Міністерство освіти і науки України  
Національний технічний університет України

«Київський політехнічний інститут ім. І. Сікорського»

Кафедра інженерії програмного забезпечення в енергетиці

Лабораторна робота № 4

з курсу: «*Новітні мови програмування*»

**Виконала:**  
студентка 4-го курсу,  
групи ТВ-13  
Новотка Вікторія Іванівна

Посилання на GitHub репозиторій:

[https://github.com/Victoriia-Ripka/](https://github.com/Victoriia-Ripka/Go)Rust

**Перевірила:**

Пироговська Т. В.

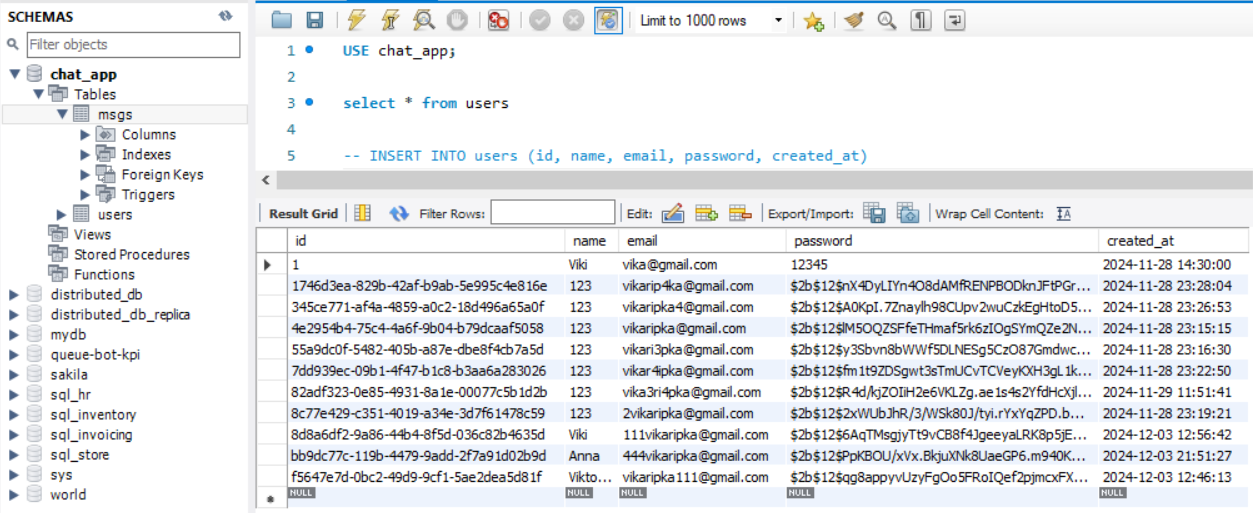
Київ 2024/2025

**Хід виконання:**

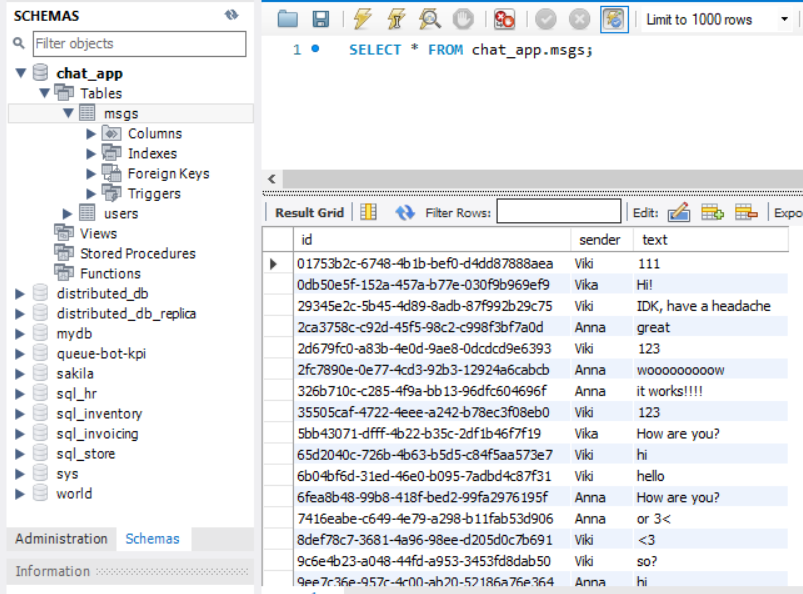
Під час виконання практичної роботи №4 нами було розроблено застосунок обміну повідомлень між користувачами у реальному часі.

