**Ejercicio 2 SAD (Websockets)**

Trabajo realizado por:

* Sylwia Zoń
* Jorge Andreu Royo

El código mencionado en esta memoria puede ser accedido desde el siguiente repositorio: <https://github.com/joranro1997/SAD/tree/main/Ejercicio%202%20(Websockets)>

En esta memoria se comenta el código desarrollado para esta tarea y las consideraciones tomadas para cada una de las funcionalidades del producto final.

El código ha sido probado bajo versiones de node recientes y hace uso de los módulos de “express”, “socket.io” y “http”.

Se ha hecho uso de npm para la instalación de los módulos necesarios.

Para el primer ejercicio propuesto se ha generado un array que contiene los ids del socket al que se ha conectado cada usuario. Se hace uso de este array para determinar cuando un nuevo usuario que no es uno mismo se ha conectado y enviar a todos los miembros del chat un mensaje informado de que un nuevo usuario se ha conectado.

Al conectarse un usuario, este mismo recibe un mensaje de bienvenida y otro que le informa de los usuarios conectados al chat en ese momento. Además, se envía a su cliente html un array que contiene el nombre de todos los usuarios conectados para mostrarlos en una lista con la que luego podrá interactuar para enviar mensajes privados.

El soporte de “nicknames” o nombres de usuario se ha solucionado a través de una función que genera strings de 5 caracteres aleatorios y los asigna a cada usuario que se conecta. De esta forma cada usuario puede identificarse a si mismo y a los demás, mientras que no puede cambiar a su gusto su nombre de usuario. La función que genera los strings aleatorios es “makeid()”.

Para que los mensajes generados por un usuario no deban pasar del cliente al servidor y de vuelta al cliente, se ha generado la función en el fichero html “sendLocal()”, que es llamada por el cliente siempre que este hace click en el botón de enviar o en “enter”. Esta función comprueba el valor del input (el rectángulo donde se estriben los mensajes) y si hay algo escrito lo añade a la lista de mensajes mostrados en la pantalla. Si no hay nada escrito se añade un mensaje a la pantalla informando al usuario de que debe escribir algo para que se envíe el mensaje. Los mensajes son enviados al servidor para hacer llegar al resto de clientes, pero el servidor no lo envía al cliente del que lo ha recibido.

La funcionalidad que informa de cuando hay un usuario escribiendo se apoya en un número de funciones. En primer lugar, en el cliente se detecta si se ha pulsado alguna tecla y escrito algo sobre el input. De ser así se informa al servidor y se inicia un timeout de 5 segundos, tras los cuales se enviará un mensaje al servidor indicando que ya no se está escribiendo un mensaje. Este timeout se cancela cada vez que se detecta un nuevo pulsado de una tecla sobre el input.

En el lado del servidor, cuando le llega el mensaje de que alguien ha pulsado una tecla se comprueba si la lista de usuarios que están escribiendo está vacía. De ser así se mete en dicha lista (un array) el nombre del usuario que ha pulsado la tecla. A continuación, se envía a todos los demás usuarios un mensaje indicando qué usuario ha pulsado la tecla. Si ya existiera un usuario dentro del array de usuarios que están escribiendo, entonces se añade el nuevo usuario que escribe a dicho array (si no está ya en él) y se envía igualmente un mensaje a todos los demás usuarios para informar de que un nuevo usuario está escribiendo.

Estos mensajes que informan a los clientes de que un usuario está escribiendo lo que hacen es añadir un mensaje a la lista de mensajes mostrados indicando qué usuario está escribiendo. Al contrario, cuando los clientes reciben en mensaje de que un usuario no está escribiendo lo que hacen es retirar este mensaje de la lista de mensajes mostrados.

Cuando expira el timeout de la función que detecta pulsaciones de teclas, se envía un mensaje al servidor y este retira al usuario que ha dejado de teclear del array de usuarios que están escribiendo y envía a todo el resto de los usuarios un mensaje indicando que el usuario ha parado de escribir.

Cuando el cliente recibe un mensaje que no es privado, este también retira el texto que indica que dicho usuario estaba escribiendo.

Como he mencionado antes, se ha creado un elemento html para recoger la lista de usuario conectados. Esta lista se envía a todos los usuarios cuando se conectan y se actualiza con cada nueva conexión o desconexión de un usuario.

En el lado del cliente, cuando se recibe una actualización de la lista, esta se limpia de los datos que ya hubiera en ella y se puebla con cada uno de los elementos presentes en la nueva lista.

Esta lista de usuario conectados es también utilizada par enviar mensajes privados. Cada usuario de la lista tiene un “check box” que puede ticarse. Al envío de un mensaje, se comprueba si alguno de esto “check boxes” está ticado. De ser así, se envía al servidor un mensaje por cada uno de los “check boxes” marcados indicando el contenido del mensaje y el id del que estaba ticado en el “check box”.

El servidor luego envía a dicho usuario el mensaje, siempre que el mensaje privado no vaya dirigido al cliente que lo ha enviado.

En la función que maneja los mensajes privados del lado del servidor también se gestiona en envío del mensaje que informa al resto de usuarios que el usuario ha dejado de teclear, dado que el mensaje ya ha sido enviado.