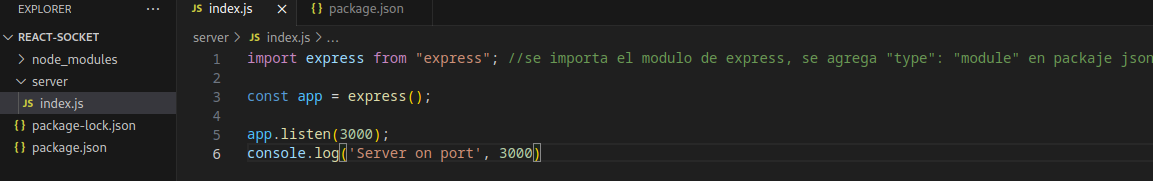
Iniciamos el proyecto con npm init

Instalamos npm i cors express socket.io morgan

En la carpeta server creamos la carpeta index.js

p

Para poder importar el modulo de esta manera se agrega type": "module", s En el archivo packaje.json

En el archivo index.js creamos importamos el servidor de socket, para eso debemos crear un servidor http

importamos http

import http from 'http'

const app = express();

const server = http.createServer(app)

const io = new SocketServer(server);

server.listen(3000);

console.log('Server on port', 3000)

Creamos el front con el modulo vite nmp create vite

Seleccionamos react y javascript

Ingresamos a la carpeta de front que creó y corremos npm install

Con npm run dev dentro de la carpeta del front, ejecutamos el front

Tambien ejecutamos el back con npm run dev desde la carpeta raiz

creamos una escucha del evento io

io.on('connection', socket => {

console.log('client connected')

})

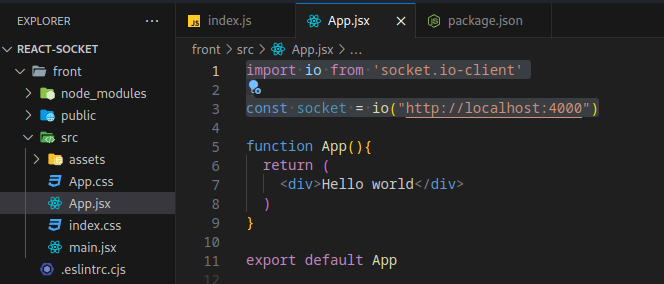
Se crea en el index.js, cuando detecta una conección, crea un socket y escribe el mensaje en consola

En el front instalamos el socket cliente con npm install socket.io-client

Importamos el socket cliente en assets>app.jsx (Cuando carga, crea la conexión con el socket)

import io from 'socket.io-client'

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



Especificamos en app.js el cors para permitir conexión desde el cliente

const io = new SocketServer(server, {

cors: {origin: "http://localhost:5173"}

});

Ahora preparamos el back para recibir mensajes del front

io.on("connection", (socket) => {

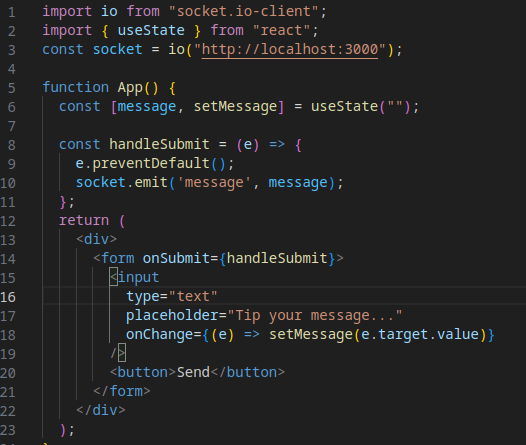
console.log("client connected");

socket.on('message', (data)=>{console.log(data)})

});

En la parte de socket.on(message)) le estamos diciendo que va a recibir un evento llamado message, va a recibir data y luego salen por consola

En el front creamos el evento de onchange, y en el handleSubmit, podemos ver como cuando se da clic al boton, envía el submit, y envía un mensaje al socket socket.emit(‘nombremensaje’, contenidomensaje)



En el back, se recibe el evento de nombre message, lo escribe en consola, y lo devuelve a todos los demas clientes, excepto el cliente que envió (los otros clientes pueden ser otras pestañas del cliente, u otros navegadores/pcs)

socket.on("message", (data) => {

console.log(data);

socket.broadcast.emit("message", data); //al recibir el evento, esta es la respuesta a todos los clientes excp el que envía

});

En el front usamos un usseEffect, cuando escucha un mensaje en un socket lo recibe

useEffect(()=>{

socket.on('message', message=>{

console.log(message);

})

});