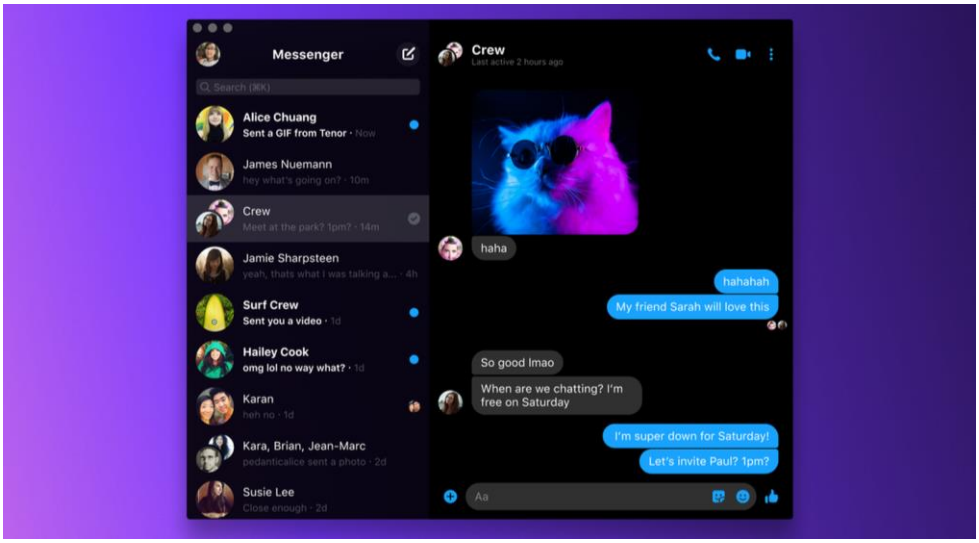

Рисунок 1.3 – Facebook Messenger

Інтерфейс Facebook Messenger (див. рисунок 1.3) характеризується яскравим та сучасним дизайном, з акцентом на легку доступність до всіх функцій. Він має інтеграцію з іншими продуктами Meta, що дозволяє користувачам легко перемикатися між платформами. Проте велика кількість інформації та функцій може створювати відчуття перевантаженості, що впливає на зручність використання.

1.3 Специфікація вимог до програмного забезпечення

ПРИЗНАЧЕННЯ ТА МЕЖІ ПРОЄКТУ

Призначення системи (застосунку), для якої розробляється програмне забезпечення

Призначенням системи є автоматизація процесу комунікації в реальному часі шляхом розробки програмного забезпечення вебзастосунку для чатів.

Погодження, що ухвалені в програмній документації

Було погоджено, що для створення програмного забезпечення та його злагодженої роботи будуть використовуватися фреймворки Nuxt.js 3 та Nest.js.

Межі проєкту ПЗ

Крайня дата завершення роботи над ПЗ – 13.06.2024 р.

ЗАГАЛЬНИЙ ОПИС

Сфера застосування

Застосунко призначений для використання як інструмент для комунікації між користувачами в реальному часі без обмежень у сферах застосування.

Характеристики користувачів

Користувачами застосунку будуть звичайні користувачі та адміністратори групових чатів. Вони повинні мати пристрій з підтримкою сучасних браузерів та доступом до мережі Інтернет.

Загальна структура і склад системи

Система складатиметься з API, бази даних та вебзастосунку, який включає фронтенд та бекенд частини.

Загальні обмеження

Система повинна забезпечувати високу продуктивність, безпеку даних, зручність користування та бути доступною для широкого кола користувачів.

ФУНКЦІЇ СИСТЕМИ

Функція реєстрації та аутентифікації

Опис функції: Користувачі повинні мати можливість реєструватись і аутентифікуватись через вебзастосунок.

Вхідна і вихідна інформація: Введення користувачем даних для реєстрації та отримання доступу до системи.

Функціональні вимоги: Забезпечити безпечну реєстрацію та аутентифікацію.

Функція пошуку та перегляду користувачів

Опис функції: Система повинна мати можливість пошуку та перегляду зареєстрованих користувачів.

Вхідна і вихідна інформація: Запит користувача та результати пошуку.

Функціональні вимоги: Забезпечити швидкий і точний пошук користувачів.

Функція створення групових чатів

Опис функції: Користувачі можуть створювати групові чати для обміну повідомленнями.

Вхідна і вихідна інформація: Інформація про створення групи та повідомлення в групі.

Функціональні вимоги: Забезпечити легкість створення та управління груповими чатами.

Функція редагування та перегляду особистих даних

Опис функції: Користувачі можуть редагувати та переглядати свої особисті дані.

Вхідна і вихідна інформація: Введення змін користувачем та оновлені дані.

Функціональні вимоги: Забезпечити безпеку та зручність редагування особистих даних.

Система сповіщень

Опис функції: Система повинна інформувати користувачів про нові повідомлення.

Вхідна і вихідна інформація: Сповіщення про нові повідомлення.

Функціональні вимоги: Забезпечити своєчасні та ненав'язливі сповіщення.

Налаштування приватності

Опис функції: Користувачі можуть налаштовувати рівень приватності.

Вхідна і вихідна інформація: Налаштування приватності користувача.

Функціональні вимоги: Забезпечити гнучкість у налаштуваннях приватності.

ВИМОГИ ДО ІНФОРМАЦІЙНОГО ЗАБЕЗПЕЧЕННЯ

Джерела і зміст вхідної інформації (даних)

Система буде спиратися на інформацію, введену користувачами, включаючи реєстраційну інформацію, повідомлення та налаштування.

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Архітектура програмної системи

Система повинна мати масштабовану та ефективну програмну архітектуру з чітким розподілом між інтерфейсом користувача, базою даних та внутрішніми системами.

Програмне забезпечення для ведення інформаційної бази

Система повинна мати програмне забезпечення для управління та ведення бази даних користувачів та повідомлень.

Мова та технологія розробки ПЗ

Програмне забезпечення для чат-застосунку буде розроблено з використанням сучасного та надійного програмного стеку. Фронтенд-застосунок буде побудований з використанням Nuxt.js 3, Tailwind, Socket.IO-Client, та Shadcn-Vue. Бекенд-застосунок буде побудований з використанням Nest.js для забезпечення реального часу та TypeORM для взаємодії з базою даних.

ВИМОГИ ДО ЗОВНІШНІХ ІНТЕРФЕЙСІВ

Інтерфейс користувача

Інтерфейс користувача для чат-застосунку повинен бути інтуїтивно зрозумілим, зручним та швидким у користуванні.

Апаратний інтерфейс

Чат-застосунок буде розроблений для роботи на стандартному апаратному забезпеченні та операційних системах. Користувачі не мають жодних специфічних вимог до апаратного забезпечення, оскільки система буде доступна через веббраузер на будь-якому пристрої з доступом до Інтернету.

Програмний інтерфейс

Програмний інтерфейс системи буде заснований на RESTful API для взаємодії між фронтендом та бекендом. Цей API дозволить користувачам взаємодіяти з системою, обмінюючись даними та запитами.

Протокол зв'язку

Чат-застосунок буде використовувати протоколи HTTP (Hypertext Transfer Protocol) та HTTPS (HTTP Secure) для забезпечення безпечного зв'язку між клієнтом та сервером. Протокол HTTPS буде використовуватися для шифрування всіх даних, що передаються через Інтернет, гарантуючи захист конфіденційної інформації.

Використання WebSockets

Чат-застосунок використовуватиме Socket.IO для забезпечення функціонування WebSockets, що дозволяє реалізувати комунікацію в реальному часі між клієнтами та сервером. Це забезпечить швидке і надійне передавання повідомлень, а також дозволить підтримувати активні з'єднання для миттєвих оновлень чату.

ВЛАСТИВОСТІ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Доступність

Чат-застосунок повинен бути доступним для широкого кола користувачів, з інтуїтивно зрозумілим інтерфейсом та підтримкою різних пристроїв.

Супроводжуваність

Застосунок буде розроблений таким чином, щоб його було легко обслуговувати та оновлювати.

Переносимість

Система буде побудована з використанням відкритих стандартів і технологій, що гарантує її роботу на будь-якому обладнанні та операційній системі, які їх підтримують.

Надійність

Надійність системи забезпечується завдяки її здатності виконувати свої функції послідовно і без помилок протягом тривалого часу. Регулярне тестування та моніторинг забезпечать виявлення і виправлення проблем на ранніх етапах.

Безпека

Система повинна передбачати заходи для захисту даних користувачів. Використання протоколу HTTPS забезпечить захист даних під час передачі через Інтернет.

Висновки до розділу 1

У ході описання вебзастосунка було ретельно проаналізовано існуючі аналоги вебзастосунка. По-перше, розглядались їхні функціональні можливості, переваги та недоліки. Цей огляд допоміг виокремити ключові інструменти та методи, необхідні для вирішення поставлених завдань шляхом розробки програмного коду. Враховуючи отриманий аналіз, були розроблені стратегії для досягнення основної мети роботи.

2 МОДЕЛЮВАННЯ СИСТЕМИ ЧАТ-ЗАСТОСУНКУ

2.1 Варіанти використання (Use cases)

У цьому розділі розглянуто сценарії використання системи. Для цього створено діаграму варіантів використання та описано сценарії взаємодії з системою за допомогою Use Case діаграм, які також відомі як діаграми прецедентів. Діаграма прецедентів фіксує домовленості між користувачами системи щодо її поведінки. Вона описує, як застосунок реагує на запити різних учасників, які називаються дійовими особами, у різних умовах. У застосунку передбачено дві дійові особи: користувач та адміністратор групового чату. Кожен із них має свої ролі та функції на сайті. Інформацію про дійових осіб представлено у таблиці 2.1.

Таблиця 2.1 – Опис ролей у чат-застосунку

Роль	**Опис**
Звичайний користувач	Може відправляти та отримувати повідомлення, управляти особистими налаштуваннями, взаємодіяти з іншими користувачами.
Адміністратор групового чату	При створенні групового чату звичайний користувач стає адміністратором цього чату, отримуючи додаткові можливості управління учасниками та правилами групи.

Діаграма використання на рис. 2.1 показує дійових осіб (Адміністратор і Звичайний користувач) та функції, доступні їм у чат-застосунку.

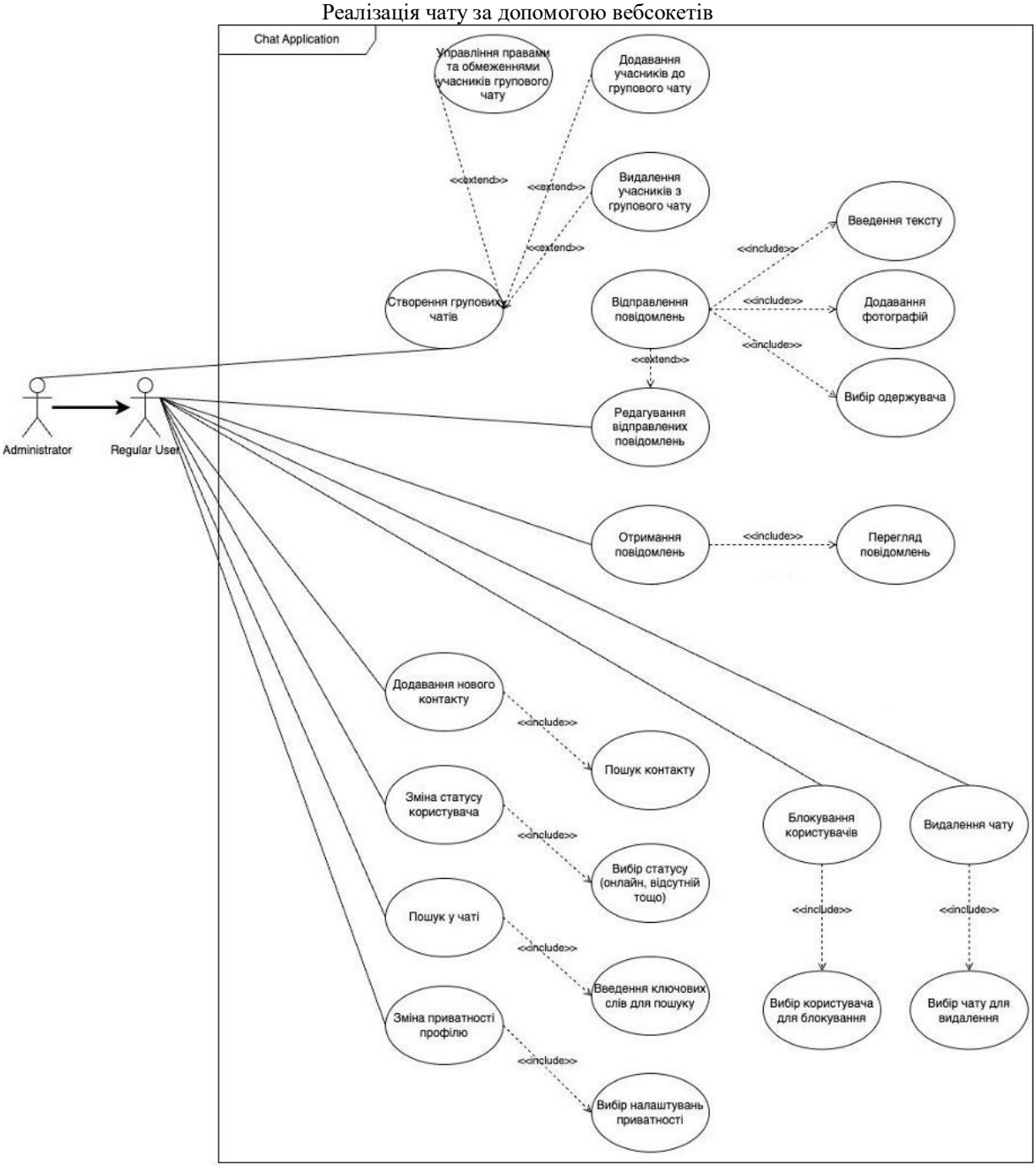

Рисунок 2.1 – Діаграма використання

Звичайний користувач може надсилати та отримувати повідомлення, додавати контакти, змінювати статус, шукати чати, редагувати приватність профілю тощо. Адміністратор додатково має можливості управління груповими чатами, такими як створення, додавання та видалення учасників, а також керування правами і обмеженнями учасників чату.

2.2 Сценарії використання

Сценарії використання – це опис способів взаємодії користувачів із системою для досягнення конкретної мети. Вони допомагають розробникам зрозуміти вимоги користувачів та забезпечити їх реалізацію.

Коротка форма – це короткий абзац або кілька речень, які описують основну мету та результати сценарію використання. Використовується для швидкого розуміння ідей та цілей користувачів на ранніх етапах розробки.

Поверхнева форма – це більш детальний, але менш структурований сценарій використання. Вона включає деякі з елементів повного сценарію, але не всі. Використовується для гнучкого підходу до опису сценаріїв, коли повна деталізація не потрібна.

Повна форма – це детальний опис сценарію використання, який включає всі необхідні елементи та кроки для реалізації. Використовується на етапі детальної розробки та проєктування системи, щоб забезпечити правильну реалізацію функціональності.

Сценарії використання допомагають усім учасникам проекту зрозуміти вимоги та очікування, забезпечують ефективну комунікацію між технічними та нетехнічними учасниками, служать важливими документами для фіксації вимог та створення тестових сценаріїв.

Сценарій №1 (коротка форма): Відправлення текстового повідомлення

Користувач відкриває діалог (переписку) з іншим користувачем або груповий чат. Вводить текст у поле для повідомлень та натискає кнопку «Відправити». Система здійснює відправлення повідомлення через сокет-з'єднання. Отримувач бачить нове повідомлення в чаті.

