

## PROCESO DE APRENDIZAJE IV (PROGRAMACION WEB II) 2025-10

APELLIDOS Y NOMBRES DEL ESTUDIANTE:	CORREO ELECTRÓNICO:
Torres Cerdan, Andrea Melissa	72933838@mail.isil.pe
Huaman Solis, Gean	72789992@mail.isil.pe

Deberás leer detenidamente cada una de las indicaciones de la evaluación con la finalidad de cumplir con todos los puntos solicitados.

### CONSIDERACIONES GENERALES PARA EL DESARROLLO DEL TRABAJO / PROYECTO:

- Esta es una actividad individual.
- Si tuvieras consultas con respecto a lo solicitado en uno o varios puntos, deberás comunicarte oportunamente con tu docente para que la inquietud sea aclarada en un plazo prudente y puedas cumplir con los plazos de entrega de la actividad.
- Culminada la evaluación, deberás subir el archivo guardándolo con el nombre ep1 nombre y apellido.
- Es responsabilidad exclusiva del estudiante subir adecuadamente el documento solicitado corroborando que sea el correcto y que se haya cargado sin errores a la plataforma ISIL+.
- NO SE REVISARÁN LAS EVALUACIONES ENTREGADAS FUERA DEL PLAZO ESTABLECIDO.

### CONSIDERACIONES DEL ENTREGABLE

- Se le agradece que genere su propio código, no haga uso de nuestros códigos realizados en clases, no utilice plantillas. Procure hacer uso de su imaginación, ingenio, creatividad y genere su propio producto.
- Debe enviar la carpeta comprimida del trabajo desarrollado, tenga en cuenta que la carpeta enviada contenga todo lo que se requiere para que proyecto funcione adecuadamente.
- Tenga presente que el nombre de la carpeta a enviar por la plataforma [isilmas](#) debe estar identificada de la siguiente forma: pa3- con su nombre, apellido. *Ejemplo: pa4-daniel-fuentes*

### 1. FINALIDAD / OBJETIVO DE LA ACTIVIDAD

La meta de esta evaluación es que demuestren sus conocimientos y habilidades en el desarrollo backend, específicamente con **Tecnologías en Tiempo Real (WebSockets/Socket.IO)**: Se busca que investigue de forma autónoma y aplique soluciones para la comunicación bidireccional instantánea.

- **Uso Ético de la Inteligencia Artificial:** Integrar y documentar el uso de herramientas de IA en su proceso de desarrollo, demostrando una comprensión profunda del código generado.

El objetivo final es que pueda construir aplicaciones robustas, escalables y con capacidades en tiempo real, esenciales en el desarrollo web moderno.

## Investigación y Aplicación de WebSockets/Socket.IO Descripción

Esta sección te reta a investigar de forma autónoma sobre **WebSockets o Socket.IO** y aplicar los conocimientos adquiridos para crear una funcionalidad básica en tiempo real.

### Tareas a Realizar

#### 1. Investigación Teórica:

- o Investigue sobre **WebSockets o Socket.IO** (pueden elegir uno o hacer una breve comparación).
- o Expliquen con sus propias palabras qué problema resuelven estas tecnologías y **cómo funcionan** a un nivel conceptual.
- o Mencione al menos **dos casos de uso reales** donde WebSockets/Socket.IO sean fundamentales.

#### 2. Aplicación Práctica Sencilla:

- o Implementen una funcionalidad muy básica utilizando WebSockets o Socket.IO con Node.js.
- o **Ejemplos sugeridos (elija uno):**
  - Un chat simple de una sola sala (mensajes que se envían y reciben en tiempo real por todos los conectados).
  - Un contador que se actualiza en tiempo real en todos los clientes conectados.
  - Un botón que, al ser presionado por un cliente, cambia el estado de algo visible para todos los demás clientes (ej. el color de un div).
  - Siéntase libre de proponer cualquier otra idea que le parezca útil.
- o La implementación debe ser lo más sencilla posible, enfocándose en demostrar que comprende el concepto de **comunicación bidireccional en tiempo real**.

