Національний технічний університет України «КПІ ім. Ігоря Сікорського»

Факультет Інформатики та Обчислювальної Техніки

Кафедра інформаційних систем та технологій

Лабораторна робота №6

з дисципліни «Проектування інформаційних систем»

на тему

«Реалізація

патерну управління, заснованого на

подіях, протоколом WebSockets»

Виконала:

студентка групи ІС-34

Ященко О.О.

Викладач:

Ковальов М.О

Київ – 2025

**Завдання:** Організувати двонаправлений обмін повідомленнями з сервером за допомогою протоколу прикладного рівня WebSockets. Запрограмувати корпоративний чат, зміст якого сформульований в описі предметної області. Реалізується лише мінімальна конфігурація — тобто без доступу до бази даних, дозволяється використання захардкожених даних. Для ознайомлення з WebSocket-протоколом дозволено використовувати echo-сервер або інші прості приклади.

**Виконання**

Посилання на GitHub репозиторій   
<https://github.com/yash4enk0/PIS_5-6>

Для реалізації поставленого завдання був використаний мінімальний стек технологій:

Node.js — для реалізації простого WebSocket-сервера.

Бібліотека ws — як реалізація WebSocket-протоколу на сервері.

HTML + JavaScript — для створення клієнтської частини корпоративного чату.

**Опис предметної області:**

Предметна область — це рух основних засобів (ОЗ). Одним із важливих документів є Акт введення в експлуатацію, який формує бухгалтер. Необхідно, щоб всі відповідальні особи (користувачі чату) миттєво отримували повідомлення про введення ОЗ в експлуатацію.

**Опис роботи серверної частини (Node.js):**

Запускається HTTP-сервер на порті 8080, який також обробляє запит на / та повертає HTML-сторінку.

Через WebSocket (ws) запускається сервер, який прослуховує підключення клієнтів.

Кожен новий клієнт отримує привітальне повідомлення.

Коли будь-який клієнт надсилає повідомлення, воно транслюється всім іншим клієнтам — це і є реалізація "подієвої моделі".

**Код серверної частини (server.js):**

const WebSocket = require('ws');

const http = require('http');

const fs = require('fs');

const path = require('path');

const server = http.createServer((*req*, *res*) => {

if (*req*.url === '/') {

const filePath = path.join(\_\_dirname, 'index.html');

fs.readFile(filePath, (*err*, *data*) => {

if (*err*) {

*res*.writeHead(500);

return *res*.end('Error loading index.html');

}

*res*.writeHead(200, { 'Content-Type': 'text/html' });

*res*.end(*data*);

});

}

});

const wss = new WebSocket.Server({ server });

wss.on('connection', *ws* => {

*ws*.on('message', *message* => {

wss.clients.forEach(*client* => {

if (*client*.readyState === WebSocket.OPEN) {

*client*.send(*message*.toString());

}

});

});

*ws*.send('✅ Підключено до корпоративного чату.');

});

const PORT = 8080;

server.listen(PORT, () => {

console.log(`Сервер запущено на http://localhost:${PORT}`);

});

**Код клієнтської частини (index.html):**

<!DOCTYPE html>

<html lang="uk">

<head>

<meta charset="UTF-8" />

<title>Корпоративний чат ОЗ</title>

<style>

body { font-family: Arial; margin: 20px; }

#messages { border: 1px solid #ccc; padding: 10px; height: 200px; overflow-y: scroll; }

#input { width: 80%; }

</style>

</head>

<body>

<h2>Корпоративний чат: Введення в експлуатацію ОЗ</h2>

<div id="messages"></div>

<input id="input" placeholder="Введіть повідомлення про введення ОЗ..." />

<button onclick="sendMessage()">Надіслати</button>

<script>

const ws = new WebSocket(`ws://${location.host}`);

const messages = document.getElementById('messages');

ws.onmessage = *event* => {

const msg = document.createElement('div');

msg.textContent = *event*.data;

messages.appendChild(msg);

messages.scrollTop = messages.scrollHeight;

};

function sendMessage() {

const input = document.getElementById('input');

if (input.value.trim()) {

ws.send(`ОЗ введено в експлуатацію: ${input.value.trim()}`);

input.value = '';