Сценарій №2 (поверхнева форма): Управління статусом онлайн

Головний сценарій:

1) користувач активує застосунок;
2) система автоматично встановлює статус користувача як «Онлайн»;

3) користувач виходить з застосунку або закриває його;

4) система встановлює статус користувача як «Офлайн».

Альтернативні сценарії:

– з'єднання з сервером перервано; користувач отримує повідомлення про помилку;

– користувач вручну встановлює статус через налаштування;

– система автоматично встановлює статус «Офлайн» / Онлайн» при відсутності активності;

– система не може оновити статус через внутрішню помилку.

Таблиця 2.2 – Сценарій №3 (повний): Створення групового чату

Title	Створення групового чату
Primary Actor	Користувач (Ініціатор)
Scope	Застосунок чату на сокетах
Level	Ціль користувача
Preconditions	Користувач авторизований у системі.
Stakeholders and Interests	*Користувач:* створити груповий чат для обміну повідомленнями з обраними учасниками. *Система:* забезпечення надійного та ефективного спілкування між користувачами.
Main Success Scenario 1) користувач обирає опцію створення групового чату; 2) користувач вводить назву групи; 3) користувач додає учасників зі списку контактів; 4) система створює груповий чат з вказаними учасниками; 5) учасники отримують сповіщення про додавання до нового групового чату; 6) користувач (ініціатор) відправляє привітальне повідомлення в чат; 7) система зберігає історію повідомлень чату;	

Продовження таблиці 2.2

8) учасники можуть відправляти та отримувати повідомлення у групі; 9) користувач може змінювати налаштування групи (наприклад, назву чи зображення групи); 10) адміністратор групи може додавати або видаляти учасників.	
Result	Груповий чат успішно створено та готовий до використання.
Extensions	
***a**	Система не може створити груповий чат через внутрішню помилку. Система виводить відповідне повідомлення. Result: груповий чат не створено.
1a	Користувач намагається додати неіснуючого користувача. Система виводить повідомлення про помилку. Result: користувача не додано до чату.
2a	Ініціатор відміняє процес створення чату. Result: створення групового чату відмінено.
3a	Перевищено максимально дозволену кількість учасників. Система виводить відповідне повідомлення. Result: нових учасників не додано.
4a	Учасник, якого додають до групи, заблокував ініціатора. Система виводить повідомлення про неможливість додавання учасника. Result: учасника не додано до групи.
Special Requirements	Адаптивність інтерфейсу для різних пристроїв.
Technology and Data Variations List	Підтримка мультиплатформенності.
Frequency of Occurrence	Часто, залежно від потреб користувачів.
Miscellaneous	Можливість відновлення повідомлень, якщо це передбачено налаштуваннями системи.

Таблиця 2.3 – Сценарій №4: Редагування та перегляд особистих даних користувача

Actors	Авторизований користувач
Objective	Редагувати та переглядати особисті дані користувача
Main Success Scenario: 1) користувач увійшов у меню профілю; 2) користувач вибирає опцію «Перегляд профілю»; 3) система відображає особисті дані користувача; 4) користувач натискає кнопку «Редагувати профіль»; 5) користувач змінює ім’я або іншу інформацію у відповідних полях; 6) користувач натискає кнопку «Зберегти зміни»; 7) система обробляє запит та оновлює дані користувача.	
Result	Збережено та оновлено особисті дані користувача.

Таблиця 2.4 – Сценарій №5: Пошук та перегляд зареєстрованих користувачів через сайдбар

Actors	Авторизований користувач
Objective	Пошук та перегляд зареєстрованих користувачів через сайдбар
Main Success Scenario: 1) користувач входить до системи та бачить сайдбар на головній сторінці; 2) користувач вводить ім’я або інші дані в пошукове поле сайдбару; 3) користувач натискає кнопку «Пошук»; 4) система обробляє запит і відображає список користувачів	

Кінець таблиці 2.3

5) система обробляє запит і відображає список зареєстрованих користувачів, які відповідають критеріям пошуку; 6) користувач переглядає список та вибирає потрібного користувача; 7) система відображає профіль вибраного користувача, де користувач може переглянути додаткову інформацію.	
Result	Користувач успішно знаходить та переглядає профілі інших зареєстрованих користувачів.

2.3 Діаграми взаємодії (Interaction diagram)

Діаграма взаємодій – це графічне представлення взаємодії між об'єктами в системі, що відображає послідовність і порядок передачі повідомлень між ними. Вона фокусується на часових аспектах взаємодії, показуючи, як об'єкти системи взаємодіють між собою. Діаграма взаємодій використовується для моделювання поведінки складних систем, полегшуючи розуміння їхньої структури та функціональності. Ключовими елементами такої діаграми є об'єкти, повідомлення, лінії життєвого циклу та активні області, які разом візуалізують обмін інформацією між об'єктами системи.

Діаграма (рис. 2.2) демонструє процес надсилання приватного повідомлення між користувачами в системі. Вона показує послідовність викликів методів і об'єкти, які їх обробляють.

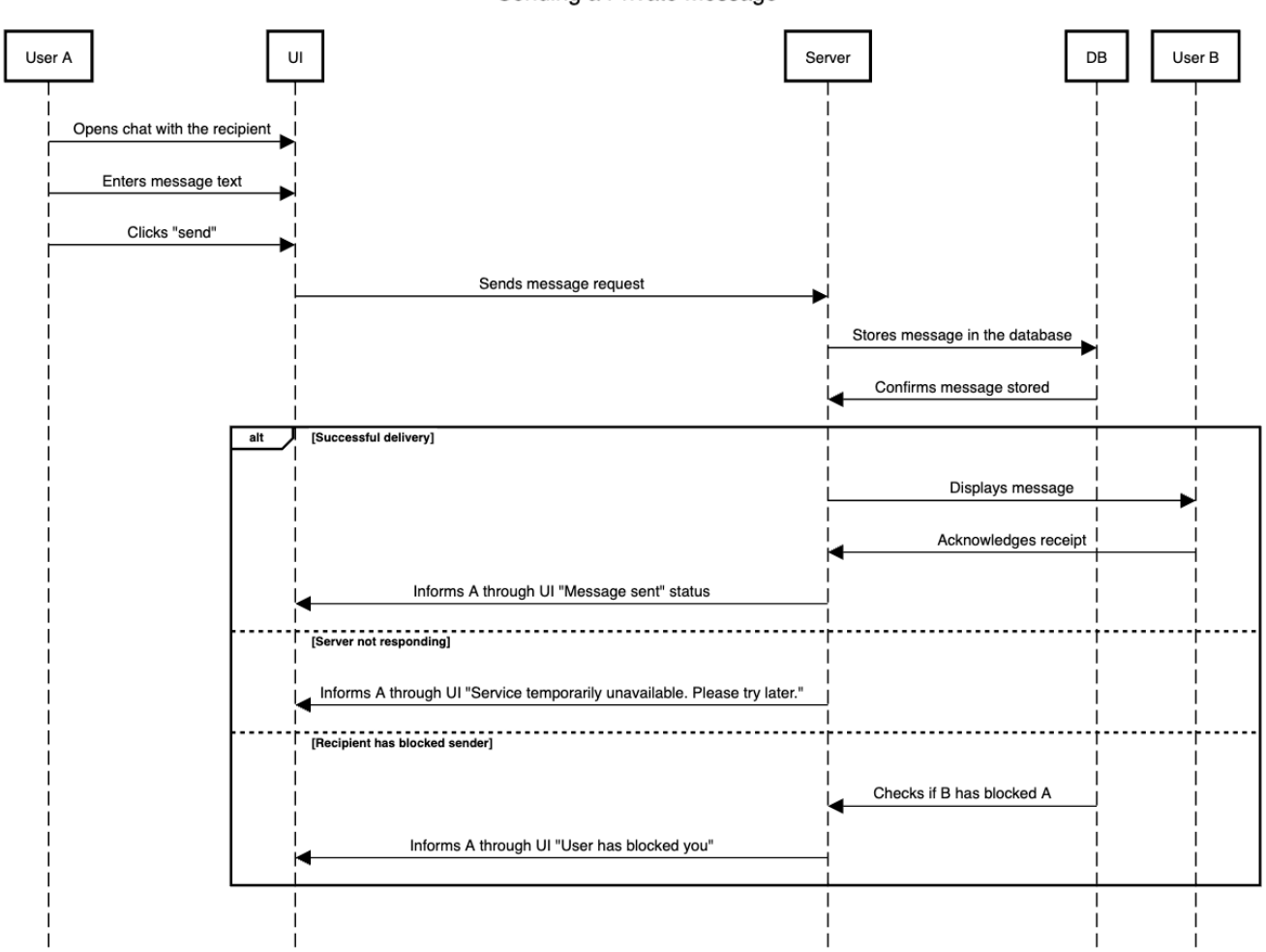

Рисунок 2.2 – Процес надсилання приватного повідомлення

Діаграма включає такі об'єкти: Користувач A (User A), Інтерфейс користувача (UI), Сервер (Server), База даних (DB) і Користувач B (User B). Користувач A відкриває чат із отримувачем через UI, вводить текст повідомлення і натискає «Надіслати». Далі UI надсилає запит на Сервер, який зберігає повідомлення в БД. БД підтверджує збереження.

Успішний сценарій: Сервер відображає повідомлення, Користувач B отримує його і підтверджує, а Сервер інформує Користувача A про успішну доставку.

Альтернативні сценарії: Якщо Сервер не відповідає, UI повідомляє Користувача A про тимчасову недоступність. Якщо Користувач B заблокував Користувача A, Сервер інформує про це через UI.

Ця діаграма допомагає візуалізувати обмін повідомленнями та можливі сценарії взаємодії.

Діаграма (рис. 2.3) демонструє процес пошуку користувачів у чат-застосунку. Вона показує послідовність викликів методів і об'єкти, які їх обробляють.

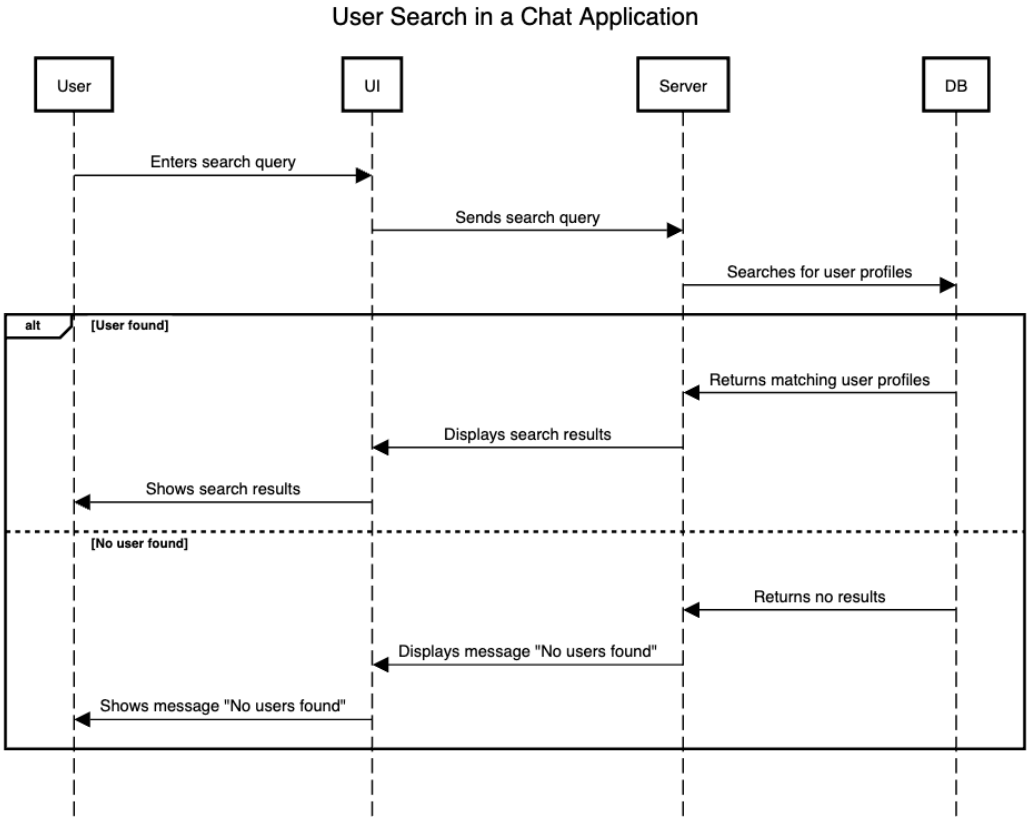

Рисунок 2.3 – Процес пошуку користувачів

Діаграма включає такі об'єкти: Користувач (User), Інтерфейс користувача (UI), Сервер (Server) і База даних (DB). Користувач вводить пошуковий запит через UI і натискає «Пошук». UI надсилає запит на Сервер, який шукає відповідні профілі користувачів у БД. БД повертає результати пошуку на Сервер.

Успішний сценарій: Сервер повертає знайдені профілі користувачів, UI відображає результати пошуку, і Користувач бачить результати.

Альтернативні сценарії: Якщо жоден користувач не знайдений, Сервер повертає відповідне повідомлення, а UI інформує Користувача, що жоден користувач не знайдений.

Ця діаграма допомагає візуалізувати процес пошуку користувачів і можливі сценарії взаємодії.

2.4 Моделювання структури бази даних вебзастосунку

У розробці програмних рішень ключове значення має проєктування бази даних, що забезпечує ефективне зберігання, доступ та управління даними. Правильне моделювання бази даних спрощує підтримку і масштабування системи. Розробка починається з визначення основних сутностей та взаємозв'язків у логічній моделі, що включає опис атрибутів та їх типів даних. На наступному етапі створюється фізична модель, яка включає створення таблиць, індексів та встановлення відносин між ними.

2.4.1 Розробка логічної моделі бази даних

Логічне моделювання бази даних є важливим етапом проєктування. Цей процес включає аналіз вимог до системи та трансформацію інформації про бізнес-процеси в моделі, що відображають зв'язки та властивості даних для застосунку. Логічна модель дозволяє візуалізувати структуру бази даних, забезпечуючи високий рівень абстракції. У рамках розробки логічної моделі визначаються сутності, їхні атрибути та відносини між ними. Такий підхід дозволяє точно спланувати базу даних для ефективної підтримки бізнес-завдань. Результатом логічного моделювання є діаграма сутностей-зв'язків (ERD), яка служить основою для подальшої розробки фізичної моделі бази даних.

На підставі ретельного аналізу вимог до системи і визначення ключових сутностей, було створено логічну модель бази даних. Ця модель детально описує структуру даних, яка необхідна для забезпечення функціональності застосунку. Основою моделі є наступні сутності:

- users – користувачі;
- messages – повідомлення;
- groups – групи;

- conversations – бесіди;
- group_messages – групові повідомлення;
- profiles – профілі;
- friends – друзі;
- group_users_users – члени груп;
- sessions – сесії;
- user_presence – присутність користувачів;
- friend_requests – запити на дружбу;
- message_attachments – вкладення в повідомленнях;
- group_message_attachments – вкладення в групових повідомленнях.

Сутність «users» зберігає інформацію про користувачів системи, включаючи особисті дані, паролі та зв'язки з їхніми профілями та статусами присутності. Це центральна сутність системи, яка забезпечує управління даними користувачів і їх аутентифікацію. Відповідні дані представлені у таблиці 2.5.

Таблиця 2.5 – Сутність users

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор користувача
username	VARCHAR(255)	ім'я користувача
email	VARCHAR(255)	електронна адреса користувача
firstName	VARCHAR(255)	ім'я користувача
lastName	VARCHAR(255)	прізвище користувача
password	VARCHAR(255)	пароль користувача
profileId	INT	ідентифікатор профілю користувача, зв'язок з profiles
password	VARCHAR(255)	пароль користувача
profileId	INT	ідентифікатор профілю користувача, зв'язок з profiles

Сутність «messages» містить дані про повідомлення, якими обмінюються користувачі, включно з вмістом, автором і відповідною бесідою. Ця таблиця відіграє ключову роль у забезпеченні функціональності обміну повідомленнями між користувачами. Дані зазначено у таблиці 2.6.