#### 3. Documentación del Proceso con IA:

- o Si utilizan algún modelo de **Inteligencia Artificial** (ej. ChatGPT, Gemini, Copilot, DeepSeek, Qwen, Claude, etc.) para su investigación o para la generación de fragmentos de código, deberá documentar claramente cómo lo realizó..
- o **Para cada uso de IA, el documento Word debe incluir:**
  - **Modelo de IA Utilizado:** Nombre específico del modelo (ej. Gemini 1.5 Pro, ChatGPT-4).
  - **Prompt Creado:** El texto exacto del prompt que utilizó y las diferentes interacciones que efectúa con el modelo.
  - **Resultado Obtenido:** El texto o código generado por la IA.
  - **Explicación línea por línea:** Explicar *cada una de las líneas* del ejemplo de código (o de un fragmento significativo del resultado) que la IA le proporcionó y cómo lo integró en su proyecto. Demuestre su comprensión del código, no solo lo copie y pegue.

## Entregables

- El **código fuente completo** de la aplicación Node.js con la funcionalidad de WebSockets/Socket.IO.
- Un **documento Word** (o PDF si no pueden adjuntar .docx) con la investigación teórica y la documentación detallada del uso de IA (Modelo, Prompt, Resultado, Explicación línea por línea, así como las pantallas de la ejecución del código).
- **Instrucciones claras para ejecutar el proyecto.**

## Recomendaciones

- Comience temprano para tener tiempo suficiente para investigar y experimentar.
- No dude en preguntar si tiene dudas sobre algún punto de la evaluación.
- La creatividad en la elección del tema es bienvenida, siempre y cuando cumplan con los requisitos técnicos.

**¡El mejor de los éxitos!**

RESOLUCIÓN

1. Investigación Teórica:

1.1 Cuadro Comparativo: WebSockets vs Socket.IO

Característica	WebSockets	Socket.IO
<i>Tipo</i>	Protocolo nativo (estándar)	Biblioteca basada en WebSockets
<i>Bidireccional</i>	Sí	Sí
<i>Compatibilidad</i>	Solo navegadores que soporten WebSocket	Compatible incluso con navegadores antiguos
<i>Reconexión automática</i>	No	Sí
<i>Multiplexación de canales</i>	No (debe implementarse manualmente)	Sí (usando “rooms” y “namespaces”)
<i>Soporte de eventos</i>	No (solo mensajes de texto o binarios)	Sí (permite definir eventos personalizados fácilmente)
<i>Manejo de caídas/red lenta</i>	Limitado	Mejorado (incluye reconexión, timeouts, etc.)
<i>Instalación</i>	Nativo en navegadores	Requiere instalación (npm install socket.io)
<i>Facilidad de uso</i>	Requiere más configuración manual	Más amigable y estructurado para desarrolladores

## 1.2 ¿Qué problema resuelven WebSockets y Socket.IO?

- **Problema que resuelven :**
  - Las aplicaciones web tradicionales utilizan el protocolo HTTP, que es de tipo "petición-respuesta": el cliente (navegador) debe solicitar datos al servidor para obtener una respuesta. Esto es ineficiente para aplicaciones que requieren comunicación en tiempo real, como chats, juegos o monitoreo en vivo.
- **Solución :**
  - **WebSockets y Socket.IO** permiten una comunicación **bidireccional y en tiempo real** entre el cliente y el servidor. Esto significa que el servidor puede enviar información al cliente en cuanto ocurra un evento, sin que el cliente tenga que estar pidiendo actualizaciones constantemente.

### Funcionamiento :

- **WebSocket** abre una conexión persistente entre cliente y servidor, que se mantiene activa mientras ambas partes lo deseen.
- **Socket.IO** es una biblioteca que **usa WebSockets** (cuando están disponibles) pero agrega funcionalidades extra como reconexiones automáticas, manejo de eventos, y compatibilidad con navegadores antiguos usando otras técnicas (como polling largo).

## 1.3 Casos de uso reales de WebSockets / [Socket.IO](#)

### 1.3.1. Aplicaciones de chat en tiempo real

**Problema :** Las aplicaciones de mensajería requieren que los usuarios vean los mensajes al instante, sin necesidad de refrescar la página.

**Cómo ayudan WebSockets/[Socket.IO](#) :**

Permiten una conexión persistente entre los usuarios y el servidor, de modo que cada vez que un usuario envía un mensaje, este se transmite automáticamente y en tiempo real a todos los usuarios conectados.

### Ejemplo real:

- **Slack:** plataforma de mensajería para equipos que usa WebSockets para entregar mensajes instantáneamente, mantener sincronizadas las conversaciones, y mostrar cuándo los usuarios están escribiendo.

