УДК 004.738.5

Р. Ю. Чехместрук
Н. О. Панасов

РОЗРОБКА МЕТОДУ ТА ПРОГРАМНИХ ЗАСОБІВ СИНХРОНІЗАЦІЇ ДАНИХ У ВЕБ-ОРІЄНТОВАНИХ СИСТЕМАХ КОМАНДНОЇ РОЗРОБКИ

Вінницький національний технічний університет

Анотація

Розглянуто проблематику забезпечення цілісності та миттєвого оновлення даних у системах командної розробки. Запропоновано архітектурне рішення на базі повнодуплексного протоколу WebSockets із використанням сучасного технологічного стека (NestJS та Angular). Проаналізовано переваги такого підходу порівняно з традиційними методами HTTP-опитувань.

Ключові слова: вебзастосунок; синхронізація даних; WebSockets; NestJS; Angular; командна розробка.

Abstract

The problem of ensuring data integrity and instant updating in collaborative development systems is considered. An architectural solution based on the full-duplex WebSockets protocol using a modern technology stack (NestJS and Angular) is proposed. The advantages of this approach compared to traditional HTTP polling methods are analyzed.

Keywords: web application; data synchronization; WebSockets; NestJS; Angular; collaborative development.

Вступ

Сучасний етап розвитку інженерії програмного забезпечення характеризується стрімким переходом до інструментів колективної роботи, таких як Jira, Trello чи Planka [1]. У таких системах критично важливою є актуальність даних у реальному часі. Проте більшість багатофункціональних рішень продовжують покладатися на традиційні архітектурні стилі (REST API) або методи довгого опитування (Long Polling) [2, 3], що призводить до затримок у синхронізації та конфліктів при спільному редагуванні. З іншого боку, швидкі open-source рішення часто мають занадто обмежений набір інструментів. Отже, актуальним є розробка оптимізованих модулів, що поєднують багатий функціонал із подієво-орієнтованою архітектурою для моментального обміну даними.

Основна частина

Традиційна модель взаємодії у вебі базується на принципі «запит-відповідь» протоколу HTTP, де ініціатором обміну завжди виступає клієнт. Для систем командної роботи, де стан об'єктів може змінюватися щосекунди різними користувачами, цей підхід створює надмірне навантаження на мережу через передачу великих заголовків та утримання зайвих з'єднань [3].

Найбільш прогресивною технологією для вирішення цієї проблеми є стандартизований протокол WebSockets (RFC 6455) [4]. Він забезпечує повнодуплексний зв'язок через одне постійне TCP-з'єднання, що дозволяє серверу ініціювати передачу даних клієнту в момент виникнення події, оминаючи затримки HTTP-запитів.

Для реалізації модулів синхронізації було обрано стек технологій, що включає Node.js-фреймворк NestJS [5] для серверної частини та платформу Angular для клієнтської. Серверна архітектура побудована навколо парадигми шлюзів (Gateways) на базі бібліотеки Socket.io [6], що дозволяє інкапсулювати низькорівневу роботу з сокетами. Програмні модулі бекенду використовують декоратори для маршрутизації подій. Важливим архітектурним рішенням є використання концепції «кімнат» (rooms), що дозволяє ізолювати потоки даних.

Клієнтська частина, розроблена на Angular, використовує реактивний підхід за допомогою бібліотеки RxJS [7] для асинхронної обробки вхідних потоків даних. Це забезпечує миттєве оновлення графічного інтерфейсу: переміщення карток завдань чи зміна статусів автоматично ініціюють перемальовування відповідних компонентів у всіх учасників команди без необхідності перезавантаження браузера (reload).

Для обґрунтування переваг обраного підходу було проведено порівняльний аналіз архітектурних особливостей популярних існуючих платформ та запропонованого програмного рішення. Результати порівняння за ключовими критеріями наведено в таблиці 1.