Таблиця 2.6 – Сутність messages

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор повідомлення
content	TEXT	текст повідомлення
created_at	DATETIME(6)	час створення повідомлення
authorId	INT	ідентифікатор автора повідомлення, зв'язок з users
conversationId	INT	ідентифікатор бесіди, зв'язок з conversations

Сутність «groups» описує групи створені користувачами з інформацією про назву, аватар, адміністраторів та останні повідомлення. Вона сприяє організації колективного спілкування та обміну даними.

Таблиця 2.7 – Сутність groups

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор групи
title	VARCHAR(255)	назва групи
created_at	DATETIME(6)	час створення групи
updated_at	DATETIME(6)	час оновлення інформації групи
avatar	VARCHAR(255)	аватар групи
creatorId	INT	ідентифікатор створювача групи, зв'язок з users
ownerId	INT	ідентифікатор власника групи, зв'язок з users

Інформація про бесіди між користувачами, включаючи деталі про створення, учасників та останні повідомлення, представлена у таблиці 2.8, що відображає сутність «conversations».

Таблиця 2.8 – Таблиця conversations

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор бесіди
created_at	DATETIME(6)	час створення бесіди
updated_at	DATETIME(6)	час останнього оновлення бесіди
creatorId	INT	ідентифікатор створювача бесіди, зв'язок з users
recipientId	INT	ідентифікатор одержувача, зв'язок з users
last_message_sent	INT	ідентифікатор останнього повідомлення, зв'язок з messages

Сутність «group_messages» зберігає повідомлення, що надсилаються в рамках групових чатів, з інформацією про вміст, автора та групу (таб. 2.9). Вона є важливою для реалізації групових комунікацій.

Таблиця 2.9 – Таблиця group_messages

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор повідомлення у групі
content	TEXT	текст повідомлення
created_at	DATETIME(6)	час створення повідомлення
authorId	INT	ідентифікатор автора, зв'язок з users
groupId	INT	ідентифікатор групи, зв'язок з groups

Сутність «profiles» містить деталізовану інформацію про профілі користувачів, включаючи опис, аватар та банер. Це дозволяє користувачам персоналізувати свої профілі. Більш детальні дані зазначено в таблиці 2.10.

Таблиця 2.10 – Таблиця profiles

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор профілю
about	VARCHAR(255)	інформація про користувача
avatar	VARCHAR(255)	аватар користувача
banner	VARCHAR(255)	банер профілю

Сутність «friends» відстежує дружні зв'язки між користувачами, включно з датами створення і учасниками відносин. Ця інформація сприяє управлінню дружніми зв'язками. Подробиці можна переглянути в таблиці 2.11.

Таблиця 2.11 – Таблиця friends

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор дружніх зв'язків
created_at	DATETIME(6)	час створення дружнього зв'язку
senderId	INT	ідентифікатор відправника запиту, зв'язок з users
receiverId	INT	ідентифікатор отримувача, зв'язок з users

Управління членством користувачів у групах описано через сутність «group_users_users», яка вказує, хто є учасником кожної групи. Це сприяє організації доступу до групових ресурсів. Детальніше в таблиці 2.12.

Таблиця 2.12 – Таблиця group_users_users

Поле	**Тип даних**	**Опис**
groupId	INT	ідентифікатор групи, зв'язок з groups
userId	INT	ідентифікатор користувача, зв'язок з users

Сутність «sessions» керує сесіями користувачів, зберігаючи час завершення сесії, деталі в JSON форматі та час їх знищення, що забезпечує безпеку та контроль доступу. Відомості про сесії відображено у таблиці 2.13.

Таблиця 2.13 – Таблиця sessions

Поле	**Тип даних**	**Опис**
id	VARCHAR(255)	унікальний ідентифікатор сесії
userId	INT	ідентифікатор користувача, зв'язок з users
expiredAt	BIGINT	час завершення сесії
json	TEXT	деталі сесії
destroyedAt	DATETIME(6)	час знищення сесії

Сутність «user_presence» зберігає статуси присутності користувачів, включаючи повідомлення про статус та індикацію відображення офлайн. Це допомагає в реалізації статусів присутності. Інформація наведена у таблиці 2.14.

Таблиця 2.14 – Таблиця user_presence

Поле	**Тип даних**	**Опис**
id	INT	унікальний ідентифікатор
statusMessage	VARCHAR(255)	статусне повідомлення
showOffline	TINYINT	чи показувати користувача як офлайн

У процесі створення логічної моделі за методом ER основним результатом роботи є ER-діаграма. Ця діаграма є спеціалізованою моделлю даних, яка відображає визначені елементи та їхні взаємозв'язки. Кінцева структура бази даних і розподіл стовпців у таблицях представлені на рис. 2.4.

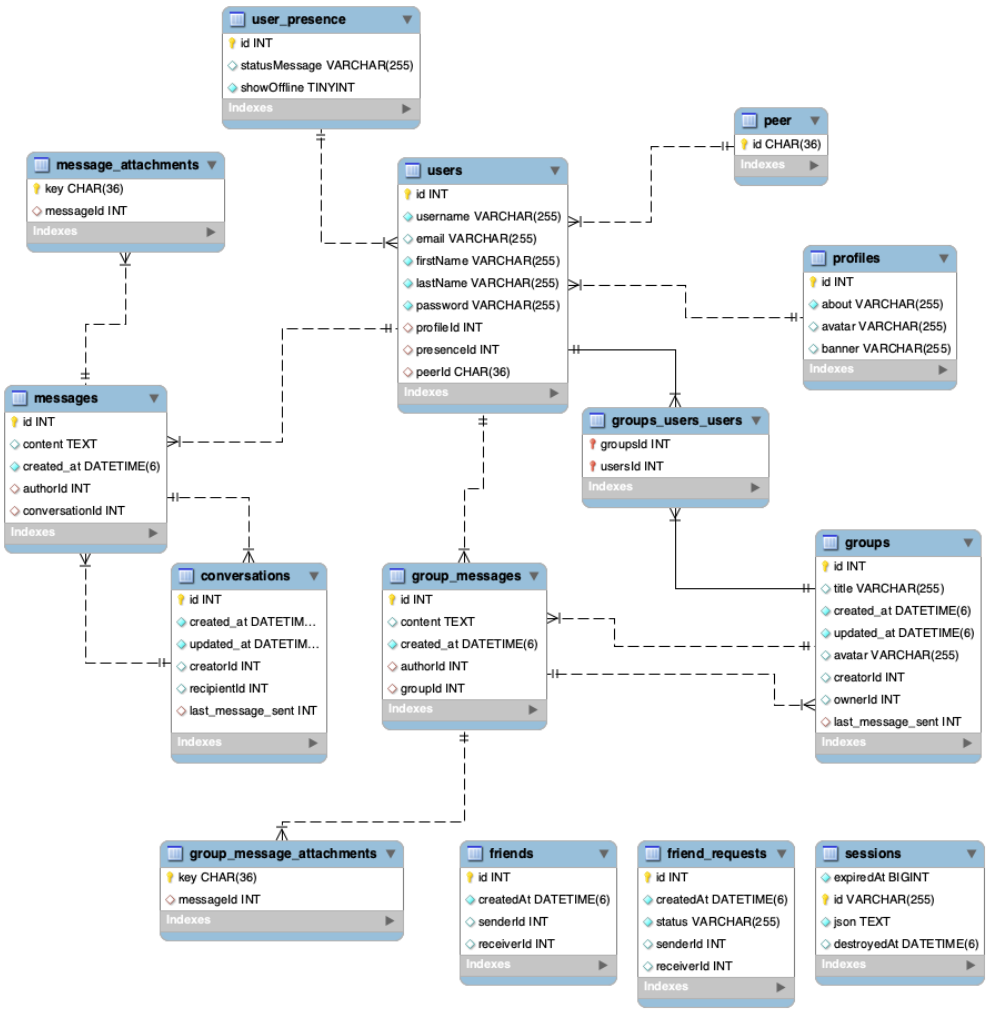

Рисунок 2.4 – ER-діаграма застосунку

2.4.2 Розробка фізичної моделі бази даних та роль СКБД

Фізичне моделювання бази даних є продовженням логічного проєктування та орієнтоване на адаптацію до конкретної СКБД. Цей процес включає розробку специфікацій для зберігання даних та управління транзакціями з урахуванням ефективності та швидкості виконання запитів, особливо для великих та складних систем.

Фізичне моделювання розпочинається з перетворення логічної моделі в структури, що ефективно реалізовані на обраній платформі СКБД. Це включає:

– визначення таблиць та індексів для поліпшення швидкості пошуку та доступу до даних;

– оптимізацію схеми даних для мінімізації часу відгуку та ефективного використання ресурсів;

– накладання обмежень цілісності для забезпечення точності та надійності інформації.

Кожна СКБД має свої особливості, що враховуються при фізичному моделюванні, такі як підтримка транзакцій та відновлення даних після збоїв. Вибір СКБД є ключовим для відповідності технічним вимогам і бізнес-потребам проєкту.

Висновки до розділу 2

У цьому розділі проведено моделювання системи чат-застосунку, визначено ролі користувачів та адміністраторів, створено діаграми варіантів використання та сценаріїв взаємодії. Це допомагає зрозуміти роботу системи та забезпечує відповідність її функціональності вимогам користувачів.

Моделювання бази даних включало розробку логічної та фізичної моделей, що забезпечують ефективне зберігання та управління даними. Проведене моделювання створює основу для проєктування та реалізації чат-застосунку, забезпечуючи його функціональність, продуктивність та масштабованість.

3 ПРОЄКТУВАННЯ СИСТЕМИ ЧАТ-ЗАСТОСУНКУ

UML-діаграми є загальноприйнятим стандартом для моделювання програмного забезпечення. Вони дозволяють аналізувати систему на різних рівнях деталізації. Діаграма класів описує структуру системи, включаючи класи, їхні атрибути, методи, спадкування та асоціації між класами. Діаграма компонентів відображає фізичну структуру системи та взаємозв’язки між її компонентами, наприклад, залежності та інтерфейси. Діаграма пакетів допомагає відобразити організацію компонентів системи на рівні пакетів, що дозволяє структурувати та керувати складністю системи.

Всі ці діаграми, як зазначено в таблиці 3.1, є ключовими для розуміння та проєктування системи. Вони полегшують комунікацію між розробниками, бізнес-аналітиками та іншими учасниками процесу розробки програмного забезпечення.

Таблиця 3.1 – Опис використання UML-діаграм

Назва	**Використання**
Діаграма класів	Відображає класи, інтерфейси, спадкування, асоціації, методи та властивості.
Діаграма компонентів	Відображає компоненти системи та зв'язки між ними.
Діаграма пакетів	Відображає пакети та залежності між ними.

На етапі проєктування перед початком розробки чат-застосунку були створені діаграми класів, компонентів та пакетів, щоб забезпечити чітке розуміння майбутньої структури коду.

3.1 Діаграма класів

Діаграма класів – це тип діаграми UML [4], який показує структуру системи, відображаючи її класи, їхні атрибути, методи та взаємозв'язки між класами. Вона допомагає розробникам зрозуміти, як різні частини системи взаємодіють одна з одною.

У контексті чат-застосунку, діаграма класів (рис. 3.1) ілюструє основні класи, такі як користувачі, повідомлення, групи, та їхні атрибути і взаємозв'язки.

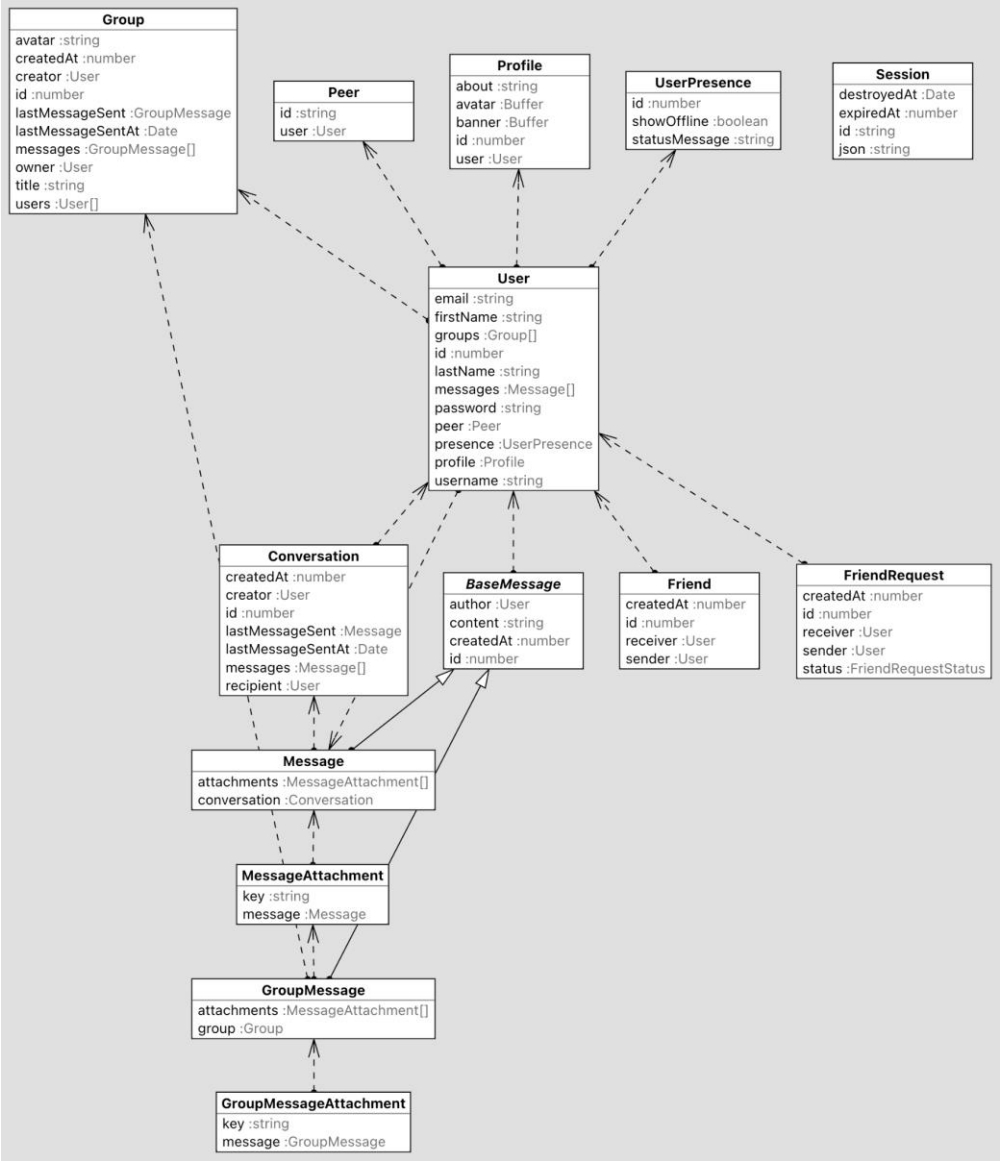

Рисунок 3.1 – Діаграма класів

Ця діаграма допомагає візуалізувати взаємозв'язки між об'єктами та полегшує розуміння складної структури системи, що важливо для її розробки та підтримки.

3.2 Діаграма компонентів

Діаграма компонентів на рис. 3.2 ілюструє архітектуру чат-застосунку, що складається з трьох основних компонентів: клієнтський застосунок, серверний застосунок і база даних.