- **WhatsApp Web:** utiliza WebSockets para mantener la sincronización en tiempo real entre el navegador del usuario y el dispositivo móvil, reflejando nuevos mensajes de forma inmediata.

### 1.3.2. Juegos multijugador en línea

**Problema :** En juegos en línea, los movimientos y acciones de los jugadores deben reflejarse al instante para garantizar una experiencia fluida y justa.

**Cómo ayudan WebSockets/[Socket.IO](#) :**

Mantienen una conexión abierta entre todos los jugadores y el servidor. Cada vez que un jugador se mueve o realiza una acción, esa información se transmite inmediatamente a todos los demás.

**Ejemplo real:**

- [Agar.io](#) : juego en línea donde múltiples jugadores controlan células en un mapa. Los movimientos y colisiones deben actualizarse en tiempo real, y WebSockets permite esa fluidez.
- **Minecraft Realms** : aunque no usa Socket.IO directamente, se basa en una arquitectura de comunicación en tiempo real similar, donde cada acción del jugador se sincroniza al instante con el servidor.

### 1.2.3. Sistemas de monitoreo en tiempo real (dashboards)

**Problema :** En sistemas que muestran datos en vivo, como dashboards financieros, de sensores, o de tráfico, es fundamental mostrar la información más reciente sin que el usuario actualice la página.

**Cómo ayudan WebSockets/[Socket.IO](#) :**

El servidor puede "empujar" (push) los nuevos datos al cliente en cuanto estén disponibles.

**Ejemplo real:**

- **TradingView** : plataforma de análisis bursátil que muestra precios de acciones en tiempo real. WebSockets permiten actualizar los gráficos instantáneamente conforme cambia el mercado.
- **Paneles de monitoreo de servidores (como Grafana con plugins)** : muestran métricas como uso de CPU, tráfico de red o logs actualizados al segundo.

## 2. Documentación Oficial PA4

**Modelo de IA Utilizado:** ChatGPT con GPT-4O

### Prompt creado:

“necesito que me ayudes a implementar una funcionalidad básica utilizando WebSockets o Socket.IO con Node.js. para realizar un chat simple de una sola sala (mensajes que se envían y reciben en tiempo real por todos los conectados).”

### Resultado obtenido:

- Estructura del proyecto
- Guía para inicializar el proyecto
- Código server.js (reemplazado por app.js)
- Html
- Guía para ejecutar el proyecto
- Server.js

### Proceso y sustento

Para la realización del proyecto se toma en cuenta lo realizado en clase junto con la ayuda de la IA.

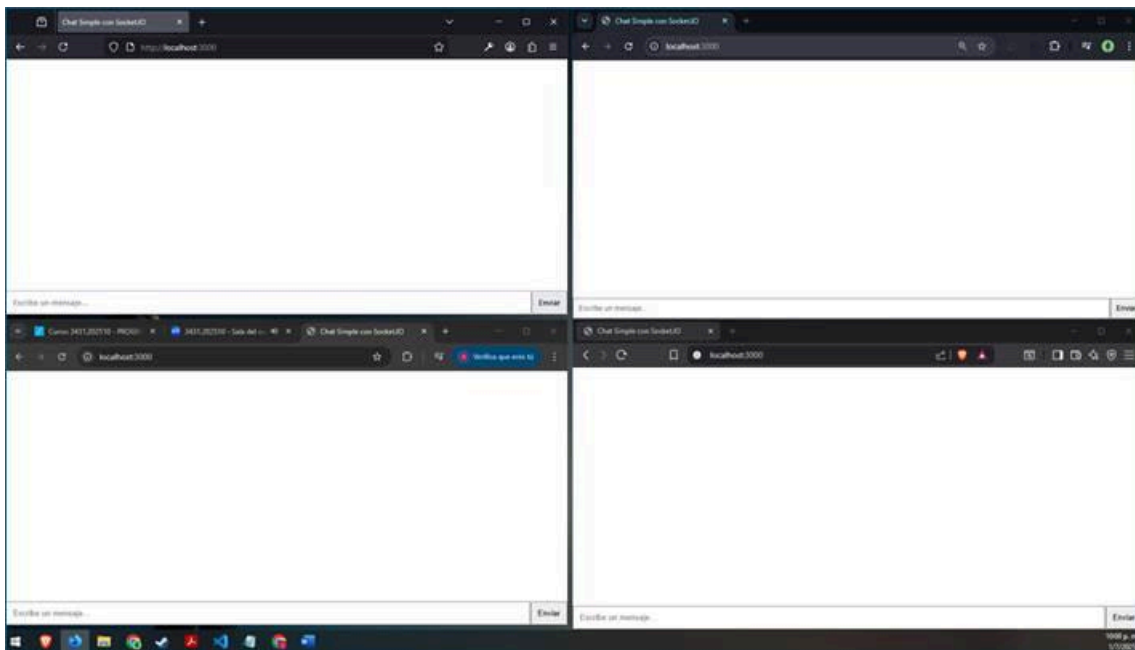
- Se inicializa el proyecto a través de la consola con “npm init -y”.
- Se edita el package.json.
- Se inicializa “npm i express”.
- Se inicializa el socket.io con “npm install express socket.io”
- Se valida y copia el código server.jp y html
  - Se crean las constantes para requerir de express, app, http y socket.io
  - Luego se realiza el llamado a la carpeta public para la lectura de index
  - Se usa una función de socket que se encargará de escuchar las conexiones, haciendo que emita un mensaje en la consola cada que alguien esta en línea o conectado
  - Luego le decimos a socket que escuche los mensajes que otro usuarios podrían enviar y

