

Práctica 4: Node.js

Índice

→ Introducción

→ Parte 1

→ Parte 2 - Explicación

- Servidor.js

- Client.html

→ Funcionamiento y capturas de pantalla

Introducción

Esta práctica ha consistido en 2 partes:

1º → Ha consistido en ejecutar los códigos de ejemplo para ver su funcionamiento y comprobar que funciona.

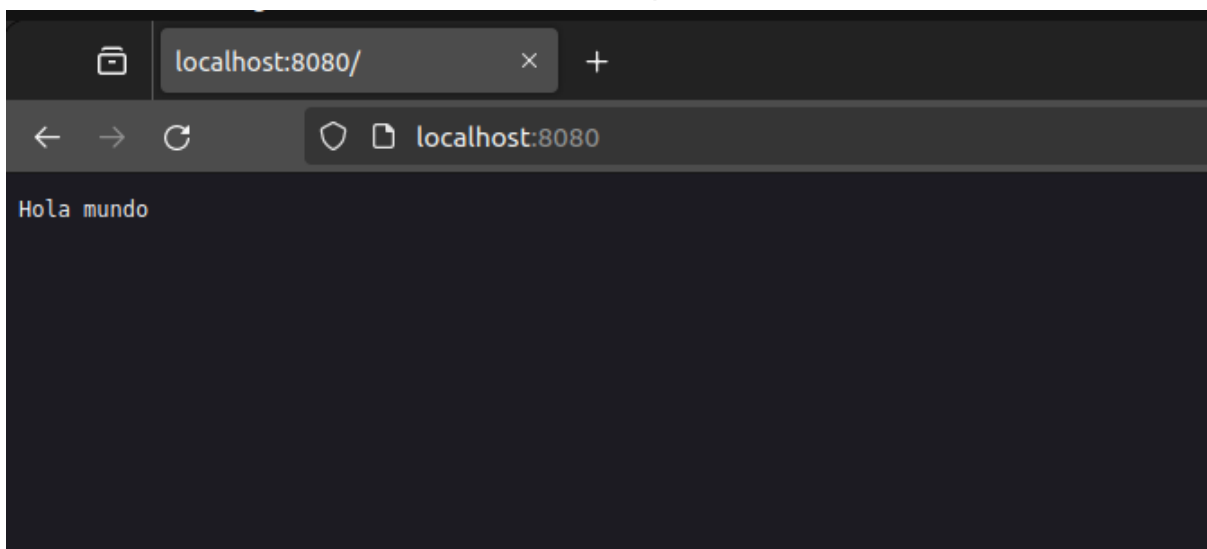
2º → Ha consistido en crear un sistema domótico con sensores y un agente. Los sensores envían la temperatura y la luminosidad percibida al servidor y el agente realiza algunas acciones según los valores obtenidos. El cliente ve en su página todos los eventos ya sea cambio de temperatura/luminosidad, lo que hace el agente. El cliente en su página también puede cambiar el valor de la persiana y del aire acondicionado.

PARTE 1

Ejemplo 1 Helloworld.js

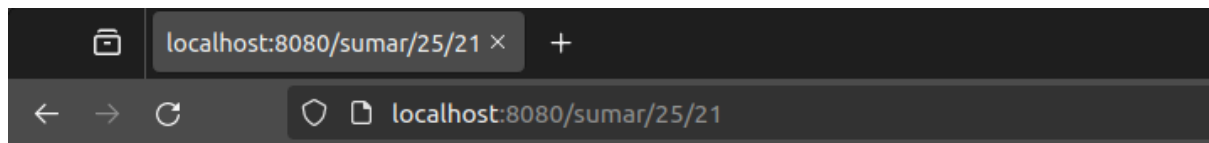
```
helloworld.js > ...  
import http from 'node:http';  
  
http.createServer((request, response) => {  
  console.log(request.headers);  
  response.writeHead(200, {'Content-Type': 'text/plain'});  
  response.write('Hola mundo');  
  response.end();  
})  
  .listen(8080);  
  
console.log('Servicio HTTP iniciado');
```

Un ejemplo básico de un hola mundo, crea el servidor y muestra con por consola los headers de la solicitud HTTP que recibe en el localhost.

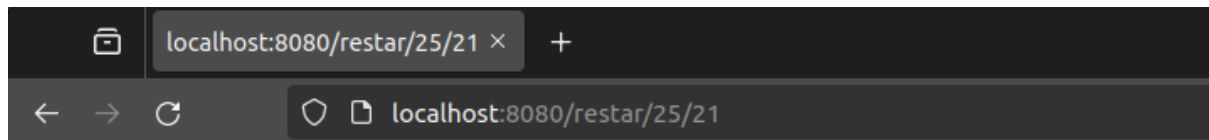


Ejemplo 2 calculadora.js

Este ejemplo lo que hace es recoger del url y lo divide en distintas variables cuando están separadas por / así pudiendo elegir entre sumar, restar, producto, dividir.



