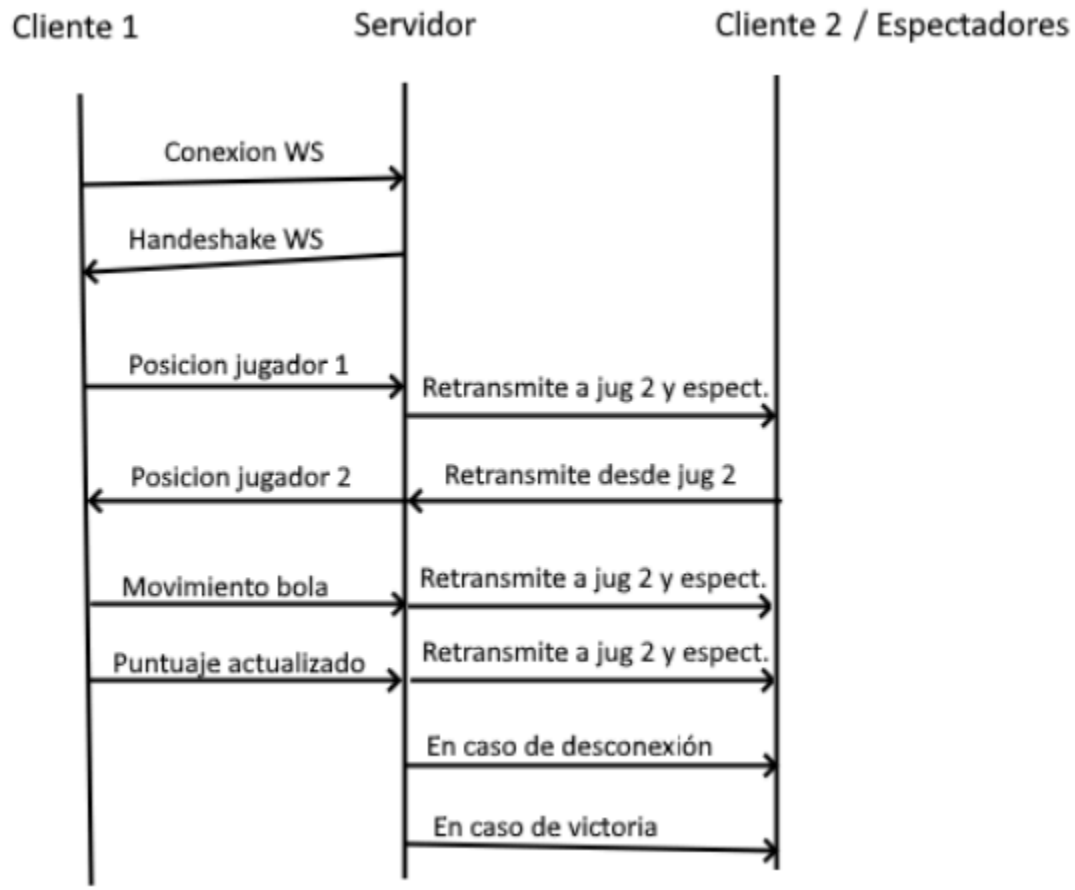
Documentación Multipong

David Reyes

¿Que es WebSocket?

WebSocket es un protocolo que mantiene una conexión entre un navegador y un servidor, parecido a HTTP pero con la diferencia de que WebSocket mantiene abierta la conexión sin cerrarla permitiendo una comunicación bidireccional y asíncrona con baja latencia. Es ideal para chats, videojuegos ya que es en tiempo real.



¿Como funciona el proyecto?

INDEX.JS:

En las líneas del **1 al 4**, importamos los módulos y definimos los puertos.

En las líneas del **6 al 10**, se monta el servidor HTTP y sirve todo lo que haya en el directorio **public.** Se arranca el puerto 7777 y se loguea con un mensaje al iniciarse

En las líneas del **12 al 16,** se crea el servidor de WebSocket sobre el mismo puerto del HTTP, y se crean las variables globales para guardar las conexiones del jugador 1, 2 y espectadores

En las líneas del **18** al **23,** creamos la función **broadcastToSpectators**, que sirve para enviar cualquier objeto **info** como JSON a todos los sockets en **spectators**.

En las líneas del **25 al 34,** se asignan los roles de los jugadores que entran segun el orden en el que llegan.

En las líneas del **36 al 41,** en cuanto ambos jugadores estan conectados, se envía **{game_start: true}** a todos, tanto jugadores como espectadores para que empiece la partida.

En las líneas del **43 al 45,** se crea el manejador de mensajes entrantes y se intenta parsear la cadena JSON

En las líneas del **47 al 55,** se gestiona el envío de mensajes de las posiciones de la pala enviando la **y**.

En las líneas del **57 al 61,** se gestiona el envío de mensajes de la posicion **x** e **y** de la pelota.

En las líneas del **63 al 74,** se gestiona las puntaciones y el final de partida, al recibir **{score1, score2 },** las reenvía a ambos jugadores y espectadores. Si alguno llega a tres puntos, se envía **{game_over, winner}** a los jugadores y **{game_over, spectator_msg}** en amarillo

En las líneas del **76 al 84,** se gestiona la desconexión, se elimina el socket de su variable o del array de espectadores.

INDEX.HTML:

En las líneas del **1 al 10**, construimos a URL de WebSocket usando **location.hostname** (nuestra IP) y declaramos las variables de estados.

En las líneas del **12 al 37,** se gestionan los mensajes de WebSocket según el contenido de **msg.** Según el mensaje hace lo siguiente:

- Guarda el rol del player.
- Arranca la partida y llama a **resetBall().**
- Actualiza la posición de las palas (**player1_y / player2_y**)
- Mueve la bola (**bx / by**).
- Actualiza puntuaciones en pantalla.
- Muestra mensaje de desconexión
- Muestra mensaje de fin de partida (diferente para jugadores y espectadores)

En la línea **39,** se crea la instancia de Phaser con resolución y escenas definidas.

En las líneas del **41 al 50**, **function create,** se inicializa la escena con palas, bola, textos y captura del teclado.

En las líneas del **52 al 59, function resetBall,** centra la bola y le asigna un vector de movimiento con ángulo aleatorio entre -75º y 75º, evitando direcciones casi horizontales.

En las líneas del **61 al 86, function update,** el bucle de juego, el jugador 1 mueve la bola, detecta colisiones con techo/suelo y palas, suma puntos y notifica al servidor con **{score1, score2}** y **{bx, by}.** Cada jugador mueve su pala con flechas y envía **{type:'p1' | 'p2', y}** al servidor.