

**Paso 1: Crear la Carpeta Principal**

1. Abre tu explorador de archivos.
2. Crea una nueva carpeta llamada TIC-TAC-TOE.

**Paso 2: Crear el Archivo index.html**

1. Dentro de la carpeta **proyecto**, crea un archivo nuevo llamado index.html.
2. Copia y pega el siguiente código en el archivo index.html:

index.html

<!DOCTYPE html>

<html lang="en">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Tic Tac Toe</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<div class="content">

<h1>Bienvenido al Juego TIC TAC TOE</h1>

<div>

<label for="roomId">Room ID</label>

<input type="text" id="roomId">

</div>

<div>

<label for="character">Tu personaje</label>

<select id="character">

<option value="X">X</option>

<option value="O">O</option>

</select>

</div>

<button id="startGame">Iniciar Juego</button>

<div id="gameBoard"></div>

<div id="message"></div>

</div>

<script src="scripts.js"></script>

</body>

</html>

### Paso 3: Crear el Archivo styles.css

1. En la misma carpeta TIC-TAC-TOE, crea un archivo nuevo llamado styles.css.
2. Copia y pega el siguiente código en el archivo styles.css:

style.css

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

text-align: center;

background: url('') no-repeat center center fixed;

background-size: cover;

height: 100vh;

display: flex;

justify-content: center;

align-items: center;

}

.content {

background-color: rgba(255, 255, 255, 0.8);

padding: 20px;

border-radius: 10px;

max-width: 400px; /\* Ancho máximo para el contenido \*/

width: 90%; /\* Ancho responsivo \*/

box-sizing: border-box;

box-shadow: 0px 4px 15px rgba(0, 0, 0, 0.2); /\* Sombra para un efecto de elevación \*/

}

h1 {

color: #333;

margin-bottom: 20px;

font-size: 24px;

}

label {

font-weight: bold;

color: #333;

margin-right: 10px;

display: inline-block;

margin-bottom: 10px;

}

input[type="text"], select {

width: calc(100% - 20px); /\* Ancho menos el padding \*/

padding: 10px;

margin-bottom: 20px;

border-radius: 5px;

border: 1px solid #ccc;

font-size: 16px;

}

button#startGame {

width: calc(100% - 20px); /\* Ajusta el botón al mismo tamaño que los inputs \*/

padding: 10px;

background-color: #007bff;

color: white;

border: none;

border-radius: 5px;

cursor: pointer;

font-size: 16px;

transition: background-color 0.3s ease;

}

button#startGame:hover {

background-color: #0056b3;

}

#message {

margin-top: 20px;

font-size: 18px;

color: #555;

}

#gameBoard {

margin-top: 20px;

display: grid;

grid-template-columns: repeat(3, 1fr);

gap: 10px;

}

#gameBoard div {

width: 100px;

height: 100px;

background-color: #f9f9f9;

display: flex;

justify-content: center;

align-items: center;

font-size: 32px;

font-weight: bold;

border-radius: 5px;

border: 1px solid #ccc;

cursor: pointer;

transition: background-color 0.3s ease;

}

#gameBoard div:hover {

background-color: #e9e9e9;

}

### Paso 4: Crear el Archivo scripts.js

1. En la carpeta TIC-TAC-TOE, crea un archivo nuevo llamado scripts.js.
2. Copia y pega el siguiente código en el archivo scripts.js:

scripts.js

const roomIdInput = document.getElementById('roomId');

const characterSelect = document.getElementById('character');

const startGameButton = document.getElementById('startGame');

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

const messageDiv = document.getElementById('message');

let socket;

let currentPlayer;

startGameButton.addEventListener('click', () => {

const roomId = roomIdInput.value;

const character = characterSelect.value;

if (roomId && character) {

connectToServer(roomId, character);

} else {

alert('Por favor, ingrese Room ID y seleccione un personaje.');

}

});

function connectToServer(roomId, character) {

socket = new WebSocket(`ws://localhost:8081/${roomId}`);

socket.onopen = () => {

socket.send(JSON.stringify({ type: 'join', character: character }));

currentPlayer = character;

setupBoard();

};

socket.onmessage = (event) => {

const data = JSON.parse(event.data);

if (data.type === 'move') {

handleMove(data.index, data.character);

} else if (data.type === 'message') {

messageDiv.textContent = data.text;

}

};

socket.onerror = (error) => {

console.error('WebSocket Error:', error);

};

gameBoard.addEventListener('click', (event) => {

if (event.target.classList.contains('cell')) {

const index = event.target.dataset.index;

if (event.target.textContent === '') { // Ensure cell is empty

socket.send(JSON.stringify({ type: 'move', index: index, character: currentPlayer }));