Таблиця 1 – Порівняльна характеристика рішень для командної розробки

Критерії порівняння	Jira Software	Trello	Planka	Запропоноване рішення
Базова архітектура обміну даними	REST API	Long Polling	WebSockets	WebSockets
Оновлення інтерфейсу без reload	Частково	Частково	Повністю	Повністю
Функціональна насиченість	Висока	Середня	Низька	Висока
Синхронізація при конкурентному доступі	Низька	Середня	Базова	Висока

Детальний аналіз даних, наведених у таблиці 1, дозволяє виявити специфіку кожної із систем. Корпоративна платформа Jira Software є потужним інструментом з високою функціональною насиченістю, проте її архітектура на базі REST API вимагає від клієнта постійного ініціювання запитів. Це унеможливлює повноцінне оновлення інтерфейсу без примусового перезавантаження (reload) та значно знижує надійність системи при одночасному конкурентному доступі багатьох користувачів до одного завдання.

Система Trello частково вирішує цю проблему за допомогою методів Long Polling, що покращує чутливість інтерфейсу порівняно з Jira. Однак така модель зв'язку залишається гібридною і не гарантує миттєвої синхронізації при пікових навантаженнях, покладаючись на компроміс між середньою функціональністю та середньою швидкістю оновлення.

У свою чергу, open-source проєкт Planka використовує WebSockets, забезпечуючи ідеальне оновлення в реальному часі за моделлю Push (від сервера до клієнта). Незважаючи на технічну досконалість обміну даними, ця система суттєво програє лідерам ринку через низьку функціональну насиченість: їй бракує розвиненої системи управління контентом та складних механізмів обробки конфліктів даних.

Запропоноване рішення на базі NestJS та Angular дозволяє об’єднати переваги розглянутих систем. Використання повнодуплексного обміну даними (як у Planka) вирішує ключові недоліки традиційних систем (Jira, Trello), зберігаючи при цьому можливість реалізації складного функціоналу. Передача кожної події у розробленому застосунку супроводжується часовими мітками та ідентифікаторами версій об'єктів, що дозволяє серверу коректно обробляти ситуації конкурентного доступу до даних.

Висновки

У результаті проведеної роботи розроблено програмні модулі системи синхронізації даних, які дозволяють ефективно оновлювати стан об’єктів у вебзастосунках реального часу. Використання повнодуплексного протоколу WebSockets у поєднанні з подієво-орієнтованою моделлю NestJS та реактивним підходом Angular дозволило повністю усунути затримки, притаманні класичним REST-системам. Впровадження таких архітектурних рішень забезпечує безшовну командну взаємодію, гарантує цілісність даних при конкурентному доступі та значно підвищує загальну продуктивність розробницьких команд.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2015. 816 p.
2. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures : Ph.D. dissertation. Irvine : University of California, 2000. 162 p.
3. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach. 7th ed. Boston : Pearson, 2016. 864 p.
4. The WebSocket Protocol. RFC 6455 [Електронний ресурс]. Режим доступу: https://datatracker.ietf.org/doc/html/rfc6455. Дата звернення: 21.03.2026 р.
5. Офіційна документація фреймворку NestJS [Електронний ресурс]. Режим доступу: https://docs.nestjs.com/. Дата звернення: 21.03.2026 р.
6. Офіційна документація бібліотеки Socket.IO [Електронний ресурс]. Режим доступу: https://socket.io/docs/v4/. Дата звернення: 21.03.2026 р.
7. Офіційна документація платформи Angular (RxJS) [Електронний ресурс]. Режим доступу: https://angular.io/guide/rx-library. Дата звернення: 21.03.2026 р.

Чехместрук Роман Юрійович – кандидат технічних наук, доцент кафедри програмного забезпечення, Факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: chechm@vntu.edu.ua.

Панасов Нікіта Олегович – студент групи 6ПІ-22б, Факультет інформаційних технологій та комп'ютерної інженерії, Вінницький національний технічний університет, м. Вінниця, e-mail: nikitapanasov22@gmail.com.

Roman Chekhmestruk – Candidate of Technical Sciences, Associate Professor of the Department of Software Engineering, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia.

Nikita Panasov – student of group 6PI-22b, Faculty of Information Technology and Computer Engineering, Vinnytsia National Technical University, Vinnytsia.