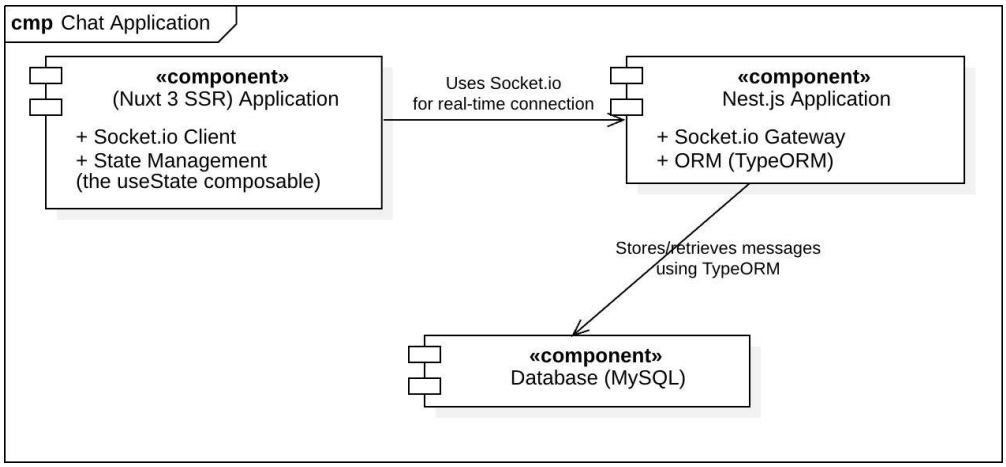

Рисунок 3.2 – Діаграма компонентів

Client (Nuxt 3 SSR) Application: Клієнтська частина застосунку побудована на Nuxt 3 з підтримкою серверного рендерингу (SSR). Вона використовує Socket.io Client для реального часу з'єднань та управління станом за допомогою composable функцій (useState).

Між клієнтом та сервером встановлюється з'єднання, що дозволяє обмінюватися повідомленнями в реальному часі.

Server (Nest.js) Application: Серверна частина застосунку побудована на Nest.js. Вона включає Gateway для обробки з'єднань Socket.io та ORM (TypeORM) для взаємодії з базою даних.

Сервер обробляє запити від клієнта та взаємодіє з базою даних для зберігання та отримання даних.

Database (MySQL): База даних, що використовує MySQL. Вона зберігає та отримує повідомлення за допомогою TypeORM.

Діаграма компонентів допомагає візуалізувати структуру застосунку та взаємодію між його основними частинами. Вона дає зрозуміти, як клієнтська частина взаємодіє з сервером за допомогою реального часу з'єднань (Socket.io). Така діаграма є важливою для розробників, оскільки вона забезпечує чітке розуміння архітектури застосунку, що полегшує його розробку, підтримку та масштабування.

3.3 Діаграма пакетів

Діаграма пакетів у контексті чат-застосунку показує організацію різних модулів і бібліотек, що використовуються як на клієнтській, так і на серверній сторонах. Вона допомагає візуалізувати залежності між пакетами та структуру застосунку.

На діаграмі пакетів фронтенду (рис. 3.3) зображено пакети, використовувані в Nuxt 3 SSR, утиліти для роботи з Vue, валідацію форм, комунікацію через Socket.io, іконки та стилізацію, а також загальні утиліти.

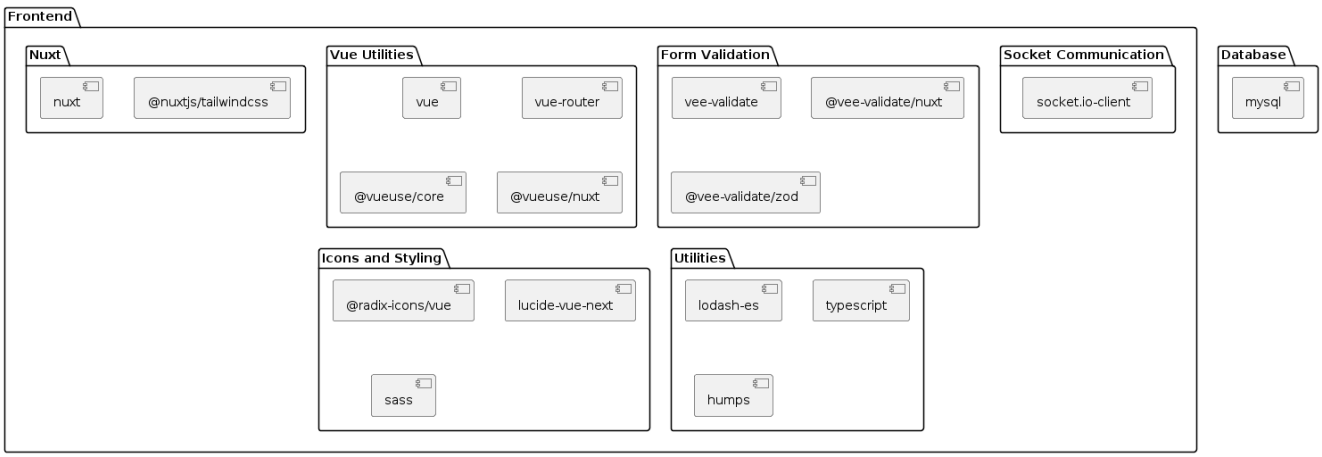

Рисунок 3.3 – Діаграма пакетів фронтенду та бази даних

На діаграмі пакетів бекенду (рис. 3.4) показані модулі Nest.js, такі як common, core, platform-express, та інші для роботи з вебсокетами, аутентифікації (passport-local, bcrypt), а також утиліти. Вона також включає пакети для роботи з базою даних через TypeORM та MySQL.

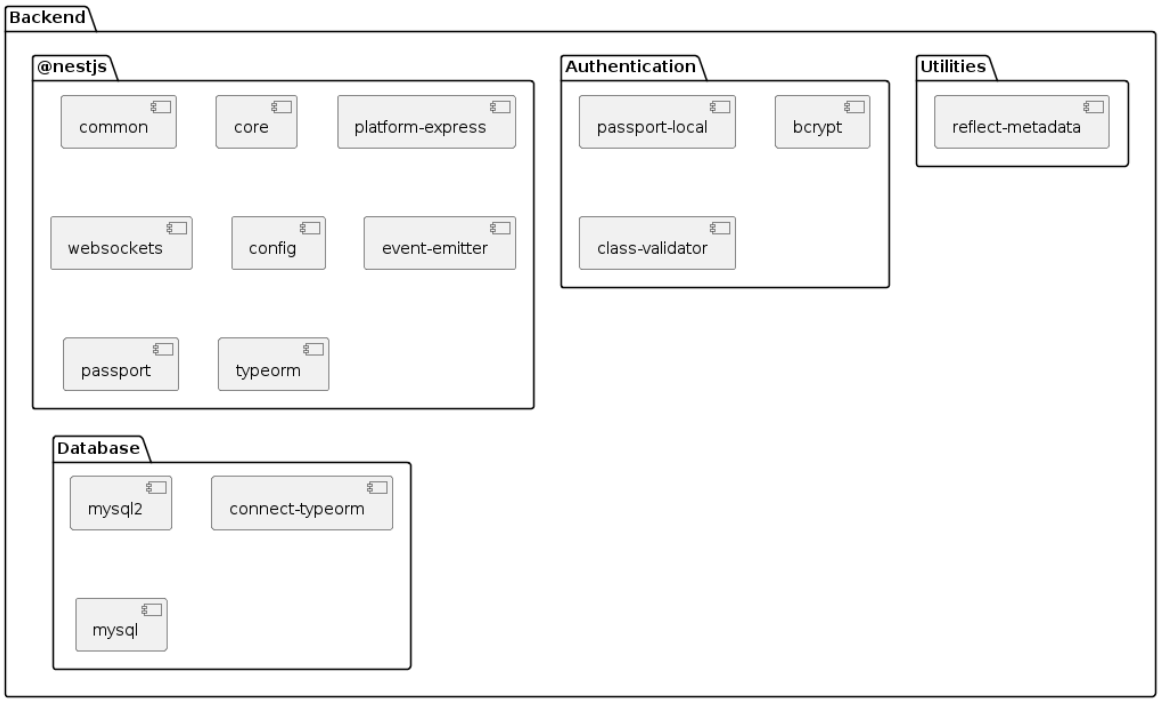

Рисунок 3.4 – Діаграма пакетів бекенду

Ці діаграми допомагають зрозуміти, як організовано різні частини чат-застосунку та їхні взаємозв'язки.

3.4 Вибір технологій розробки клієнтської частини вебзастосунку

Для розробки клієнтської частини вебзастосунку було обрано наступний технологічний стек, що забезпечує високу продуктивність, надійність та гнучкість у розробці:

Nuxt.js 3 [5] є основою застосунку. Це фреймворк для створення універсальних (SSR/SPA) застосунків на основі Vue.js, що дозволяє використовувати серверний рендеринг (SSR) для покращення продуктивності та SEO-оптимізації. SSR забезпечує швидке завантаження сторінок і кращу індексацію пошуковими системами, що є критично важливим для сучасних вебзастосунків (рис. 3.5).

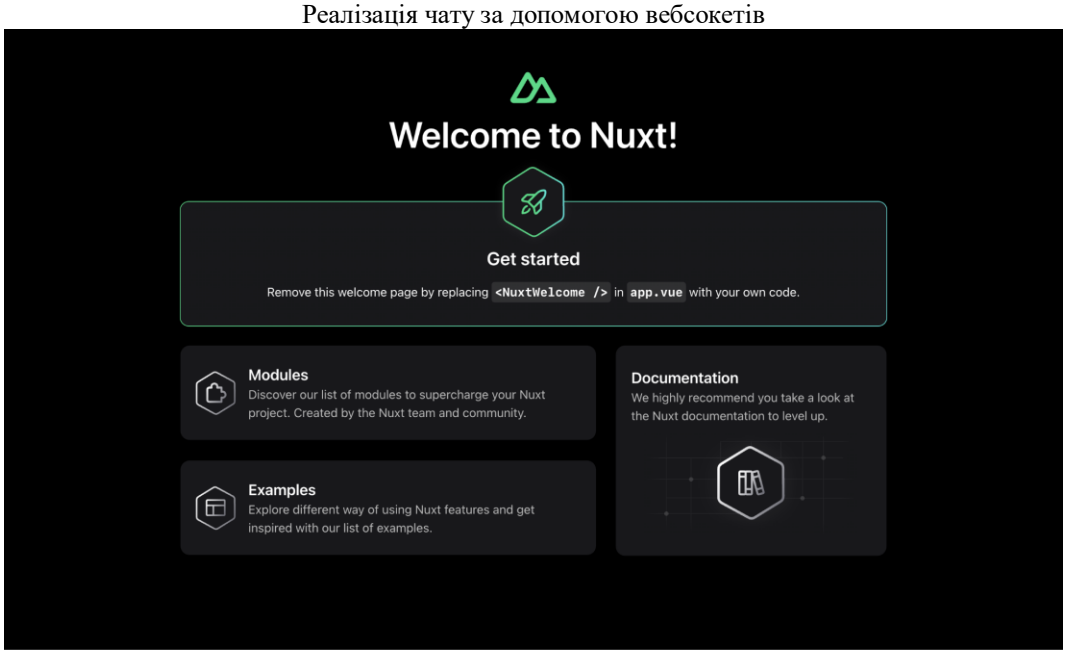

Рисунок 3.5 – Інтерфейс Nuxt.js 3

Nuxt.js 3 значно спрощує розробку складних застосунків завдяки своїй модульній архітектурі. Для забезпечення стильного та адаптивного інтерфейсу користувача обрано Tailwind CSS [6]. Це утилітарний CSS фреймворк, що дозволяє швидко та легко створювати адаптивні інтерфейси користувача. Використання Tailwind CSS забезпечує високу гнучкість та контроль над стилізацією компонентів.

Tailwind CSS дозволяє зосередитися на створенні унікального дизайну без написання великої кількості користувацьких стилів. Для підвищення якості коду і зниження кількості помилок використовується TypeScript [7]. Це супермова для JavaScript, яка додає типізацію, що допомагає знизити кількість помилок і покращує зручність розробки. TypeScript компенсує недоліки JavaScript, такі як відсутність статичної типізації, що може призводити до помилок під час виконання. Завдяки TypeScript, розробники можуть виявляти помилки ще на етапі компіляції, що робить процес розробки більш передбачуваним та надійним (рис. 3.6).

Рисунок 3.6 – Приклад коду на TypeScript

TypeScript забезпечує кращу інтеграцію з редакторами коду та інструментами розробки, що підвищує продуктивність команди. Для забезпечення миттєвої передачі даних у реальному часі між клієнтською та серверною частинами застосунку використовується Socket.IO-Client [8]. Це бібліотека (рис. 3.7) для реалізації двостороннього зв'язку в реальному часі, що є критично важливим для чат-застосунків. Socket.IO-Client дозволяє створювати постійні з'єднання між клієнтом та сервером, що забезпечує негайну доставку повідомлень і синхронізацію стану застосунку в реальному часі. Це надзвичайно важливо для функціонування чат-застосунків, де затримки у передачі повідомлень неприпустимі.

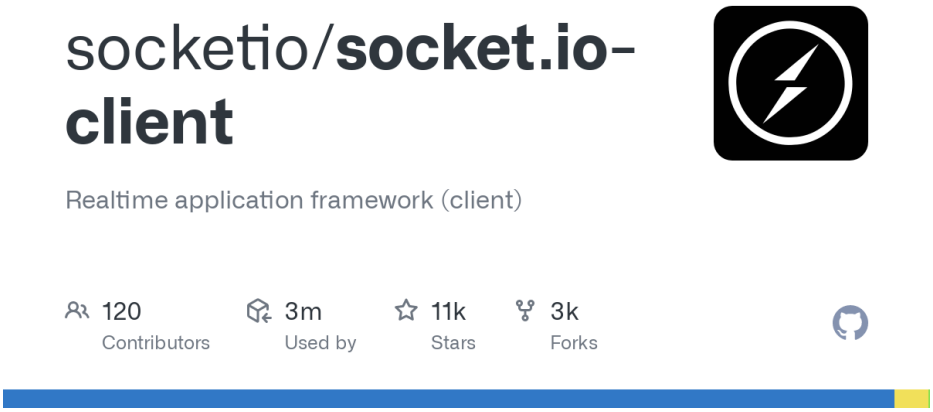

Рисунок 3.7 – Socket.IO-client

Socket.IO підтримує різні методи транспортування, що забезпечує надійність з'єднання в будь-яких умовах мережі. Для швидкої та зручної розробки інтерфейсів користувача використовується бібліотека Shadcn-Nuxt [9]. Вона надає готові компоненти, які легко налаштовуються під потреби проєкту, що дозволяє зменшити час на розробку.