los muestre

- Se usa la función de socket para la desconexión
  - Se le dice que arraquen el proyecto en el puerto 3000
  - Posteriormente se realiza un html básico usando JS DOM y se anexa el script de socket en la parte inferior
- Luego de revisar las funciones básicas y principales se realizan las pruebas.
  - Se adjunta evidencias y luego se termina con el estilo

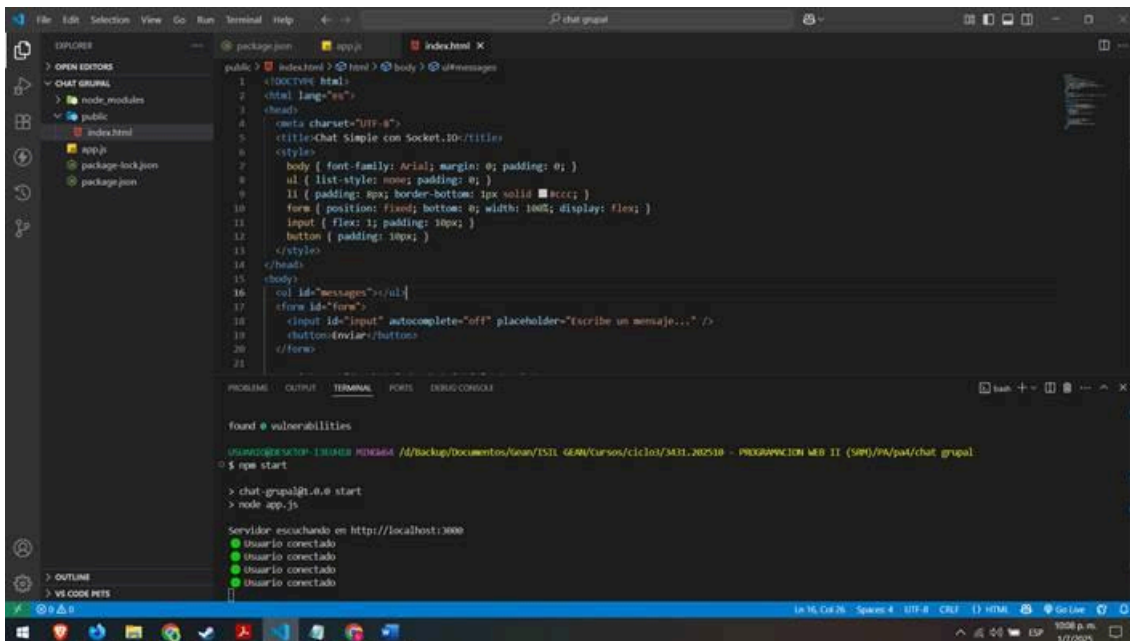
## Evidencias de Pruebas de funcionamiento