Всі дані (користувачі і повідомлення) зберігаються у БД MySQL

Структура даних користувачів наступна:



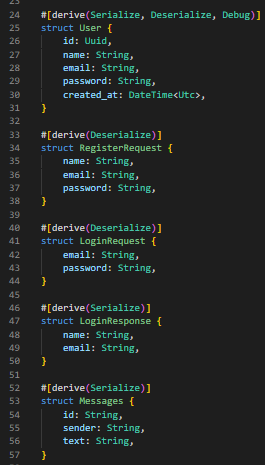
Структура даних повідомлень наступна:



Бекенд проєкту має 4 рути. Три з них асинхронні і один для WebSocket Server. Ми зробили підключення до БД, запускаємо вебсокет сервер, зберігаємо стан застосунку, і запускаємо HTTP сервер. Використовували тільки методи POST і GET. Стан застосунку зберігає в собі вебсокет сервер і підключення до БД, щоб передавати глобальні дані у кожну функцію з рута.

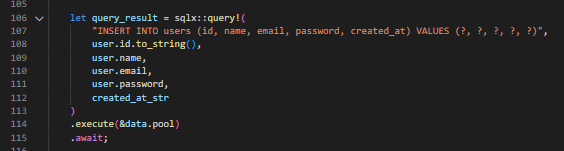


Було створено дуже багато стурктур. Для асинхронних функцій вони є наступними:

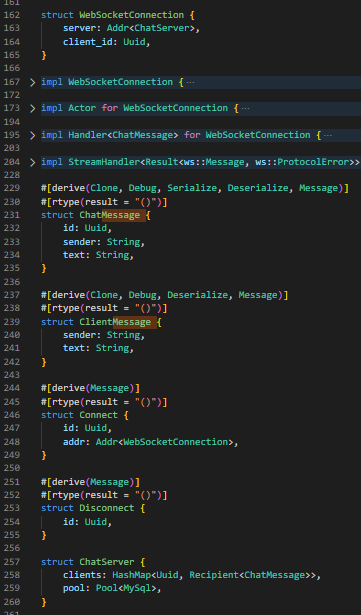


До всіх структур використовували макроси для роботи з JSON форматом, для дебагу і клонування.

Запити до БД виглядають схожим чином між усіма функціями. Приклад наведено нижче:

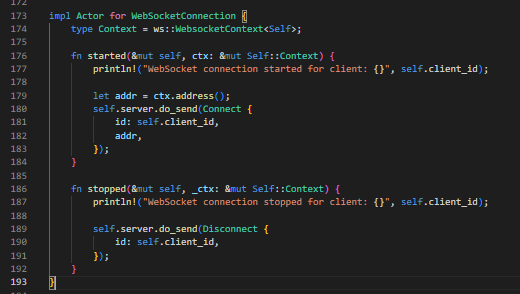


Стурктури, що були створені для вебсокет сервера наступні



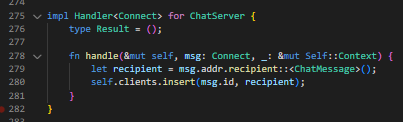
WebSocketConnection відповідає за з’єднання клієнта із сервером. Містить унікальний ідентифікатор користувача і адресу сервера. Структури Connect і Disconnect відповідають за з’єднання і роз’єднання користувача із сервером вебсокетів. Структура ChatServer є головним сервером чат-програми, містить мапу співвідношень між клієнтами і отримувачем повідомлень, а також з’єднання з БД.

Наступний блок коду описує поведінку актора, що підключається до вебсокет серверу. Описує хід дій при підключенні актора до сервера і при відключенні. Саме тут використовуємо структури Connect і Disconnect.



