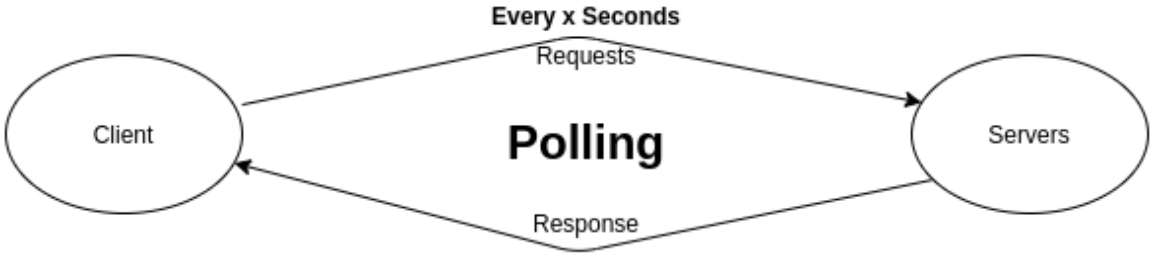
MANUAL BÁSICO: CREACIÓN DE WEBSOCKET CON SOCKET.IO Y EXPRESS

Este documento explica de manera sencilla cómo crear un servidor y un cliente utilizando Socket.IO para la comunicación en tiempo real. El uso típico de un websocket es el de los chats, es decir mensajería instantánea, ejemplo que usaremos.

¿Y por qué no se utiliza la aplicacion `const app = express()` directamente?
Porque esta instancia de Express, que es una capa de middleware/rutas, corre en el servidor http pero no accede directamente a él. El método `app.listen()` devuelve un servidor http, pero lo maneja internamente Express. Y para usar el socket necesitamos acceder al servidor http, por ello necesitamos escuchar el servidor http: `httpServer.listen()`.

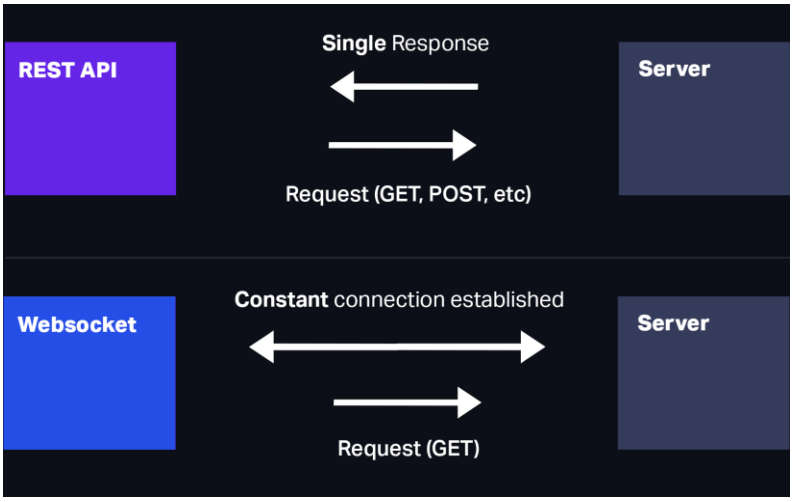
El socket necesita funcionar en el protocolo HTTP en la primera fase, y realizar la comunicación *handshake* inicial, que establece la conexión segura entre dos dispositivos.

La comunicación podría hacerse sin sockets, a través de un método `setInterval()`, pero sería muy ineficiente, ya que no sería una comunicación *full-duplex* (bidireccional), porque haría peticiones al servidor cuando realmente no hubiera nuevos datos que recibir. Es decir, `setInterval()` se comunicaría tanto si hay datos que recibir como si no, produciendo un mayor tráfico en el servidor. Es lo que se conoce como *polling*.



Fuente: Geek for Geeks

¿Y por qué?, porque en cada intercambio de información (en REST API tradicional, y en *polling*) se requiere una petición desde el lado del cliente, mientras que con un websocket la comunicación permanece abierta, sin necesidad de esa petición constante, y el servidor puede enviar directamente los datos sin necesidad de esperar a esa petición del cliente.