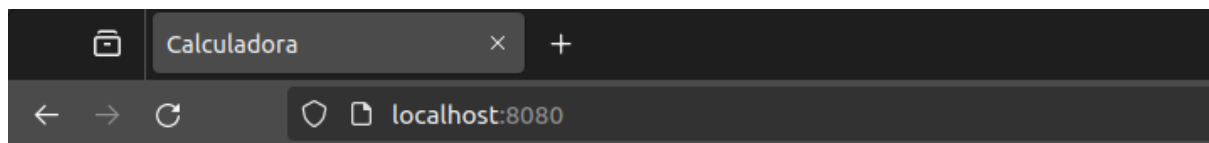
46



4

Ejemplo 3 calculadora-web.js

Este es un ejemplo de calculadora un poco más complejo donde ponemos de página en el localhost un html de una calculadora simple y le mandamos los valores al servidor para que los calcule y luego nos lo muestra.



Valor1: 23

Valor2: 9

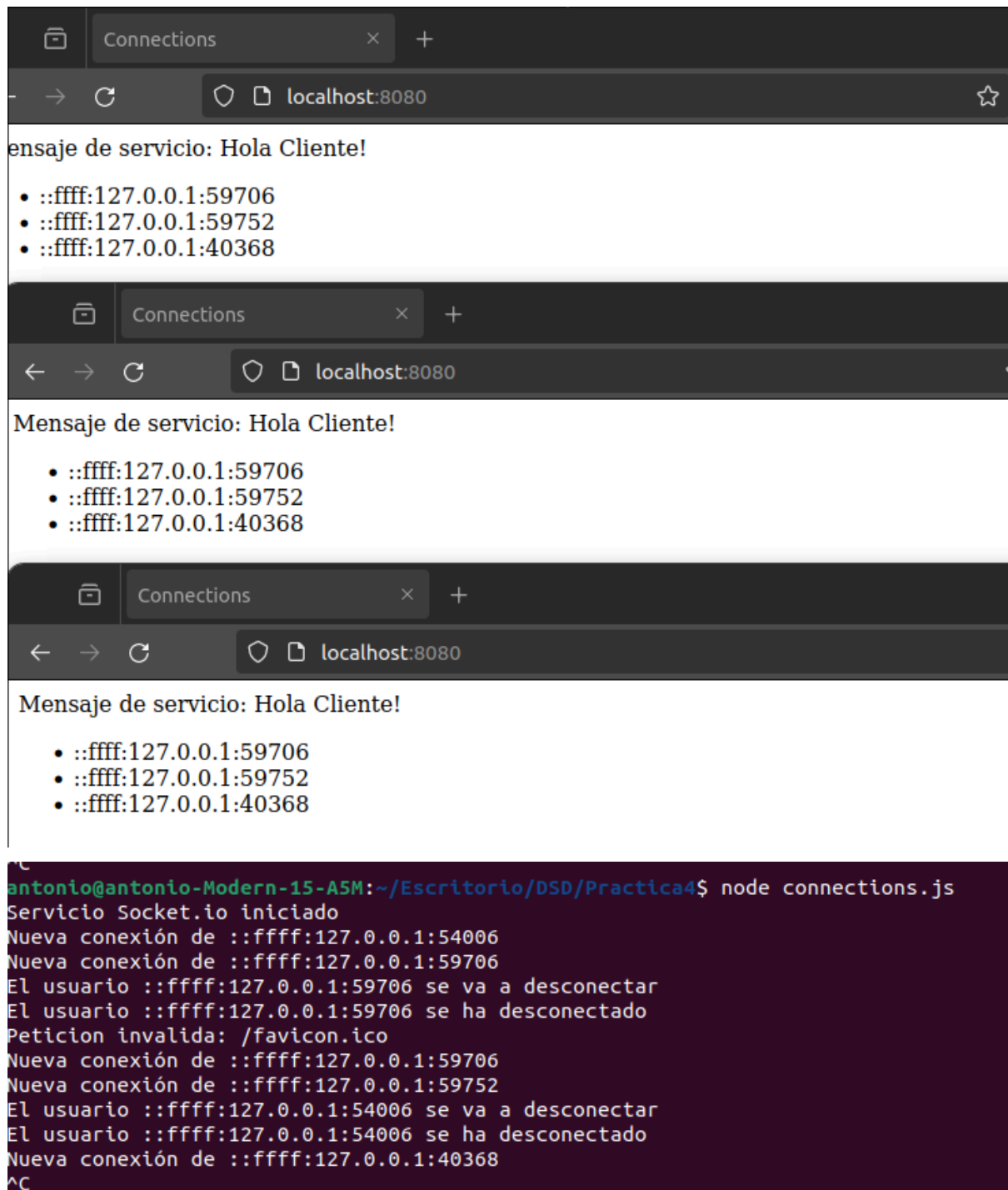
Operación: Producto

Calcular

207

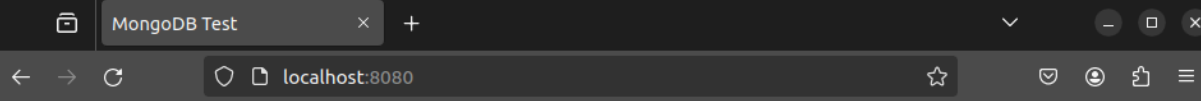
Ejemplo 4 connections.js

Este ejemplo muestra los clientes que están conectados al servidor en la página del servidor, también en consola muestra cuando se conecta un cliente y cuando se desconecta.



Ejemplo 5 mongo-test.js

En este ejemplo nos conectamos a una base de datos y vamos guardando los que se conectan al servidor y lo mostramos por la página del servidor.



```

MongoDB Test
localhost:8080
[
  { "_id": "664779d8f2daff160caf1c49", "host": "::ffff:127.0.0.1", "port": 35438, "time": "2024-05-17T15:38:00.355Z" },
  { "_id": "664779f5f2daff160caf1c4a", "host": "::ffff:127.0.0.1", "port": 48392, "time": "2024-05-17T15:38:29.874Z" },
  { "_id": "664779f6f2daff160caf1c4b", "host": "::ffff:127.0.0.1", "port": 48398, "time": "2024-05-17T15:38:30.890Z" },
  { "_id": "664779f7f2daff160caf1c4c", "host": "::ffff:127.0.0.1", "port": 48398, "time": "2024-05-17T15:38:31.972Z" },
  { "_id": "664779f8f2daff160caf1c4d", "host": "::ffff:127.0.0.1", "port": 48398, "time": "2024-05-17T15:38:32.808Z" },