**Proyecto:** Chat grupal con socket.io con node.js



Se usan 4 navegadores diferentes





The screenshot shows the VS Code editor with the following files open: `package.json`, `app.js`, and `index.html`. The `index.html` file contains the following code:

```

1 <!DOCTYPE html>
2 <html lang="es">
3 <head>
4   <meta charset="UTF-8">
5   <title>Chat Simple con Socket.io</title>
6   <style>
7     body { font-family: Arial; margin: 0; padding: 0; }
8     ul { list-style: none; padding: 0; }
9     li { padding: 5px; border-bottom: 1px solid #ccc; }
10    form { position: fixed; bottom: 0; width: 100%; display: flex; }
11    input { flex: 1; padding: 10px; }
12    button { padding: 10px; }
13  </style>
14 </head>
15 <body>
16   <ul id="messages"></ul>
17   <form id="form">
18     <input id="input" autocomplete="off" placeholder="Escribe un mensaje..." />
19     <button id="enviar">Enviar</button>
20   </form>
21 </body>

```

The terminal output shows the following commands and results:

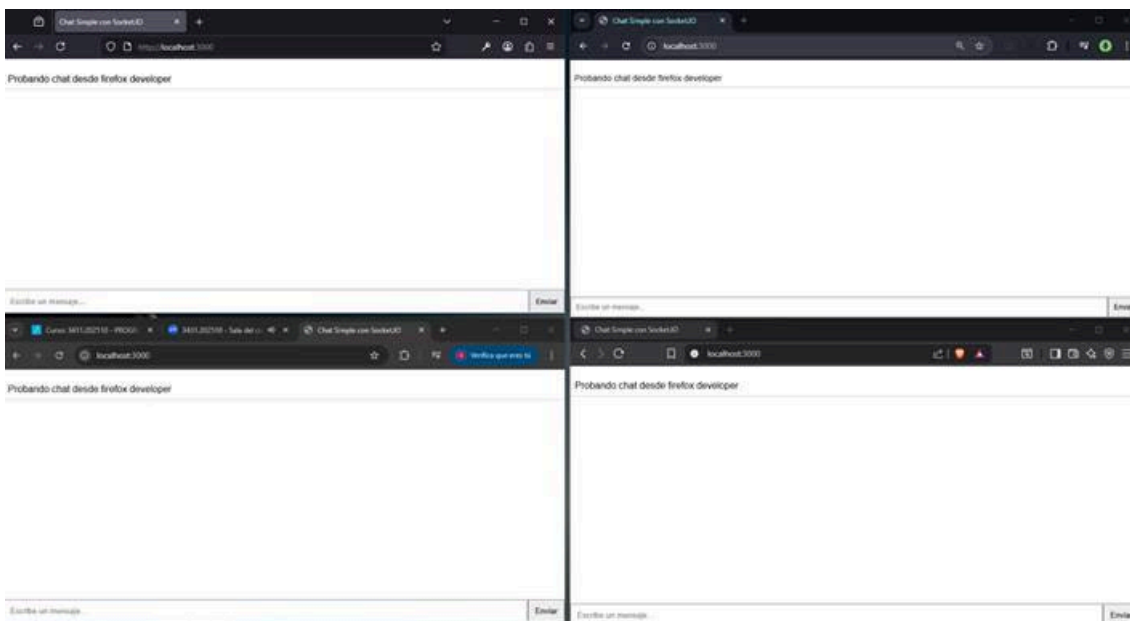
```

found vulnerabilities
$ npm start
> chat-grupal@1.0.0 start
> node app.js

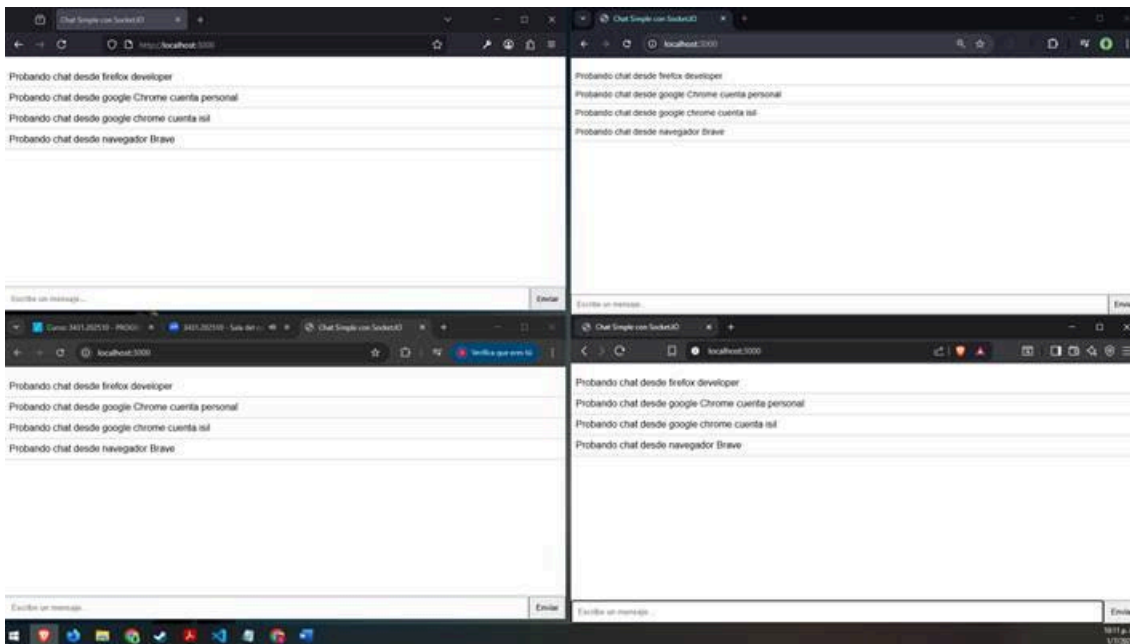
Servidor escuchando en http://localhost:3000
● Usuario conectado
● Usuario conectado
● Usuario conectado
● Usuario conectado