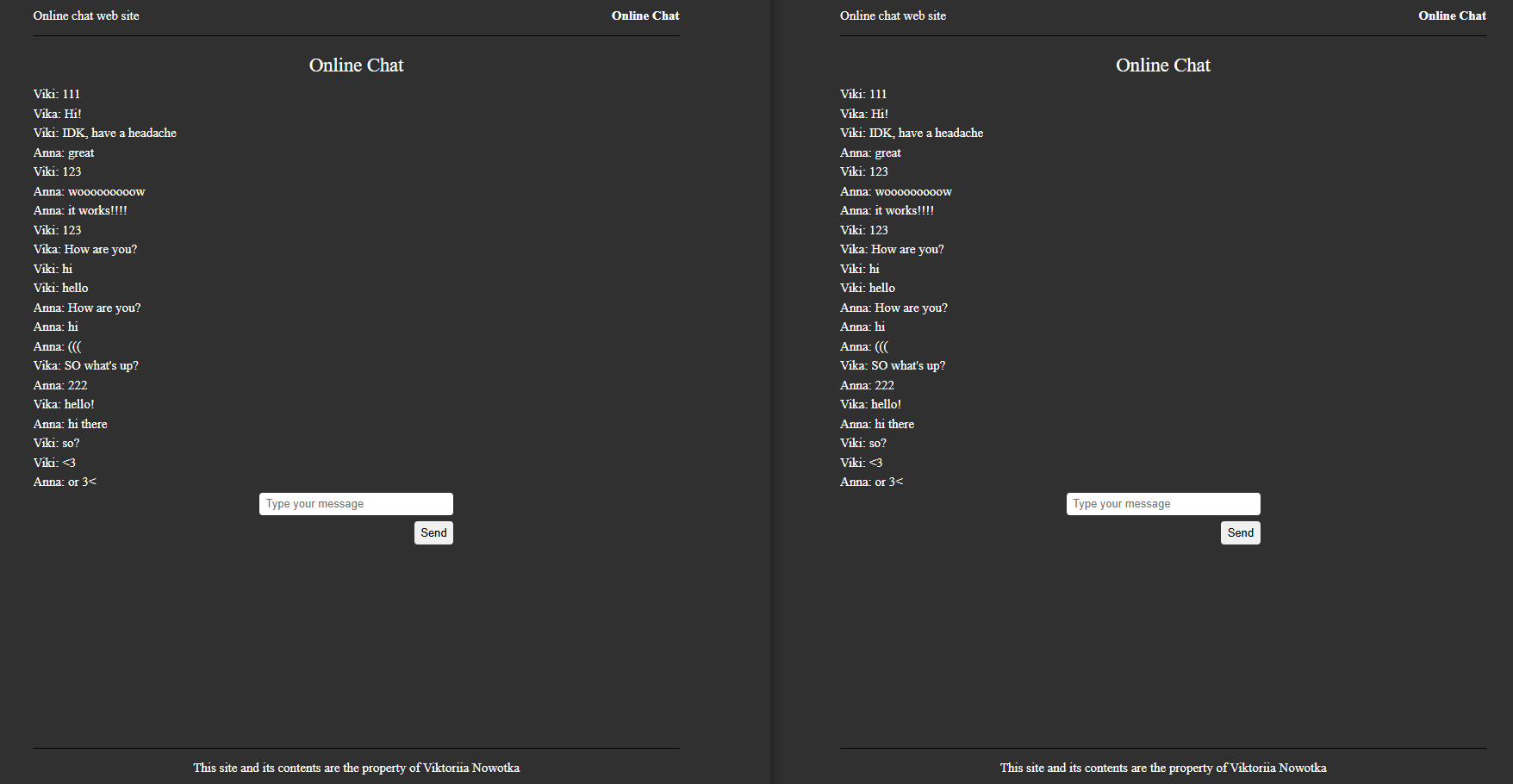
Клієнти можуть надсилати повідомлення через WebSocket, які сервер потім обробляє і передає іншим клієнтам через ChatMessage. за це відповідає StreamHandler і Handler.

Створимо також окремий Handler для обробки підключень до чат серверу. Результатом роботи буде додавання усіх користувачів до мапи підключених клієнтів. Аналогічний Handler створений для відключення.

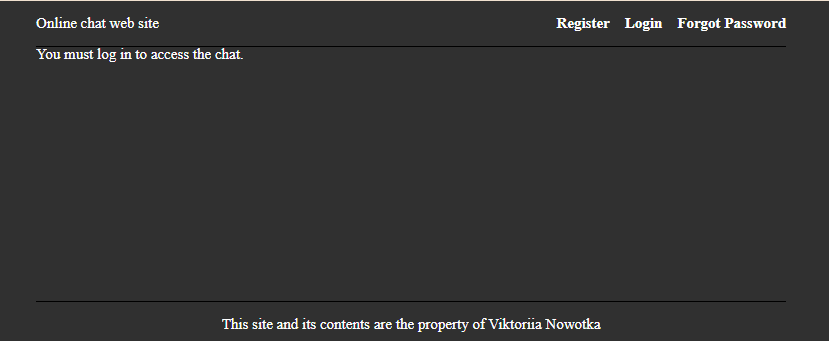


Головною частиною вебсокет сервера є обробка повідомлень. Handler для структури ClientMessage отримує повідомлення від підключеного клієнта, оброблює отримані поля і робить запит до бази даних. Якщо повідомлення успішно записано у БД – надсилається усім користувачам.

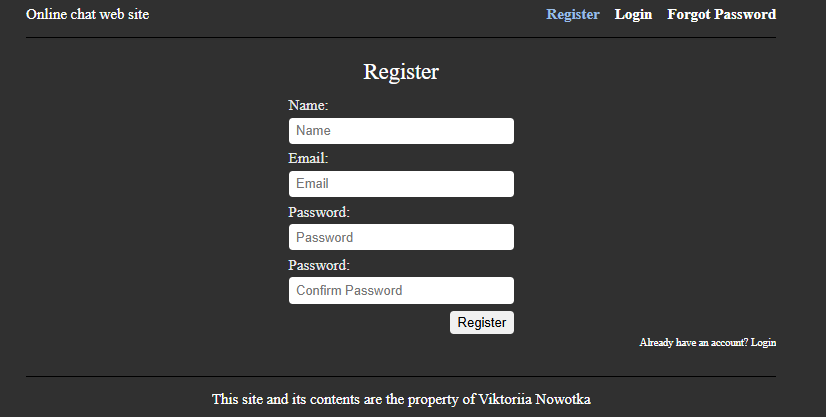
Так виглядає чат застосунку:



Так виглядає, якщо користувач не зареєстрований:



Форма реєстрації:



Форма логіну:

