# Explicación de WebSocket y Ejemplo en Node.js + React

## ¿Qué es WebSocket?

WebSocket es un protocolo de comunicación que permite establecer una conexión persistente y bidireccional entre el cliente (navegador) y el servidor. A diferencia de HTTP, donde el cliente debe realizar peticiones cada vez que requiere información, con WebSocket ambos lados pueden enviarse datos en tiempo real sin necesidad de abrir múltiples conexiones.

## Características principales de WebSocket

• Comunicación bidireccional (cliente ⇄ servidor).

• Conexión persistente, no se abre/cierra en cada mensaje.

• Ideal para aplicaciones en tiempo real como chats, videojuegos, notificaciones.

## Ejemplo práctico con Node.js y React

Para comprender mejor el funcionamiento, se construyó un ejemplo de chat en tiempo real. Este proyecto está compuesto por un servidor en Node.js que utiliza la librería 'ws' para manejar conexiones WebSocket, y un cliente en React con TypeScript y TailwindCSS que permite enviar y recibir mensajes.

### Backend (server.js)

En el backend se utiliza Node.js con la librería 'ws'. El servidor escucha en el puerto 8080 y cada vez que recibe un mensaje de un cliente, lo reenvía a todos los demás clientes conectados.  
  
Código resumido:  
  
const WebSocket = require('ws');  
const server = new WebSocket.Server({ port: 8080 });  
  
server.on('connection', (socket) => {  
 socket.on('message', (msg) => {  
 server.clients.forEach((cliente) => {  
 if (cliente.readyState === WebSocket.OPEN) {  
 cliente.send(msg.toString());  
 }  
 });  
 });  
});

### Frontend (React + TypeScript)

En el frontend se desarrolló un pequeño componente con React y TypeScript. Este abre una conexión WebSocket con el servidor en 'ws://localhost:8080' y mantiene un estado con los mensajes recibidos. Cada vez que el usuario escribe un mensaje y lo envía, este se transmite al servidor, el cual lo reenvía a todos los clientes conectados.  
  
La interfaz fue construida con TailwindCSS para darle un diseño sencillo y funcional.

## Flujo de comunicación

1. El servidor WebSocket se inicia en Node.js.  
2. Un cliente en React abre la conexión.  
3. El usuario escribe un mensaje y lo envía.  
4. El servidor recibe el mensaje y lo reenvía a todos los clientes.  
5. Cada cliente actualiza su lista de mensajes en tiempo real.

## Conclusión

Este ejemplo demuestra cómo utilizar WebSocket para implementar comunicación en tiempo real entre múltiples clientes y un servidor. La integración con React permite mostrar un chat básico, pero la misma lógica puede escalarse para aplicaciones más complejas como videojuegos, paneles de monitoreo o sistemas de notificación instantánea.