```

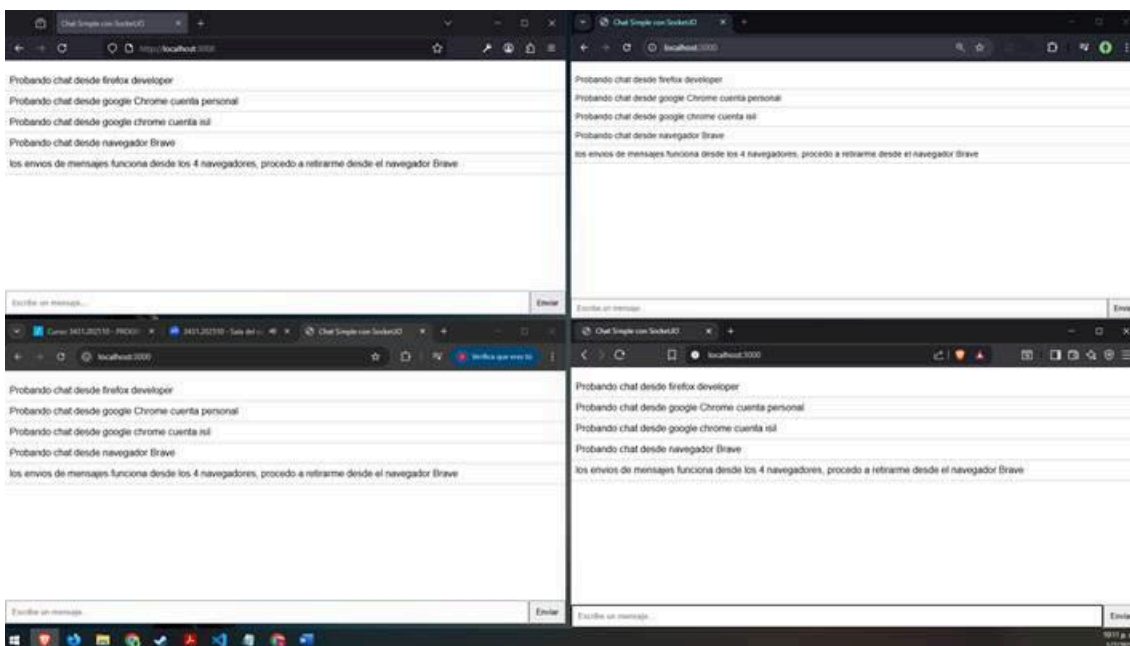
Se verifica en consola 4 usuarios conectados