}

}

});

}

function setupBoard() {

gameBoard.innerHTML = '';

for (let i = 0; i < 9; i++) {

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

cell.classList.add('cell');

cell.dataset.index = i;

gameBoard.appendChild(cell);

}

}

function handleMove(index, character) {

const cell = gameBoard.querySelector(`.cell[data-index="${index}"]`);

cell.textContent = character;

cell.style.pointerEvents = 'none'; // Disable cell after move

}

### Paso 5: Crear el Archivo server.js

1. En la carpeta TIC-TAC-TOE, crea un archivo nuevo llamado server.js.
2. Copia y pega el siguiente código en el archivo server.js:

server.js

const WebSocket = require('ws');

const wss = new WebSocket.Server({ port: 8081 });

let rooms = {};

wss.on('connection', (ws, req) => {

// Obtener el ID de la sala de la URL

const roomId = req.url.substring(1);

// Crear una nueva sala si no existe

if (!rooms[roomId]) {

rooms[roomId] = { players: [], board: Array(9).fill(null) };

}

const room = rooms[roomId];

const player = { ws, character: null };

// Añadir el jugador a la sala

room.players.push(player);

// Asignar personajes cuando hay dos jugadores

if (room.players.length === 2) {

const [player1, player2] = room.players;

// Asignar personajes

player1.character = 'X';

player2.character = 'O';

// Enviar mensajes a ambos jugadores con la información de los personajes

player1.ws.send(JSON.stringify({ type: 'message', text: `Tu personaje es X. El otro jugador es O.` }));

player2.ws.send(JSON.stringify({ type: 'message', text: `Tu personaje es O. El otro jugador es X.` }));

}

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

const data = JSON.parse(message);

if (data.type === 'join') {

// Asignar personaje cuando un jugador se une

player.character = data.character;

if (room.players.length === 2) {

room.players.forEach(p => {

if (p.ws !== ws) {

p.ws.send(JSON.stringify({ type: 'message', text: `El otro jugador es ${player.character}` }));

}

});

}

} else if (data.type === 'move') {

// Registrar el movimiento en el tablero

room.board[data.index] = data.character;

// Enviar el movimiento a ambos jugadores

room.players.forEach(p => p.ws.send(JSON.stringify({ type: 'move', index: data.index, character: data.character })));

// Verificar si hay un ganador o un empate

const winner = checkWinner(room.board);

if (winner) {

if (winner === 'Draw') {

room.players.forEach(p => p.ws.send(JSON.stringify({ type: 'message', text: 'El juego ha terminado en empate.' })));

} else {

room.players.forEach(p => {

if (p.character === winner) {

p.ws.send(JSON.stringify({ type: 'message', text: `¡${winner} ha ganado!` }));

} else {

p.ws.send(JSON.stringify({ type: 'message', text: `${winner} ha ganado, ¡tú has perdido!` }));

}

});

}

// Reiniciar el tablero para un nuevo juego

room.board = Array(9).fill(null);

}

}

});

// Limpiar jugadores desconectados

ws.on('close', () => {

room.players = room.players.filter(p => p.ws !== ws);

if (room.players.length === 0) {

delete rooms[roomId]; // Eliminar la sala si está vacía

}

});

});

// Función para verificar si hay un ganador

function checkWinner(board) {

const winningCombinations = [

[0, 1, 2], [3, 4, 5], [6, 7, 8],

[0, 3, 6], [1, 4, 7], [2, 5, 8],

[0, 4, 8], [2, 4, 6]

];

for (const [a, b, c] of winningCombinations) {

if (board[a] && board[a] === board[b] && board[a] === board[c]) {

return board[a];

}

}

if (!board.includes(null)) {

return 'Draw';

}

return null;

}

### Paso 6: Instalar Dependencias

1. Abre una terminal y navega a la carpeta TIC-TAC-TOE.
2. Ejecuta el siguiente comando para inicializar un nuevo proyecto de Node.js:

npm init –y

Instala la dependencia ws para manejar WebSockets:

npm install ws

### Paso 7: Ejecutar el Servidor

1. En la terminal, ejecuta el siguiente comando para iniciar el servidor:

node server.js

### Paso 8: Probar el Juego

1. Abre index.html en tu navegador.
2. Ingresa un Room ID y selecciona un personaje (X o O).
3. Puedes abrir la página en diferentes pestañas o navegadores para simular una partida multijugador.

¡Listo! Ya tienes tu juego de Tic Tac Toe funcionando con Node.js, WebSockets y una interfaz sencilla en HTML y CSS.