Fuente: DZone

Creación del websocket

1. Instalación

Primero instalamos las dependencias:

```
npm install express socket.io
npm install socket.io-client
```

2. Servidor con Express y Socket.IO

Creamos un archivo server.js (en Node.js):

```
import express from "express";
import { createServer } from "http";
```

```
import { Server } from "socket.io";

//Instancia de la aplicación de Express
const app = express();

//Creación del servidor http
const httpServer = createServer(app);

//Creamos el socket que correrá en httpServer
const io = new Server(httpServer, {
  cors: { origin: "*" }
});

const port = 4000;

//Emite por consola un mensaje cada vez que un usuario se conecta
io.on("connection", (socket) => {
  console.log("usuario conectado");

  //Emite por consola un mensaje cada vez que se recibe un mensaje
  socket.on("mensaje", (mensage) => {
    console.log("Mensaje recibido:", mensage);
    io.emit("mensaje", mensage);
  });

  //Emite por consola un mensaje cada vez que un usuario se desconecta
  socket.on("disconnect", () => {
    console.log("Usuario desconectado");
  });
});

//Ponemos a correr el servidor http
httpServer.listen(port, () => {
  console.log(`Servidor corriendo en ${port}`);
});
```

3. Cliente con Socket.IO (vanilla JS)

Ejemplo simple de cliente en client.html:

```
<!DOCTYPE html>
<html>
  <head>
    <title>Cliente Socket.IO</title>
    <script
src="https://cdn.socket.io/4.5.4/socket.io.min.js"></script>
  </head>
  <body>
    <h1>Chat básico</h1>
    <input id="mensaje" placeholder="Escribe un mensaje..." />
    <button onclick="enviar()">Enviar</button>
    <ul id="mensajes"></ul>
```

```
    <script>
      const socket = io("http://localhost:4000");

      socket.on("mensaje", (data) => {
        const li = document.createElement("li");
        li.textContent = data;
        document.getElementById("mensajes").appendChild(li);
      });

      function enviar() {
        const mensage = document.getElementById("mensaje").value;
        socket.emit("mensaje", mensage);
      }
    </script>
  </body>
</html>
```

4. Cliente con Socket.IO (Angular)

Crearemos un servicio donde se encontrará el método que escuche los mensajes, que será de tipo observable. Luego en el componente, nos suscribiremos a este método.

Instalación

```
npm install socket.io-client
```

Servicio websocket.service.ts

```
import { Injectable } from '@angular/core';
import { io, Socket } from 'socket.io-client';
import { Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class WebsocketService {
  private socket: Socket;

  constructor() {
    this.socket = io('http://localhost:4000');
  }

  //Escuchar mensajes
  escucharMensajes(): Observable<string> {
    return new Observable((observer) => {
      this.socket.on('mensaje', (msg: string) => {
        observer.next(msg);
      });
    });
  }
```

```
  //Enviar mensaje
  enviarMensaje(msg: string) {
    this.socket.emit('mensaje', msg);
  }
}
```

Componente chat.component.ts

```
import { Component, OnInit } from '@angular/core';
import { WebsocketService } from '../services/websocket.service';

@Component({
  selector: 'app-chat',
  template: `
    <h1>Chat Angular</h1>
    <input [(ngModel)]="mensaje" placeholder="Escribe un mensaje..." />
    <button (click)="enviar()">Enviar</button>
    <ul>
      <li *ngFor="let msg of mensajes">{{ msg }}</li>
    </ul>
  `
})
export class ChatComponent implements OnInit {
  mensaje = '';
  mensajes: string[] = [];

  constructor(private wsService: WebsocketService) {}

  ngOnInit() {
    this.wsService.escucharMensajes().subscribe((msg: string) => {
      this.mensajes.push(msg);
    });
  }

  enviar() {
    this.wsService.enviarMensaje(this.mensaje);
    this.mensaje = '';
  }
}
```

5. Cliente con Socket.IO (React)

Instalación

```
npm install socket.io-client
```

Componente de react.

```
import React, { useEffect, useState } from "react";
import { io } from "socket.io-client";

const socket = io("http://localhost:4000");

export default function Chat() {
  const [mensaje, setMensaje] = useState("");
  const [mensajes, setMensajes] = useState([]);

  useEffect(() => {
    socket.on("mensaje", (mensaje) => {
      setMensajes((prev) => [...prev, mensaje]);
    });

    return () => socket.off("mensaje");
  }, []);

  const enviar = () => {
    if (mensaje.trim() !== "") {
      socket.emit("mensaje", mensaje);
      setMensaje("");
    }
  };

  return (
    <div>
      <h1>Chat React</h1>
      <input
        value={mensaje}
        onChange={(e) => setMensaje(e.target.value)}
        placeholder="Escribe un mensaje..."
      />
      <button onClick={enviar}>Enviar</button>
      <ul>
        {mensajes.map((mensaje, i) => (
          <li key={i}>{mensaje}</li>
        ))}
      </ul>
    </div>
  );
}
```

Fuentes adicionales:

Herrera, F. (s. f.). React + Socket.IO: Aplicaciones en tiempo real con NodeJS [Curso en línea]. Udemy. https://www.udemy.com/course/react-socket-io-fernando/learn/lecture/22601036