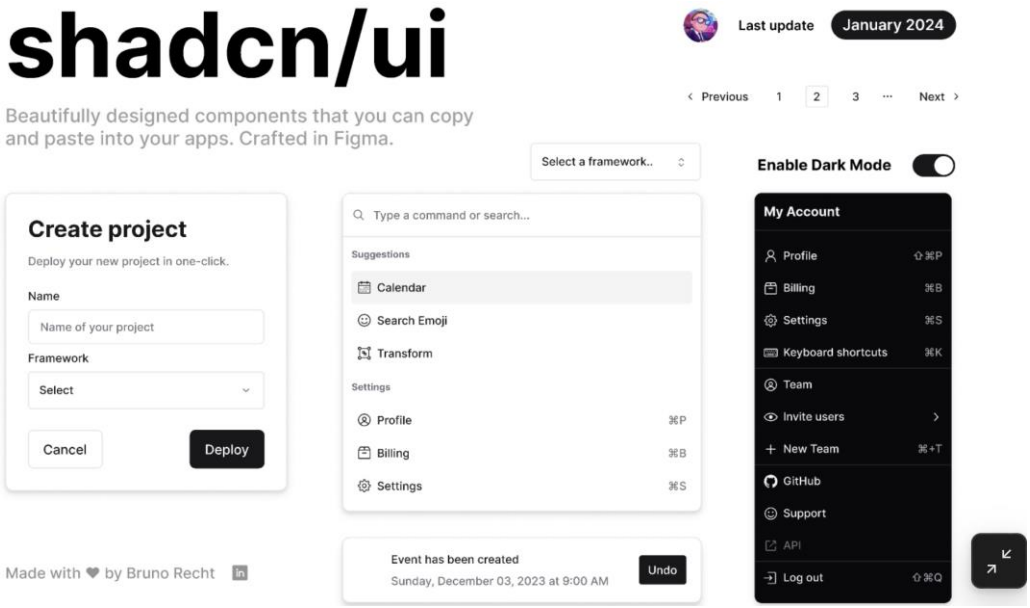

Рисунок 3.8 – Приклад компонентів Shadcn-Nuxt

Shadcn-Nuxt дозволяє швидко створювати привабливі інтерфейси (див. рисунок 3.8), мінімізуючи час на розробку користувацьких компонентів. Для валідації форм у Vue.js обрано бібліотеку Vee-Validate [10], яка дозволяє легко налаштовувати правила валідації та інтегрувати їх у компоненти. Це забезпечує високу якість та надійність введених користувачем даних.

Vee-Validate значно спрощує процес валідації форм, роблячи його інтуїтивно зрозумілим для розробників. Для стилізації застосунку також використовується Sass - препроцесор CSS, що надає можливість використовувати змінні, вкладення, міксини та інші функції, які роблять процес написання стилів більш ефективним і організованим.

Sass [11] значно підвищує продуктивність розробників, забезпечуючи організацію та повторне використання стилів. Для управління станом у вебзастосунку використовуються вбудовані можливості Nuxt.js, такі як useState. Це

дозволяє ефективно керувати станом застосунка, забезпечуючи реактивність та узгодженість даних між компонентами.

Вбудовані функції управління станом у Nuxt.js забезпечують простий і ефективний спосіб роботи зі станом застосунка.

3.5 Вибір технологій розробки серверної частини вебзастосунку

Для серверної частини вебзастосунку було обрано наступний технологічний стек, що забезпечує високу продуктивність, безпеку та масштабованість:

Основу серверного застосунку становить Nest.js [12]. Це прогресивний Node.js фреймворк (рис. 3.9) для побудови ефективних, надійних і масштабованих серверних застосунків. Nest.js використовує модульну архітектуру, що полегшує організацію коду та дозволяє легко додавати нові функціональні можливості. Завдяки TypeScript, який є основною мовою Nest.js, розробники отримують переваги статичної типізації, що підвищує якість і передбачуваність коду.

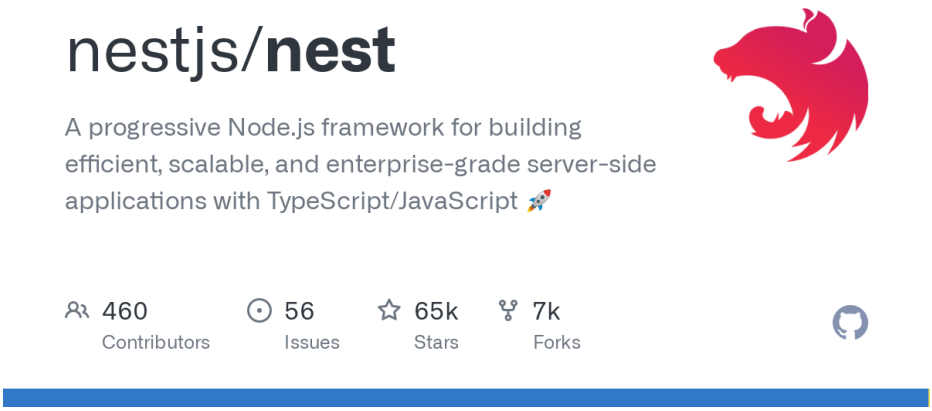

Рисунок 3.9 – Прогресивний Node.js фреймворк

Для роботи з реальними даними використовується TypeORM [13], що надає зручні інструменти для взаємодії з базою даних. Використання TypeORM (рис. 3.10) дозволяє легко створювати та керувати базами даних, а також забезпечує гнучкість у побудові складних запитів. Це важливо для чат-застосунку, де зберігання та обробка великої кількості даних є критичними.

Рисунок 3.10 – TypeORM - ORM (Object-Relational Mapping)

Для забезпечення безпеки та аутентифікації користувачів використовуються такі бібліотеки, як Passport.js і bcrypt. Passport.js - це модуль для аутентифікації [14], що підтримує різні стратегії аутентифікації, включаючи локальну, OAuth та інші. bcrypt [15] використовується для безпечного хешування паролів, що забезпечує високий рівень захисту даних користувачів.

Щоб забезпечити миттєву комунікацію в реальному часі між клієнтом і сервером, використовується бібліотека Socket.IO, інтегрована через @nestjs/platform-socket.io. Це забезпечує двосторонній зв'язок у реальному часі, що є ключовим для функціонування чат-застосунку. Socket.IO дозволяє створювати постійні з'єднання, що забезпечує швидку передачу повідомлень і синхронізацію даних.

Для управління сесіями та автентифікацією використовуються модулі express-session та connect-typeorm. Вони забезпечують надійне зберігання сесій користувачів, що підвищує безпеку та стабільність роботи застосунка. cookie-parser використовується для обробки файлів cookie, що дозволяє легко керувати сесіями та автентифікацією.

Для валідації даних та трансформації об'єктів у застосунку використовуються бібліотеки class-validator та class-transformer. Вони забезпечують високу якість даних, що вводяться користувачами, та полегшують роботу з об'єктами, роблячи код більш структурованим і читабельним.

Для завантаження та обробки зображень використовується бібліотека sharp. Вона дозволяє легко змінювати розміри зображень, конвертувати формати та виконувати інші операції з зображеннями, що є важливим для обробки аватарів користувачів у чат-застосунку.

Вибраний технологічний стек для серверної частини забезпечує високий рівень продуктивності, безпеки та гнучкості, що дозволяє створювати масштабовані та надійні вебзастосунки.

3.6 Вибір бази даних для вебзастосунку

Для чат-застосунку була обрана база даних MySQL (рис. 3.11), одна з найпопулярніших реляційних баз даних. Базуючись на [16], MySQL забезпечує високу продуктивність, надійність і масштабованість, що робить її ідеальним вибором для застосунків, які потребують обробки великих обсягів даних. Вона здатна обробляти великі обсяги транзакцій із високою швидкістю, що критично для чат-застосунку, де велика кількість повідомлень обробляється у реальному часі. MySQL підтримує складні запити та транзакції, ефективно керуючи взаємозв'язками між даними.

Рисунок 3.11 – Логотип MySQL

Використання MySQL у поєднанні з ORM бібліотекою TypeORM значно спрощує роботу з базою даних, дозволяючи легко створювати схеми баз даних, визначати зв'язки між таблицями та виконувати складні запити. Для управління базою даних та виконання складних запитів використовується DBeaver (рис. 3.12). Опираючись на [17], DBeaver – універсальний інструмент для роботи з базами

даних, який підтримує MySQL, надаючи зручний інтерфейс для виконання SQL-запитів, аналізу даних та управління схемами баз даних. Це дозволяє розробникам ефективно працювати з базою даних, налаштовувати індекси та виконувати інші адміністративні завдання.

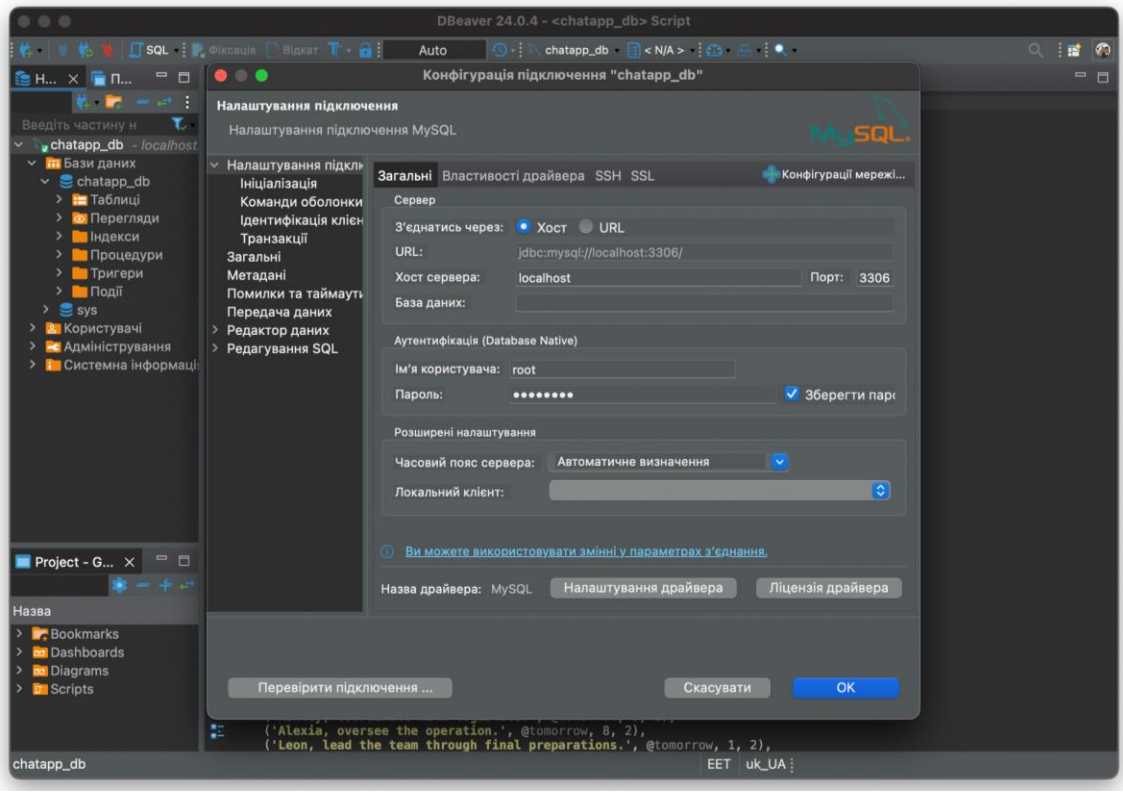

Рисунок 3.12 – Скріншот інтерфейсу DBeaver з підключеною базою даних MySQL

Для ефективного управління та підтримки бази даних, спільне використання DBeaver та MySQL забезпечує низку переваг. DBeaver надає інтуїтивно зрозумілий інтерфейс, що спрощує процес налаштування з'єднань, управління таблицями та виконання SQL-запитів. Завдяки цьому, розробники можуть зосередитися на оптимізації продуктивності бази даних та забезпеченні її надійності. Крім того, DBeaver підтримує функції резервного копіювання та відновлення баз даних, що є важливим аспектом для забезпечення безпеки даних. Це робить його незамінним інструментом для адміністраторів баз даних, які використовують MySQL у своїх проектах.

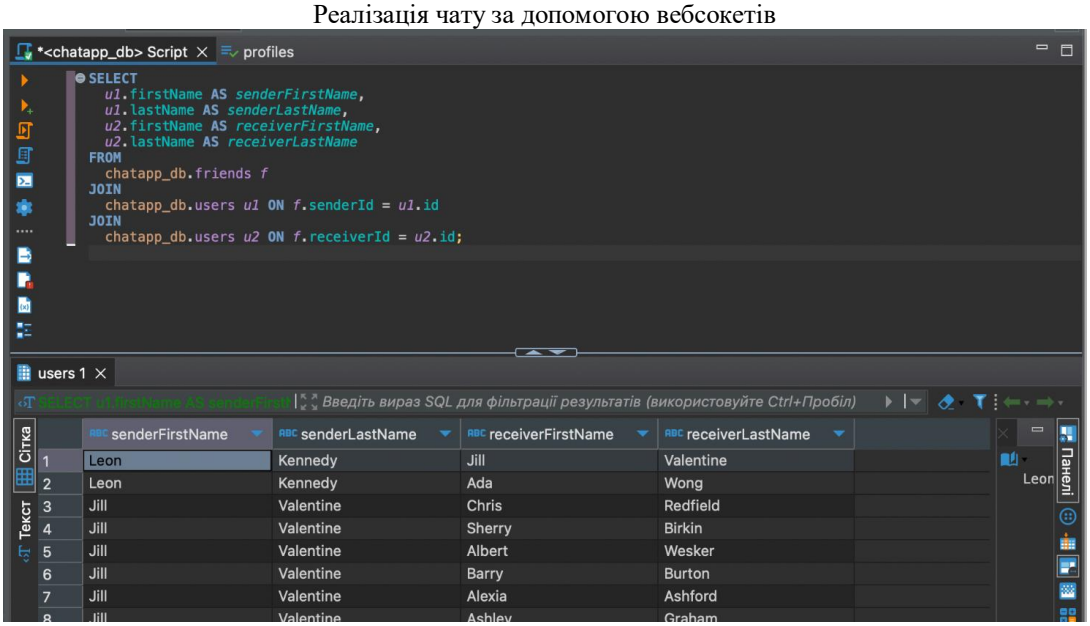

Рисунок 3.13 – Приклад використання DBeaver для виконання SQL-запиту

Вибір MySQL як основної бази даних застосунку забезпечує стабільну роботу з даними, високу продуктивність та гнучкість у розробці. Використання DBeaver додатково спрощує управління базою даних, дозволяючи розробникам швидко та ефективно виконувати необхідні операції.

Висновки до розділу 3

У третьому розділі було проведено детальне проєктування системи чат-застосунку з використанням UML-діаграм, що дозволило чітко визначити структуру та взаємодію компонентів системи. Створення діаграм класів, компонентів і пакетів допомогло організувати систему та полегшити процес її розробки та підтримки.

Завдяки ретельному проєктуванню система має чітку архітектуру, що сприяє її стабільній роботі та легкості подальшої розробки і масштабування.

4 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ ЧАТ-ЗАСТОСУНКУ

4.1 Опис frontend структури проєкту

Так як проєкт побудований на базі Nuxt 3, це забезпечує зручну структуру для розробки фронтенд-застосунків з підтримкою серверного рендерингу (SSR).

Маршрутизація:

Проєкт містить 11 основних маршрутів, що відповідають за різні сторінки та функціональності:

- / - головна сторінка.
- /auth - сторінка аутентифікації.
- /edit-profile - сторінка редагування профілю.
- /g/:id? і /g/:id?/info - маршрути для групових чатів.
- /settings, /settings/general, /settings/language - сторінки налаштувань.
- /t/:id? - маршрут для приватних чатів.
- /u/:id? - маршрут для профілю користувача.

Ці маршрути забезпечують організацію навігації в застосунку, дозволяючи користувачам легко переміщатися між різними сторінками.

Компоненти:

Проєкт містить 123 компоненти, організовані в різні категорії:

- UI-компоненти: accordion, avatar, badge, button, card, collapsible, context-menu, dialog, drawer, dropdown-menu, form, input, label, scroll-area, sheet, skeleton, switch, tabs, textarea, tooltip. Ці компоненти відповідають за різні елементи інтерфейсу користувача.
- Специфічні компоненти: InfiniteLine.vue, LoginForm.global.vue, NewUserDialog.vue, PhoneIcon.vue, RegisterForm.global.vue, Sidebar.vue, ThreadList.vue, ThreadMessages.vue.

Проєкт має два основні лейаути:

- auth.vue – лейаут для сторінок аутентифікації.
- default.vue – основний лейаут для решти сторінок.