}

}

</script>

</body>

</html>

**Результати тестування:**

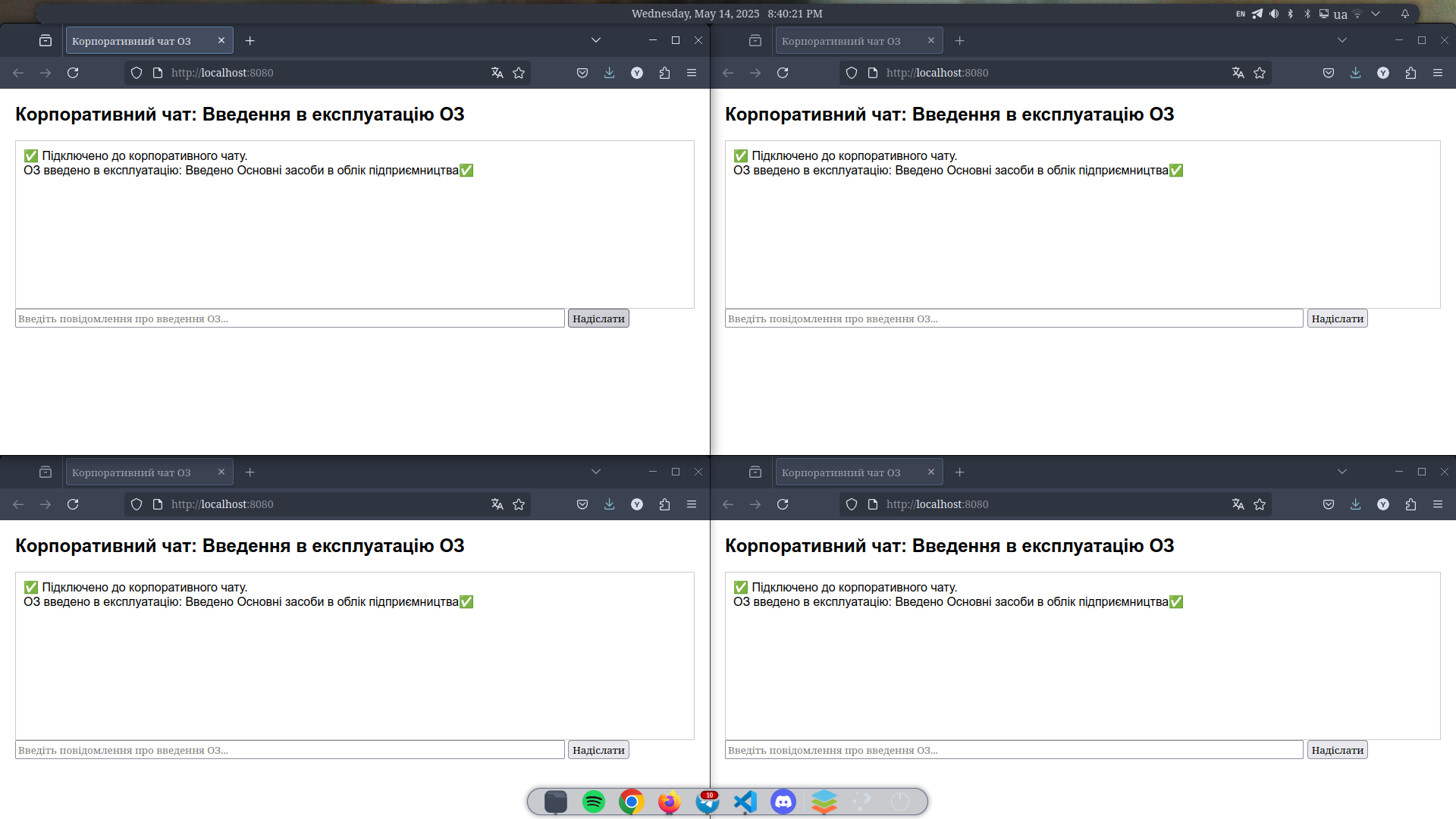
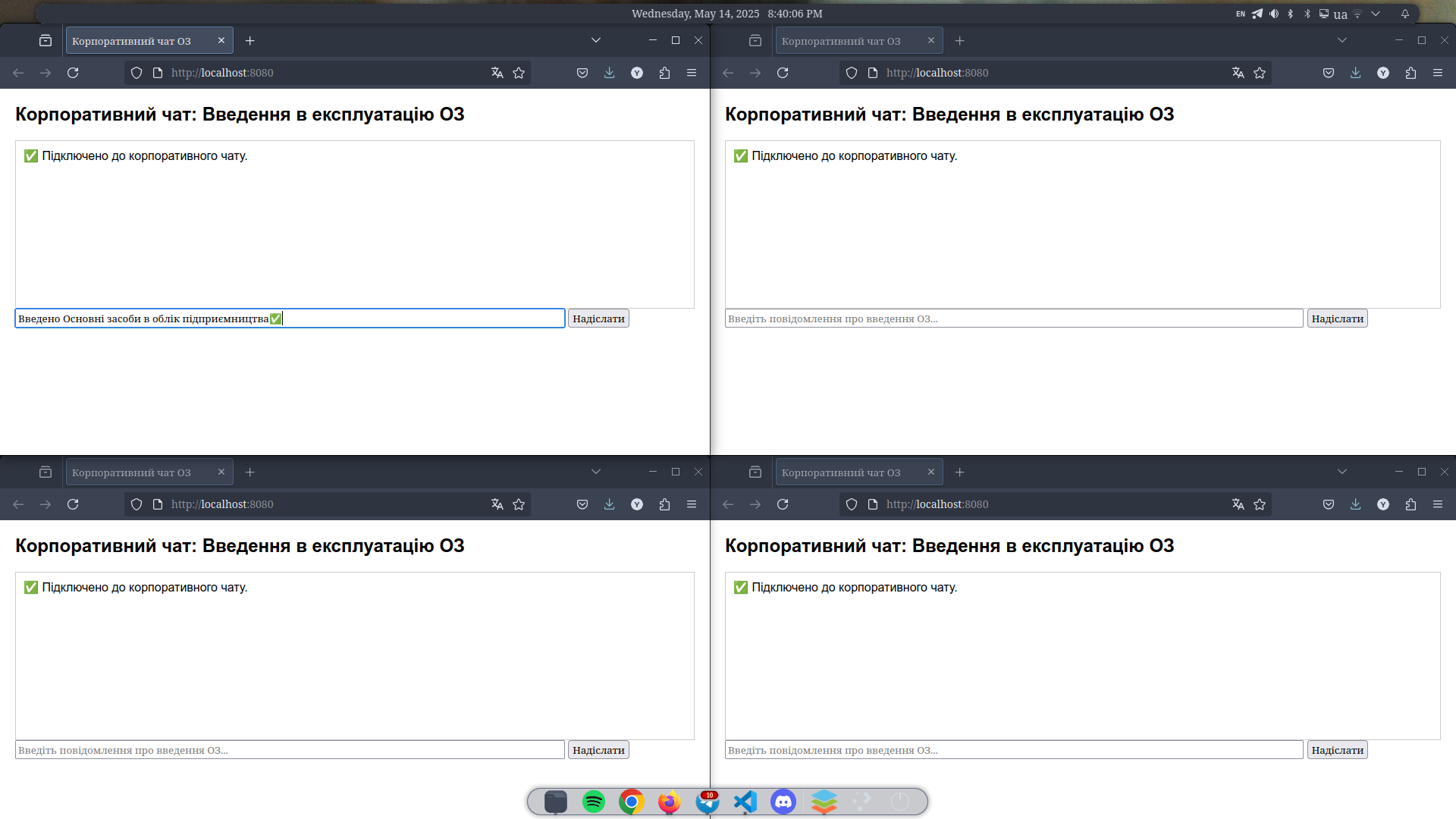
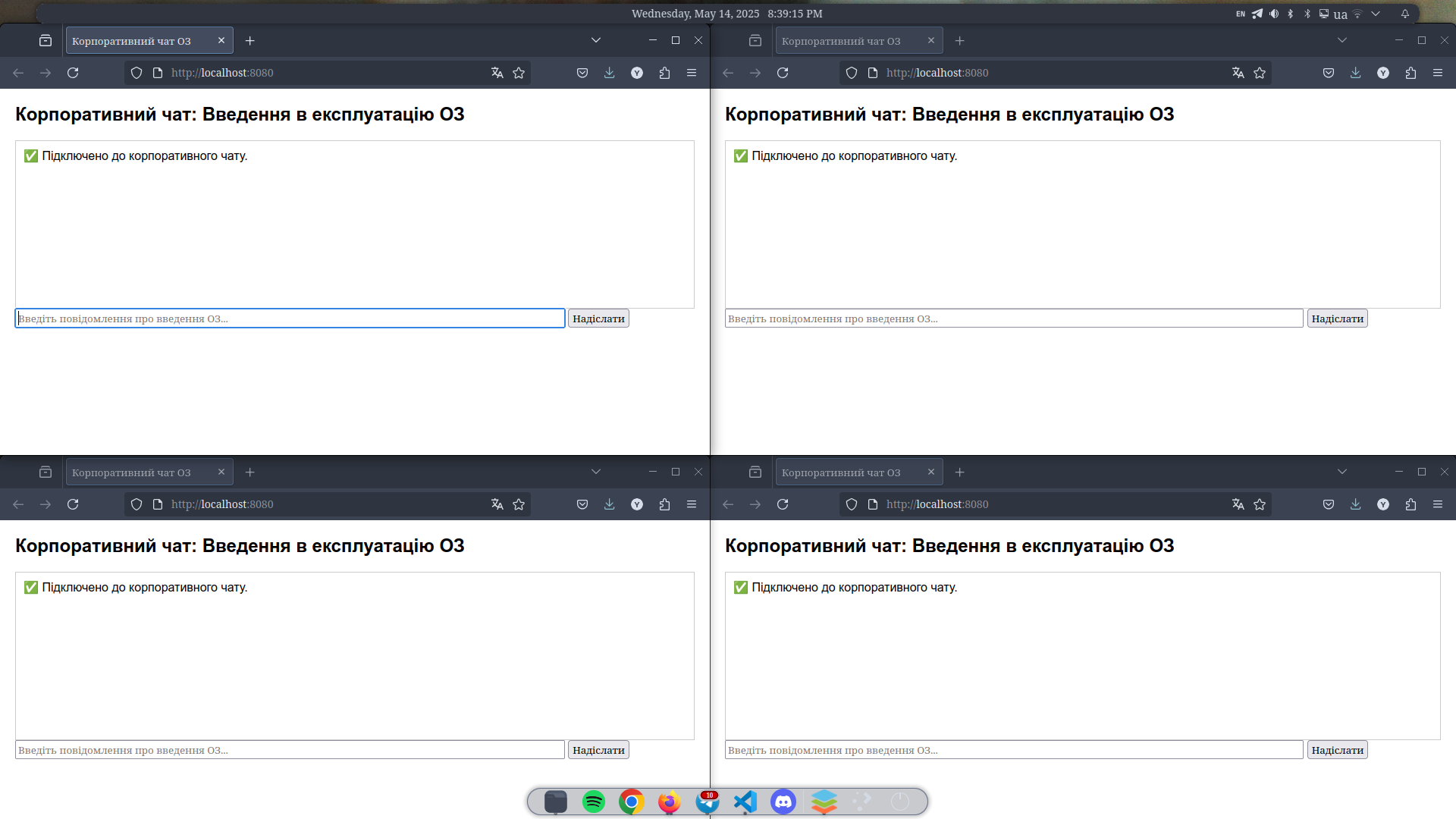
Відкрито одночасно 4 вкладки браузера.

В одній із вкладок було введено повідомлення:

*Введено Основні засоби в облік підприємництва✅*

Це повідомлення з'явилося в усіх вкладках — тобто всі активні користувачі отримали повідомлення.

Таким чином, подієву модель реалізовано коректно, протокол WebSocket забезпечує миттєву доставку.



**Висновок:**

У ході виконання лабораторної роботи було реалізовано просту подієву архітектуру на основі WebSocket-протоколу. Було створено сервер на Node.js, що підтримує WebSocket-з’єднання, та веб-інтерфейс, через який користувачі можуть обмінюватися повідомленнями в реальному часі. Цей підхід дозволяє ефективно реалізувати патерн управління подіями для задач корпоративної взаємодії.