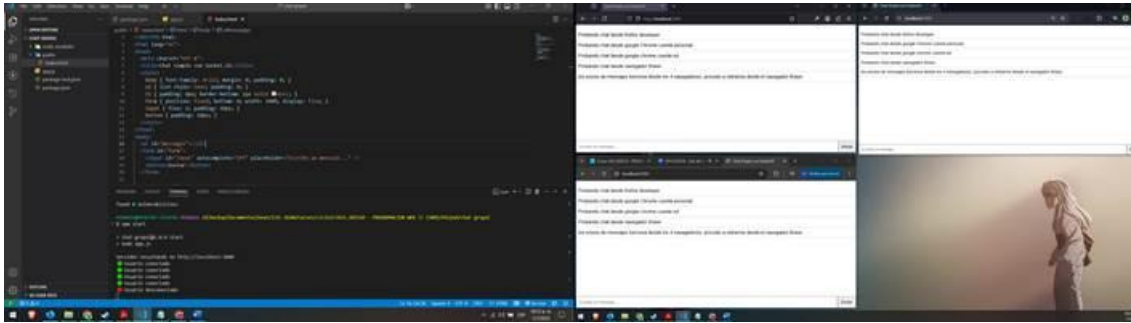
Se envía mensaje desde el navegador 1 y los demás navegadores lo reciben



Se envían mensajes desde los distintos navegadores y todas mantienen el funcionamiento básico.



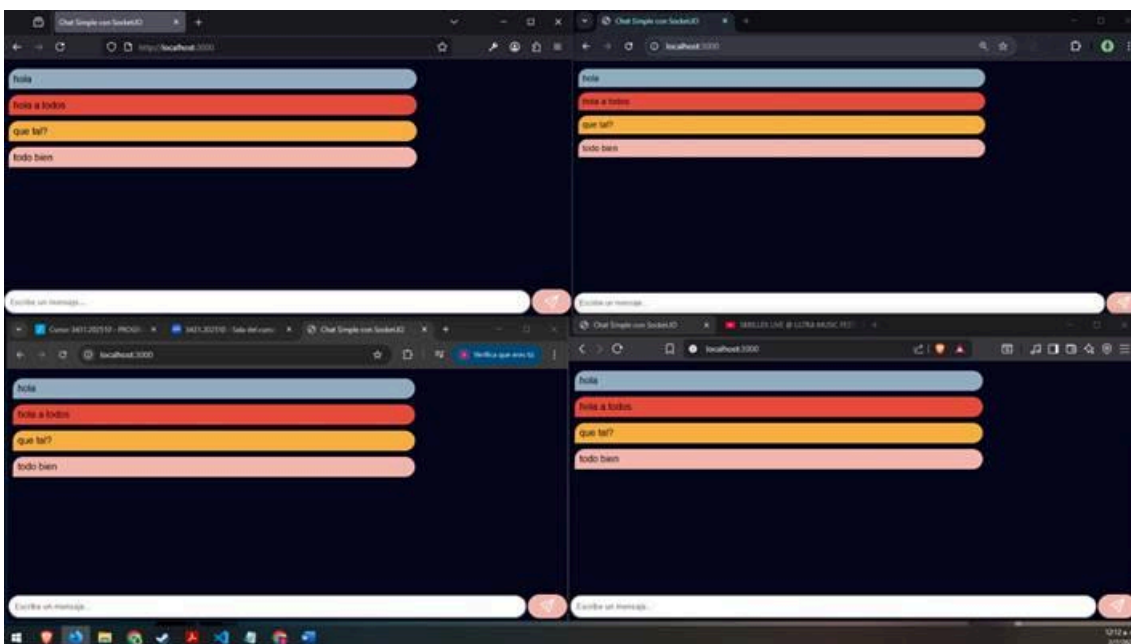
Se simula el cierre de sesión de un navegador (usuario)



Se ve en consola de que un usuario sea desconectado (icono rojo) y la derecha solo se mantiene abierto tres navegadores

Dando así el correcto funcionamiento básico de un chat grupal con implementación de socket.io

Se procedió a añadir estilo dando el siguiente resultado final en la siguiente vista y el código completo en el archivo adjunto [.zip](#)



## CRITERIOS DE EVALUACIÓN

CRITERIOS	PUNTAJE
Organización de las carpetas y archivos	2
Funcionalidad: ¿El código funciona según lo especificado?	5
Claridad del Código: ¿El código está bien organizado, es legible y sigue buenas prácticas (nombres de variables significativos, comentarios cuando sean necesarios)?	2
Comprensión Teórica: ¿Las explicaciones demuestran un entendimiento claro de los conceptos?	3
Documentación: ¿La documentación es completa, precisa y fácil de seguir? ¿El uso de IA está bien documentado y explicado?	5
Esfuerzo de Investigación (WebSockets/Socket.IO): ¿Se nota un esfuerzo genuino por comprender y aplicar la nueva tecnología?	3
<b>TOTAL</b>	<b>20</b>