Показники продуктивності:

Проєкт демонструє гарні показники продуктивності:

- SSR to full load: 154 мс.
- Page load: 68 мс.
- Navigation: 47 мс.

Ці показники вказують на ефективність роботи застосунку, забезпечуючи швидкий рендеринг і навігацію.

Власний фетч для роботи з API

Для роботи з API чат-застосунку було створено власний розширений фетч (рис. 4.1) на основі ofetch та $fetch у Nuxt 3 [18]. Цей фетч забезпечує зручний спосіб виконання HTTP-запитів з використанням конфігурації, що зчитується з налаштувань проєкту.

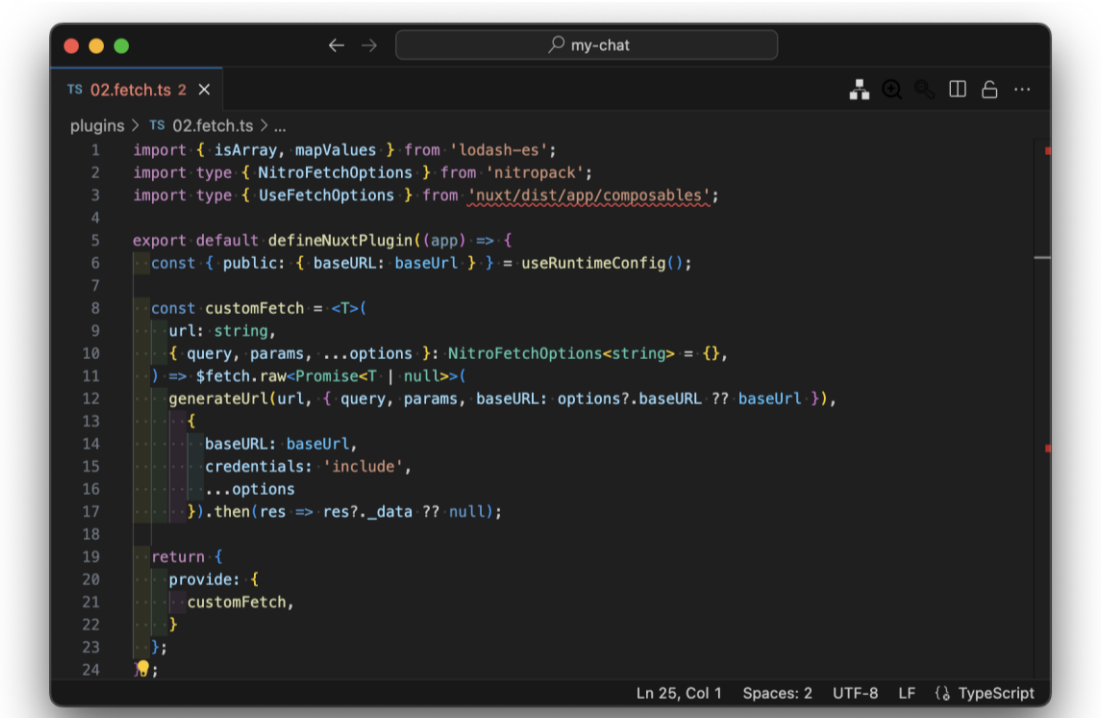

Рисунок 4.1 – Фетч-плагін на основі ofetch та $fetch

Цей плагін Nuxt 3 [19] автоматично додається до застосунку і надає метод customFetch, який можна використовувати для виконання запитів до API з налаштуваннями, що відповідають потребам проєкту.

Підключення до WebSocket сервера

Клієнтський плагін (рис. 4.2) створює підключення до WebSocket сервера за допомогою socket.io-client. Він дозволяє використовувати $io в будь-якому компоненті Nuxt.js застосунку без необхідності повторно налаштовувати підключення.

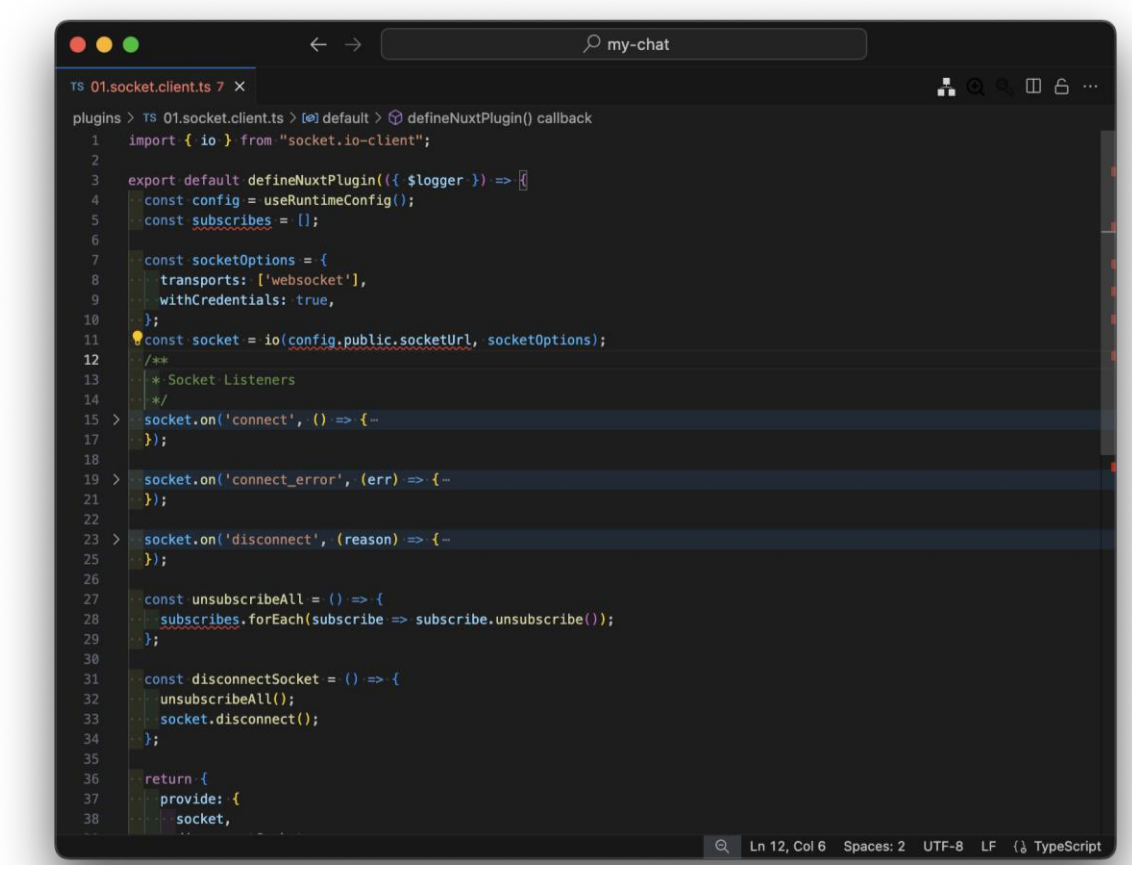

Рисунок 4.2 – Клієнтський плагін для підключення до WebSocket сервера

4.2 Опис backend проєкту

Проєкт є високопродуктивною платформою для реального часу спілкування у чатах, побудованою на основі фреймворку NestJS. Архітектура орієнтована на масштабованість та розширюваність, що досягається за рахунок розбиття логіки на окремі модулі. Кожен модуль відповідає за свою частину функціональності, що полегшує підтримку та розвиток системи.

Основні функціональні можливості розбиті на окремі модулі [20], кожен з яких відповідає за свою частину логіки:

– AuthModule: Відповідає за аутентифікацію користувачів, використовуючи бібліотеку Passport.js.

– UsersModule: Управляє обліковими записами користувачів, зберігає та обробляє дані користувачів.

– ConversationsModule: Обробляє логіку роботи з розмовами між користувачами.

– MessagesModule: Відповідає за відправку та отримання повідомлень у чатах.

– GatewayModule: Реалізує функціональність вебсокетів для реального часу, використовуючи Socket.IO.

– GroupModule: Управляє групами користувачів, дозволяючи створювати та адмініструвати групові чати.

– FriendRequestsModule: Обробляє запити на додавання в друзі.

– FriendsModule: Управляє списком друзів користувачів.

– EventsModule: Обробляє події у реальному часі за допомогою EventEmitter.

– ExistsModule: Перевіряє наявність різних ресурсів у системі.

– MessageAttachmentsModule: Обробляє вкладення у повідомленнях, такі як зображення та файли.

4.2.1 Ініціалізація та налаштування серверної частини

Імпорт бібліотек: На початку коду імпортуються основні бібліотеки, необхідні для створення застосунку. Це включає NestJS для створення серверної частини, TypeORM для роботи з базою даних, Passport.js для аутентифікації, express-session для роботи із сесіями та інші допоміжні бібліотеки. Імпорт цих бібліотек дозволяє використовувати їх функціонал у застосунку.

Функція bootstrap: Головна функція, яка виконується при старті застосунку. Вона створює інстанцію NestJS застосунку, що є основою для всього застосунку. Далі налаштовується сесія, ініціалізується Passport.js для аутентифікації

користувачів та налаштовується адаптер для вебсокетів, що дозволяє використовувати функціональність реального часу.

Налаштування сесії: Використовується TypeORM Store для зберігання сесій у базі даних, що забезпечує надійність та безпеку сесій. Секретний ключ для сесій зберігається у змінних середовища, що гарантує безпеку та гнучкість налаштувань.

Глобальні налаштування: Встановлюються глобальні налаштування для застосунку. Зокрема, використовується CORS для дозволу запитів з інших доменів, глобальна валідація даних через ValidationPipe та префікс для API, що дозволяє організувати структуру маршрутизації.

Запуск сервера: Застосунок слухає на вказаному порті, який також задається через змінні середовища. Це дозволяє запускати застосунок на різних портах у різних середовищах (наприклад, розробка, тестування, продакшн).

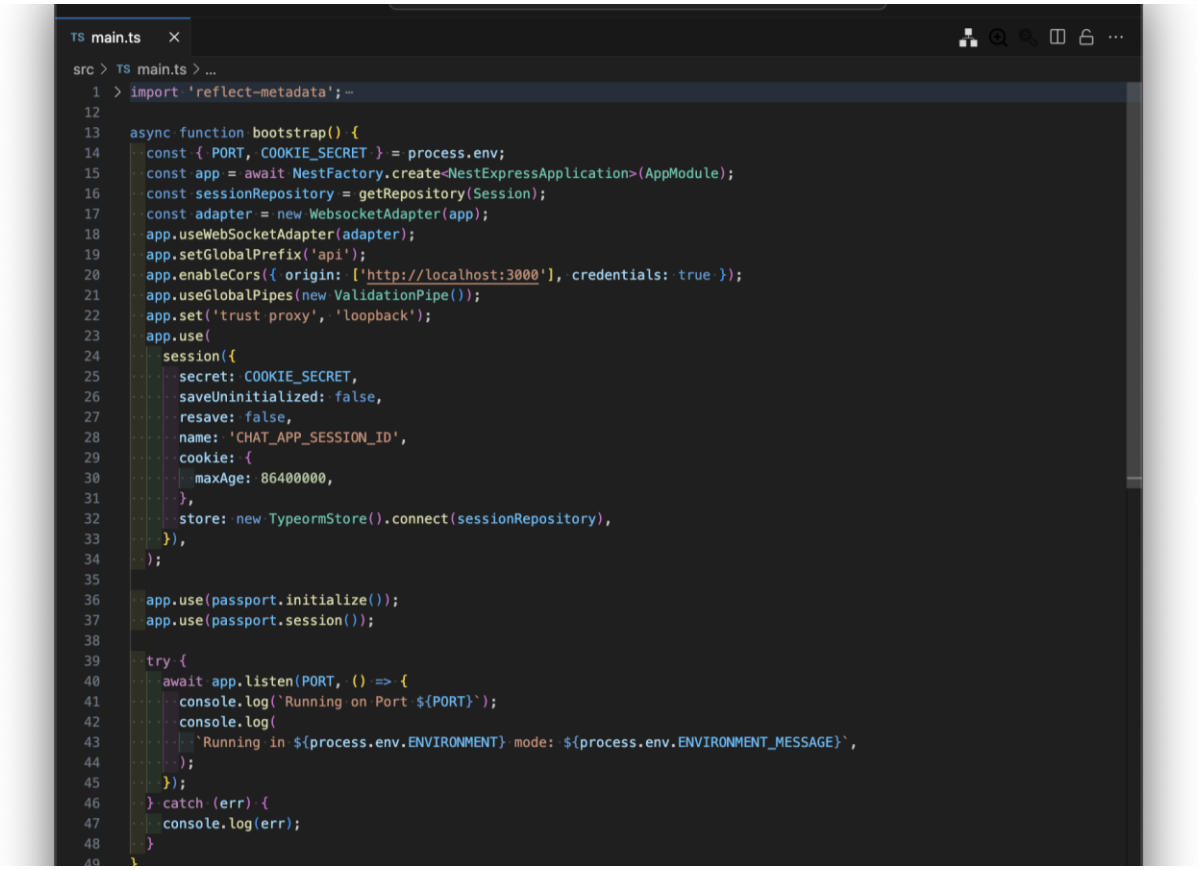

Рисунок 4.3 – Ініціалізація та налаштування серверної частини на основі NestJS

Ці налаштування (див. рисунок. 4.3) забезпечують стабільну роботу серверної частини застосунку.

4.2.2 Управління вебсокетами та обробка подій

Модуль MessagingGateway відповідає за управління вебсокетами та обробку подій, пов'язаних зі спілкуванням у реальному часі. Цей модуль використовує бібліотеки socket.io та @nestjs/websockets [21] для обробки з'єднань, відправлення та отримання повідомлень, а також керування групами та користувачами. Основні функції MessagingGateway включають управління з'єднаннями, обробку подій для отримання онлайн користувачів групи, обробку подій для відправлення та отримання повідомлень, обробку подій для приєднання та виходу з розмов та груп, а також управління подіями введення тексту.

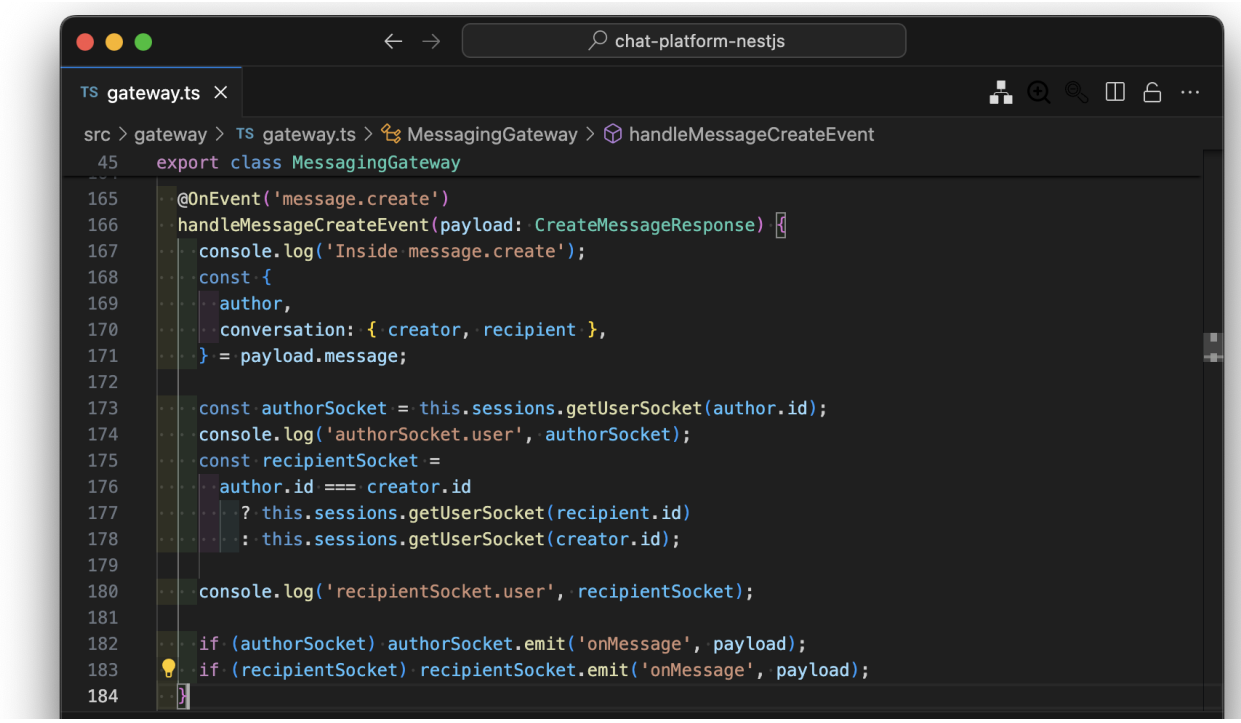

Рисунок 4.4 – Приклад: метод обробки події message.create

Метод на рисунку 4.4 обробляє подію message.create, яка виникає при створенні нового повідомлення. Він отримує дані про повідомлення через CreateMessageResponse, знаходить відповідні сокети автора та одержувача, і відправляє їм подію onMessage з даними повідомлення.

