УДК 004.4; 004.738.5; 004.056
А. Бачинський; А. Бачинський
(Тернопільський національний технічний університет імені Івана Пулюя, Україна)

ДОСЛІЖЕННЯ ЗАСОБІВ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ ОБМІНУ ПОВІДОМЛЕННЯМИ В РЕАЛЬНОМУ ЧАСІ З АУТЕНТИФІКАЦІЄЮ КОРИСТУВАЧІВ

UDC 004.4; 004.738.5; 004.056
A. Bachynskiy; A. Bachynskyi

SOFTWARE DEVELOPMENT TOOLS RESEARCH FOR REAL-TIME MESSAGING WITH USERS AUTHENTICATION

В ході вивчення підходів до створення програмного забезпечення для обміну повідомленнями в реальному часі було досліджено ряд технологій розробки та програмних фреймворків. В процесі розробки програмного забезпечення для обміну повідомленнями в реальному часі для реалізації функціоналу на клієнтській стороні можна використати бібліотеку **React JS**, що забезпечує ефективне керування станом інтерфейсу користувача [1].

Серверна частина реалізована з використанням **Node.js** та **Express.js**, що дозволяє здійснювати обробку HTTP-запитів та забезпечувати взаємодію з клієнтами через API [2]. Для реалізації реального часу комунікації між сервером і клієнтом застосовано **WebSocket** за допомогою бібліотеки **Socket.io**, що забезпечує двосторонній зв'язок і дозволяє передавати повідомлення без перезавантаження сторінки.

Під час взаємодії клієнта з сервером клієнт ініціює з'єднання через **Socket.io**. Після встановлення з’єднання клієнт може надсилати повідомлення серверу, а сервер, у свою чергу, миттєво передає це повідомлення всім підключеним клієнтам або певному користувачеві, залежно від вимог бізнес-логіки [3].

Це забезпечує високий рівень інтерактивності та дозволяє зберігати зв'язок між користувачами в реальному часі.

Коли клієнт надсилає повідомлення, воно передається через відкритий канал WebSocket на сервер, де воно обробляється та зберігається в базі даних (у даному випадку, використовується **MongoDB**) [4]. Після обробки повідомлення сервер надсилає його назад всім підключеним користувачам через **Socket.io**. Завдяки цьому користувачі можуть отримувати нові повідомлення без необхідності перезавантаження сторінки або повторних запитів.

Ключовою перевагою цього підходу є забезпечення низької затримки при передачі даних і можливість постійного збереження з'єднання між клієнтом і сервером. Використання **Socket.io** дозволяє також обробляти різні події в реальному часі, наприклад, повідомлення про нові вхідні дані або розрив з'єднання, що робить систему більш стабільною і зручнішою для користувачів. Така реалізація дозволяє забезпечити швидку доставку повідомлень у реальному часі з мінімальними затримками.

Одним з основних аспектів системи є забезпечення безпеки користувачів, яке реалізовано через механізм аутентифікації за допомогою **JWT (JSON Web Token)**.

У процесі аутентифікації користувачі проходять етап реєстрації та входу, після чого отримують унікальний токен, який використовується для доступу до захищених ресурсів системи [5].

Токен зберігається на клієнтському боці і передається разом з кожним запитом до серверу для перевірки авторизації користувача. Така система дозволяє забезпечити безпечний доступ до функцій, що вимагають ідентифікації користувача, без необхідності зберігання сесійної інформації на сервері [6].

Таблиця 1. Характеристика використаних технологій

Технологія	Опис	Переваги	Недоліки
React JS	Бібліотека для створення інтерфейсу користувача на основі компонентів	Підтримка компонентного підходу, висока продуктивність при рендерингу, гнучкість у використанні	Необхідність додаткових інструментів для повноцінної розробки складних додатків
Node.js	Платформа для виконання JavaScript на сервері	Підтримка асинхронної обробки запитів, висока ефективність обробки I/O операцій, масштабованість	Можливе низьке виконання в CPU-інтенсивних задачах
Express.js	Мінімалістичний веб-фреймворк для Node.js	Спрощує створення серверних API, підтримка middleware для обробки запитів	Необхідність додаткових налаштувань для великих проектів
MongoDB	Документо-орієнтована база даних	Масштабованість, швидкість виконання запитів до неструктурованих даних, зручність для зберігання JSON-даних	Обмежена підтримка транзакцій у порівнянні з реляційними базами даних
Socket.io	Бібліотека для двостороннього зв'язку між клієнтом і сервером	Забезпечення реального часу комунікації, підтримка WebSocket, проста інтеграція	Погіршення роботи за умов нестабільного з'єднання

Це дозволяє знизити навантаження на сервер та підвищити рівень безпеки через використання криптографічних алгоритмів для генерації та перевірки токенів.

Література

1. Behrendt, K. & Lumsden, A. (2022). Real-Time Messaging Systems: Architectures and Techniques. Springer. Available at https://reactjs.org/.
2. Möller, J. & Neumann, P. (2021). Mastering Node.js: Building Scalable and Fast Web Applications. Packt Publishing.
1. 3 Abrams, P. (2023). Real-Time Web Application Development with Node.js and Socket.io. O'Reilly Media. Available at https://socket.io/docs/.
3. Garvin, S. (2020). Learning MongoDB: A Guide to NoSQL Data Storage for Modern Web Apps. O'Reilly Media.
4. Schmidt, A. (2023). Building Real-Time Applications with Node.js, MongoDB, and JWT Authentication. Packt Publishing.
5. Hamedani, M. (2021). MERN Stack React, Node, Express, MongoDB - Web Development. Udemy Course