4.3 Опис користувацького інтерфейсу

Інтерфейс чат-застосунку розроблений таким чином, щоб забезпечити користувачам зручний та інтуїтивно зрозумілий досвід спілкування. Він поєднує в собі лаконічний дизайн, чітку структуру та просту навігацію.

На рисунку 4.5 представлена головна сторінка чат-застосунку, що містить список приватних та групових чатів з можливістю пошуку, а також вітальне повідомлення з кнопкою для створення нового чату. Інтерфейс забезпечує зручну навігацію та швидкий доступ до основних функцій спілкування.

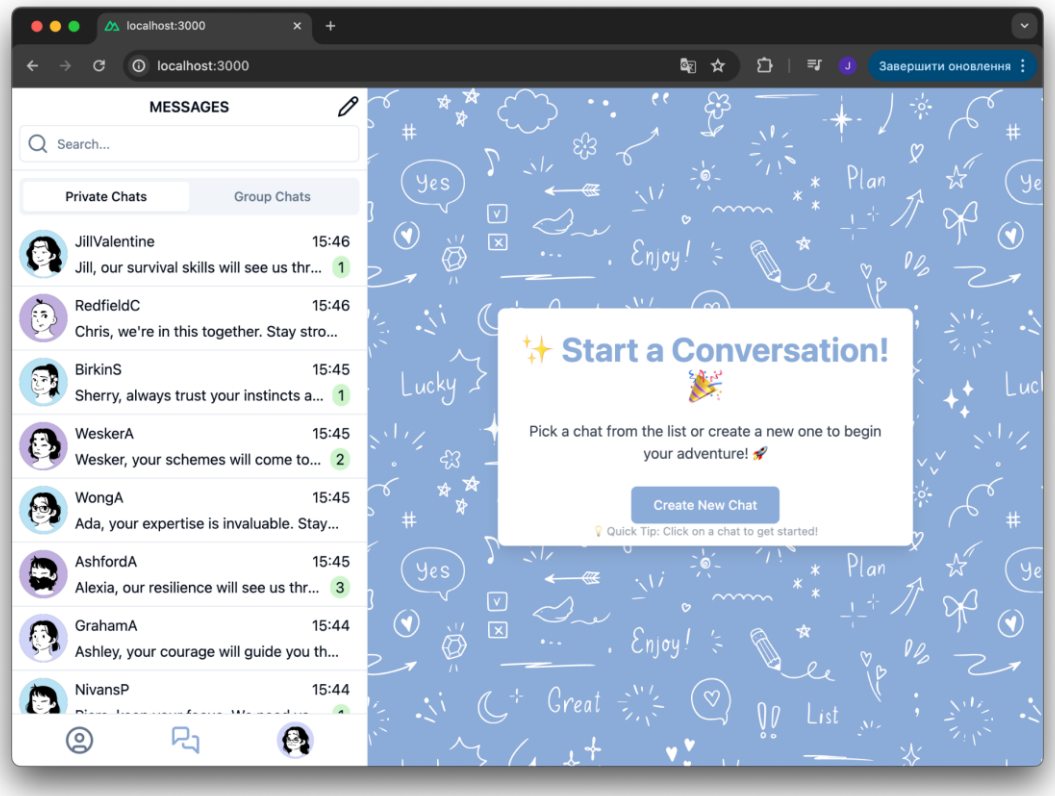

Рисунок 4.5 – Головна сторінка чат-застосунку

На рисунку 4.6 представлено інтерфейс пошуку повідомлень у чаті, що дозволяє користувачам швидко знайти конкретні повідомлення в рамках поточної бесіди. Вікно пошуку розташоване у верхній частині правої панелі, забезпечуючи зручний доступ до функції пошуку та відображаючи результати у реальному часі.

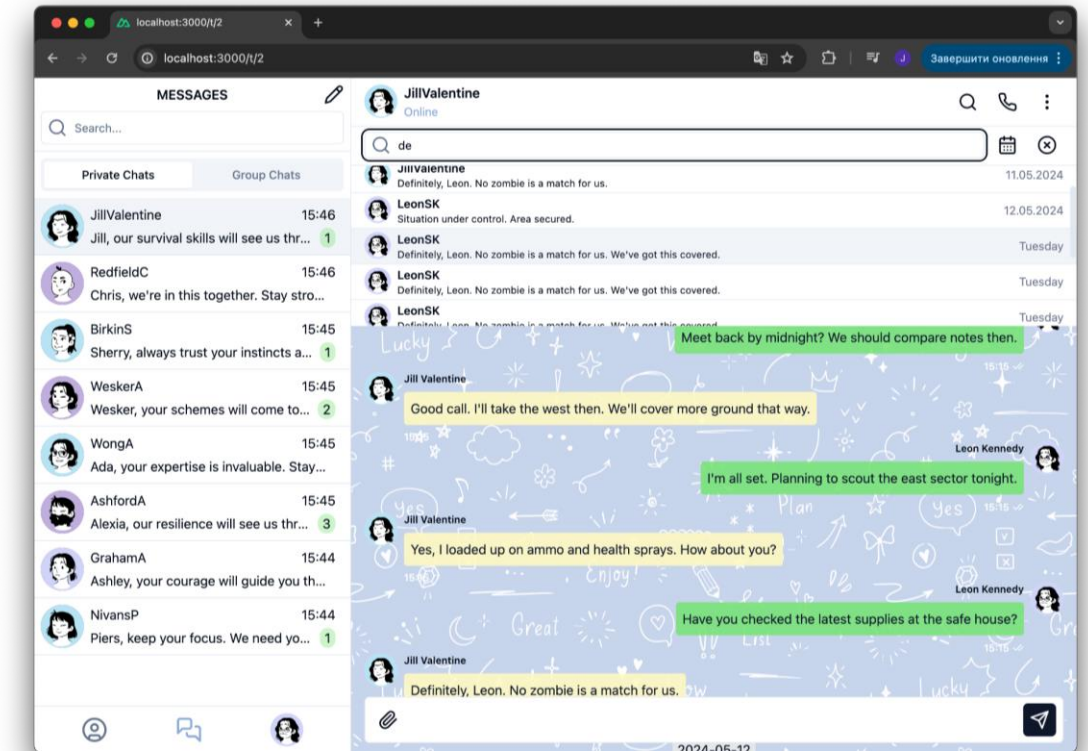

Рисунок 4.6 – Інтерфейс пошуку повідомлень у чаті

На рисунку 4.7 представлено інтерфейс групового чату, який забезпечує зручну комунікацію між учасниками групи. Ліва панель містить список групових чатів, а права панель відображає активний чат з історією повідомлень та можливістю відправки нових повідомлень.

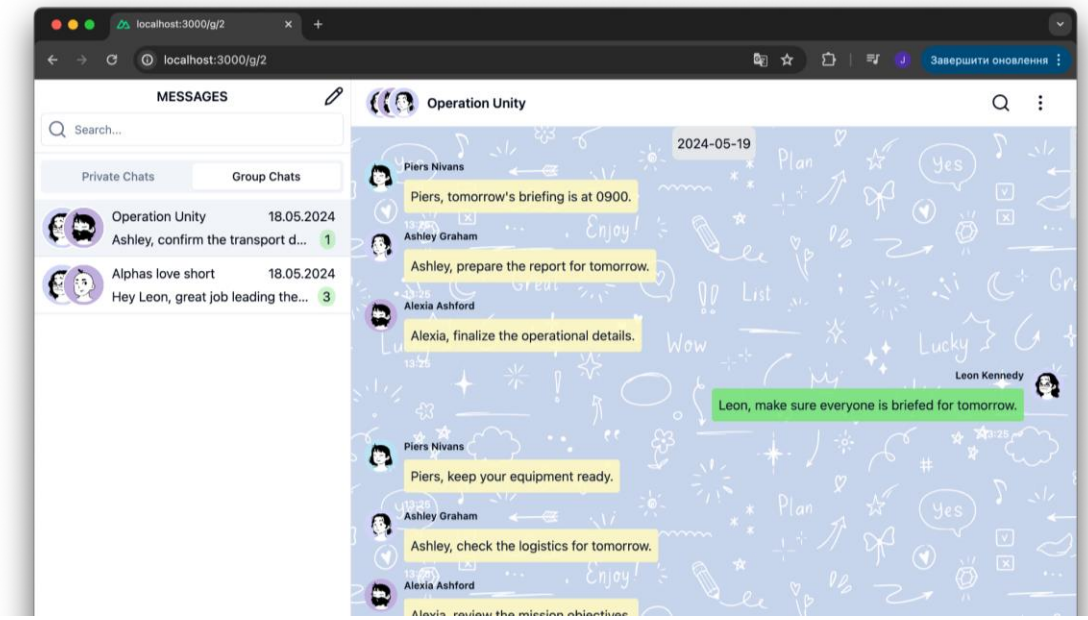

Рисунок 4.7 – Інтерфейс групового чату

Інтерфейс інформації групового чату (рис. 4.8) показує деталі про групу, включаючи назву та список учасників. У правій панелі можна побачити учасників групи з можливістю керування їхніми ролями та правами.

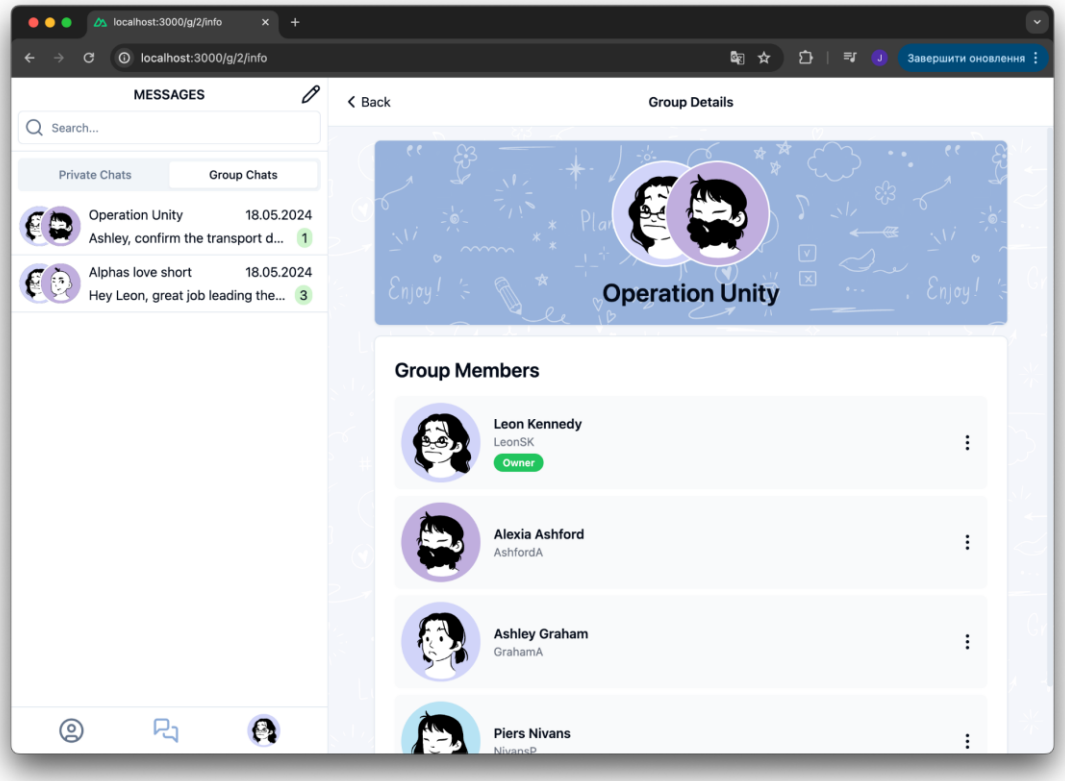

Рисунок 4.8 – Інтерфейс інформації групового чату

Інтерфейс контактів (рис. 4.9) складається з двох частин. Ліва частина показує список контактів, де можна бачити онлайн-статус кожного користувача, а також вкладку для запитів на додавання в друзі. Права частина відображає деталі вибраного контакту, включаючи ім'я, аватар, а також кнопки для надсилання повідомлення та доступу до додаткових дій.

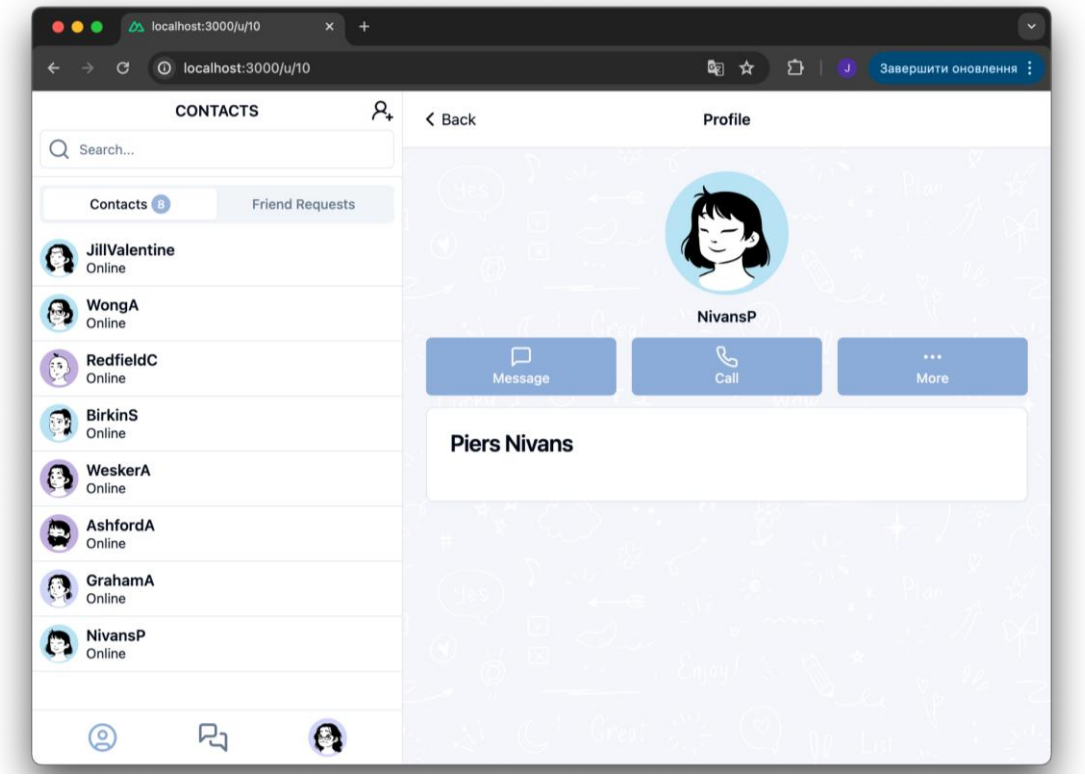

Рисунок 4.9 – Інтерфейс контактів та деталі контакту

Представлені інтерфейси забезпечують зручний доступ до функцій спілкування, включаючи перегляд та пошук повідомлень, управління груповими чатами та контактами.

4.4 Тестування запитів і вебсокетів у Postman

Усі запити HTTPS, HTTP та вебсокети в чат-застосунку були протестовані за допомогою Postman [22]. Це дозволило перевірити правильність роботи API та взаємодії з сервером у різних сценаріях.

4.4.1 Тестування HTTP/HTTPS запитів

На рисунку 4.10 видно, що для тестування використовувалася колекція UltimateMyChatTest, яка містить різноманітні запити для перевірки функціональності застосунку.

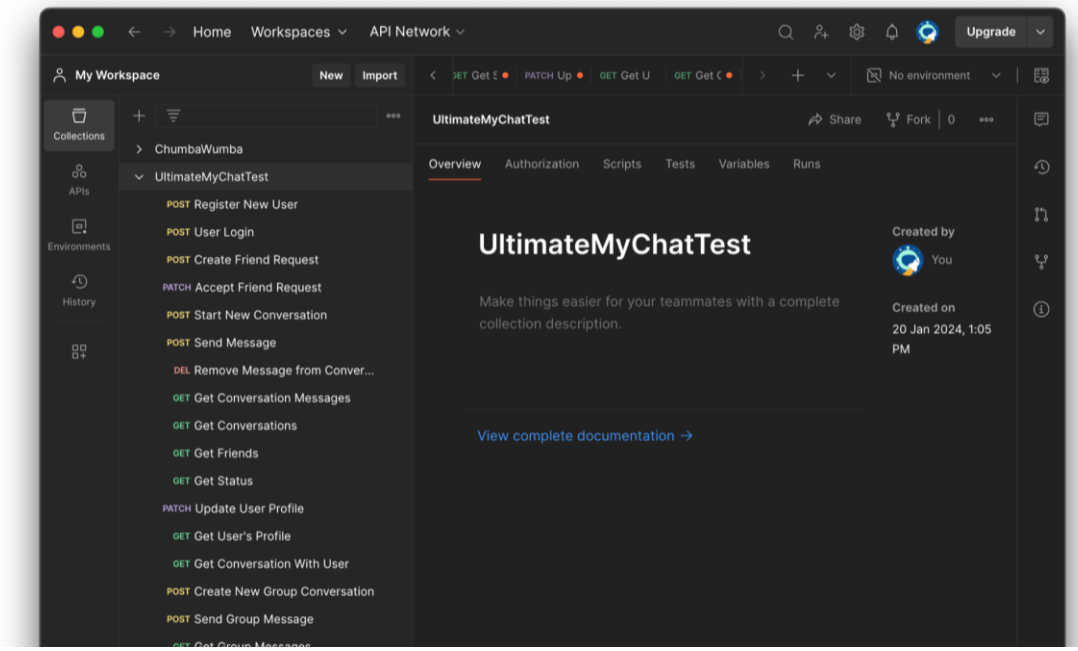

Рисунок 4.10 – UltimateMyChatTest Collection

Ось основні типи запитів, що були протестовані:

– POST: Реєстрація нового користувача, логін, створення заявки на дружбу, початок нової бесіди, відправка повідомлень, створення групового чату, відправка групових повідомлень.

– PATCH: Прийняття заявки на дружбу, оновлення профілю користувача.

– GET: Отримання повідомлень бесіди, отримання списку бесід, отримання списку друзів, отримання статусу користувача, отримання профілю користувача, отримання повідомлень групового чату.

– DELETE: Видалення повідомлення з бесіди.

4.4.2 Тестування вебсокетів

Для тестування вебсокетів використовувався вкладка WS у Postman [23]. На рисунку 4.11 показано підключення до сервера вебсокетів, де налаштовано прослуховування івенту onMessage для перевірки отримання повідомлень у реальному часі.

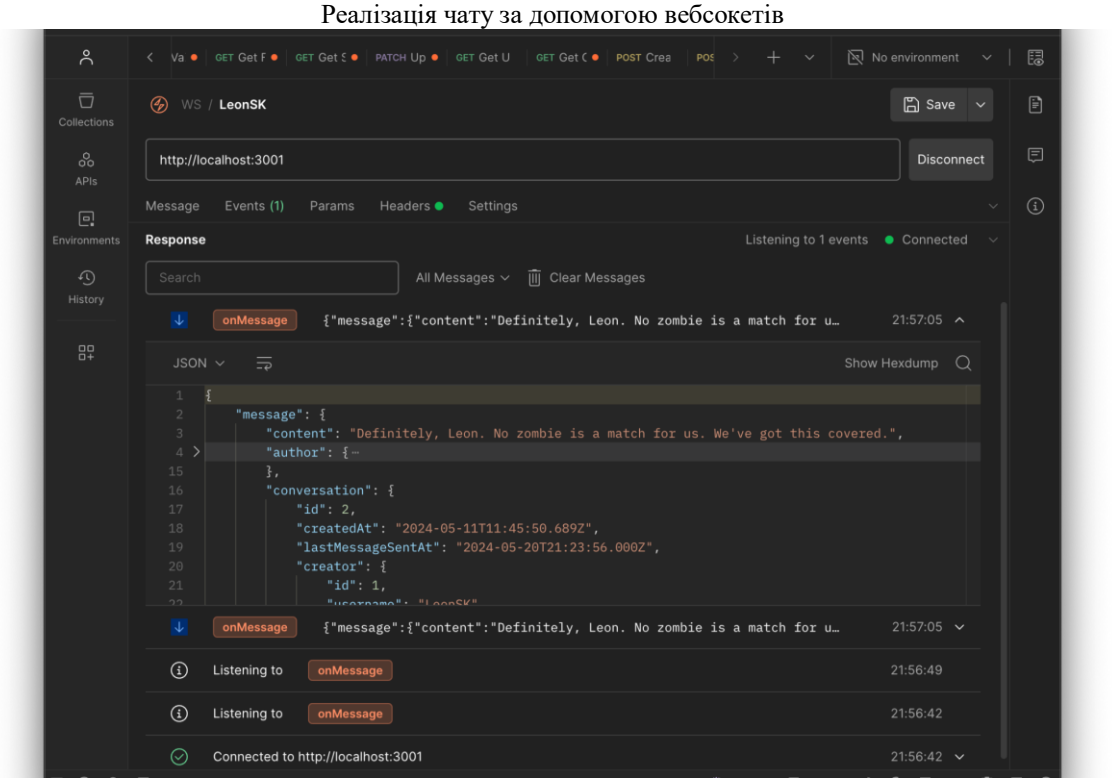

Рисунок 4.11 – Тестування WebSocket у Postman

Процес тестування вебсокетів включав наступні кроки:

– Підключення до вебсокета: Встановлення з'єднання з сервером через URL вебсокета.

– Прослуховування івентів: Налаштування прослуховування івентів, таких як onMessage, для перевірки отримання повідомлень.

– Відправка повідомлень: Надсилання повідомлень до сервера для перевірки, чи правильно сервер обробляє ці повідомлення та надсилає відповідь.

4.4.3 Використання сесійного ID для аутентифікації

Для аутентифікації та тестування використовується сесійний ID, що зберігається в куках. На рисунку 4.12 показано збереження CHAT_APP_SESSION_ID, який використовується для підтримки аутентифікації під час тестування запитів.

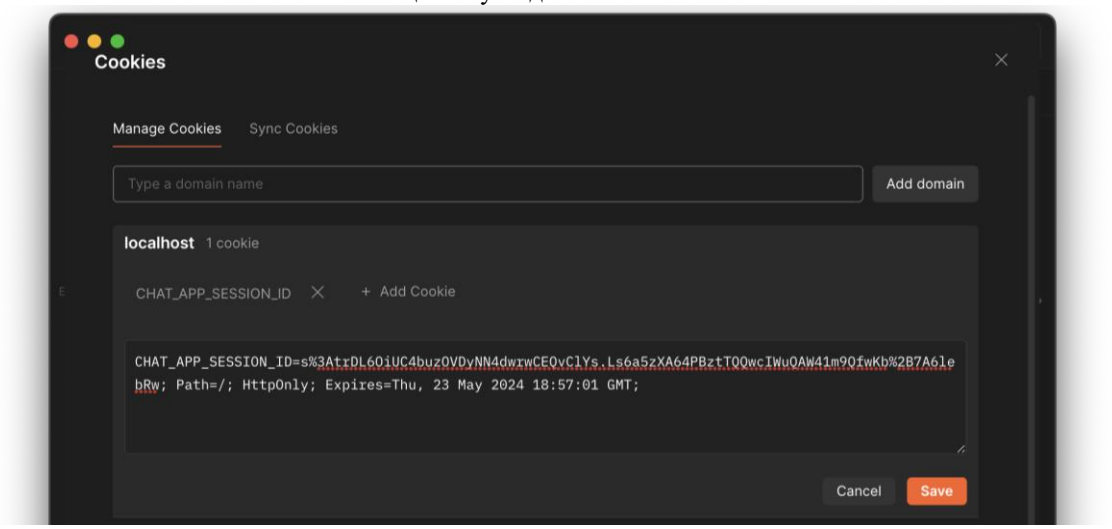

Рисунок 4.12 – Збережений сесійний ID для аутентифікації в Postman

Сесійний ID забезпечує безпечну взаємодію між клієнтом і сервером, дозволяючи виконувати запити лише для аутентифікованих користувачів. Це важливо для підтримки безпеки та забезпечення правильного доступу до даних.

Висновки до розділу 4

У четвертому розділі описано програмну реалізацію чат-застосунку. Проаналізовано структуру проєкту на фронтенді та бекенді, а також взаємодію основних компонентів. Розглянуто налаштування клієнтської частини з використанням Nuxt 3, розробку плагінів для API та WebSocket, а також функціональність серверної частини на базі NestJS.

Описано дизайн та функціональність користувацького інтерфейсу, що забезпечує доступ до функцій чату, таких як пошук повідомлень, групові чати, інформація про групи, контакти та їхні деталі.

Тестування вебсокетів та HTTP/HTTPS запитів за допомогою Postman підтвердило правильність роботи API та взаємодії з сервером. Використання сесійного ID для аутентифікації гарантує безпечну взаємодію між клієнтом і сервером, забезпечуючи стабільність і надійність системи.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра розроблено чат-застосунок із використанням сучасних технологій. Аналіз існуючих аналогів дозволив визначити ключові функціональні можливості та методи, що забезпечують високу якість і ефективність застосунку.

Аналіз показав, що використання технологій Nuxt.js 3 і NestJS дозволяє створити стабільний та ефективний чат-застосунок. Nuxt.js 3 забезпечує високу продуктивність та гнучкість фронтенду, а NestJS – надійність і масштабованість бекенду. Використання Socket.io значно покращило взаємодію користувачів у реальному часі.

Розроблений чат-застосунок відповідає всім вимогам щодо структури та функціональності. Архітектура системи, визначена за допомогою UML-діаграм, забезпечила чіткість та легкість подальшої розробки і масштабування. Програмна реалізація включала налаштування клієнтської частини з Nuxt.js 3 і серверної частини на базі NestJS, що підтвердило стабільність системи під час тестування.

Розробка ефективного чат-застосунку на базі вебсокетів для комунікації в реальному часі була важливою для підтримки корпоративних комунікаційних систем, де потрібен надійний та оперативний обмін інформацією. Чат на базі вебсокетів забезпечує миттєве доставлення повідомлень та ефективну взаємодію, що є ключовим для сучасних комунікаційних сервісів.

Завершуючи кваліфікаційну роботу, можна зазначити, що успішна реалізація проєкту доводить важливість вивчення та використання сучасних вебтехнологій у розробці чат-застосунків. Подальші дослідження в цій галузі можуть сприяти вдосконаленню та розширенню можливостей веброзробки та покращенню користувацького досвіду.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Telegram Software Reviews, Pros and Cons. URL: https://www.softwareadvice.com/customer-communications-mngt/telegram-profile/reviews (Last accessed: 9.02.2024).

2. Signal Review 2024: Secure Messenger (Pros and Cons). 2011. URL: https://restoreprivacy.com/secure-encrypted-messaging-apps/signal/#:~:text=Signal%20is%20a%20secure%2C%20free,top%20privacy%20and%20security%20advocates (Last accessed: 9.02.2024).

3. Facebook Messenger: Free Voice Calling and Text Messaging. URL: https://www.lifewire.com/facebook-messenger-mobile-app-2654387 (Last accessed: 9.02.2024).

4. UML Class Diagram. Wikipedia. URL: https://en.wikipedia.org/wiki/Class_diagram (Last accessed: 9.04.2024).

5. Nuxt.js 3. Nuxt.js. URL: https://nuxt.com/ (Last accessed: 29.03.2024).

6. TypeScript Documentation. TypeScript. URL: https://www.typescriptlang.org/docs/ (Last accessed: 29.03.2024).

7. Tailwind CSS. Tailwind CSS. URL: https://tailwindcss.com/ (Last accessed: 9.04.2024).

8. Socket.IO Client API. Socket.IO. URL: https://socket.io/docs/v4/client-api/ (Last accessed: 29.03.2024).

9. Shadcn-Nuxt UI Kit. GitHub Topics. URL: https://github.com/topics/shadcn-ui?l=vue (Last accessed: 29.03.2024).

10. Vee-Validate. GitHub. URL: https://github.com/logaretm/vee-validate (Last accessed: 29.03.2024).

11. Sass Guide. Sass. URL: https://sass-lang.com/guide/ (Last accessed: 29.03.2024).

12. Nest.js Framework. Nest.js. URL: https://nestjs.com/ (Last accessed: 2.04.2024).

13. TypeORM Documentation. TypeORM. URL: https://typeorm.io/ (Last accessed: 2.04.2024).

14. Passport.js Authentication Middleware. Passport.js. URL: https://www.passportjs.org/ (Last accessed: 2.04.2024).

15. bcrypt NPM Package. npm. URL: https://www.npmjs.com/package/bcrypt (Last accessed: 2.04.2024).

16. MySQL Documentation. MySQL. URL: https://dev.mysql.com/doc/mysql-getting-started/en/ (Last accessed: 2.04.2024).

17. DBeaver Database Tool. DBeaver. URL: https://dbeaver.io/download/ (Last accessed: 2.04.2024).

18. Nuxt.js Data Fetching. Nuxt.js. URL: https://nuxt.com/docs/getting-started/data-fetching (Last accessed: 2.04.2024).

19. Nuxt.js Plugins. Nuxt.js. URL: https://nuxt.com/docs/guide/directory-structure/plugins (Last accessed: 2.04.2024).

20. NestJS Modules. NestJS. URL: https://docs.nestjs.com/modules (Last accessed: 3.04.2024).

21. NestJS WebSockets Gateways. NestJS. URL: https://docs.nestjs.com/websockets/gateways (Last accessed: 3.04.2024).

22. Postman Tools. Postman. URL: https://www.postman.com/product/tools/ (Last accessed: 2.05.2024).

23. Postman WebSocket Documentation. Postman. URL: https://www.postman.com/postman/workspace/websockets/documentation/14057978-712d684f-c252-4bd9-a7a6-6a893e41adea (Last accessed: 2.05.2024).