  { "_id": "664779f9f2daff160caf1c4e", "host": "::ffff:127.0.0.1", "port": 48398, "time": "2024-05-17T15:38:33.442Z" },
  { "_id": "66477a05bcb4ea6fd284aa35", "host": "::ffff:127.0.0.1", "port": 53032, "time": "2024-05-17T15:38:45.530Z" },
  { "_id": "664e4f8d409a5a1b8ea62647", "host": "::ffff:127.0.0.1", "port": 60198, "time": "2024-05-22T20:03:25.398Z" },
  { "_id": "664e4f8f409a5a1b8ea62648", "host": "::ffff:127.0.0.1", "port": 34836, "time": "2024-05-22T20:03:27.897Z" },
  { "_id": "664e4f92409a5a1b8ea62649", "host": "::ffff:127.0.0.1", "port": 34836, "time": "2024-05-22T20:03:30.109Z" },
  { "_id": "664e52f919bbd97b5ad19137", "host": "::ffff:127.0.0.1", "port": 55990, "time": "2024-05-22T20:18:01.386Z" },
  { "_id": "664e52fc19bbd97b5ad19138", "host": "::ffff:127.0.0.1", "port": 55996, "time": "2024-05-22T20:18:04.137Z" },
  { "_id": "664fd3e04f6195a43a05cbc5", "host": "::ffff:127.0.0.1", "port": 50230, "time": "2024-05-23T23:40:16.980Z" },
  { "_id": "664fd3e24f6195a43a05cbc6", "host": "::ffff:127.0.0.1", "port": 50230, "time": "2024-05-23T23:40:18.712Z" },
  { "_id": "664fd3e34f6195a43a05cbc7", "host": "::ffff:127.0.0.1", "port": 50230, "time": "2024-05-23T23:40:19.419Z" },
  { "_id": "664fd3e34f6195a43a05cbc8", "host": "::ffff:127.0.0.1", "port": 50230, "time": "2024-05-23T23:40:19.923Z" },
  { "_id": "6650580ccbfab7f7323932d0", "host": "::ffff:127.0.0.1", "port": 34528, "time": "2024-05-24T09:04:12.821Z" }
]
```

podemos ver que guarda todas las conexiones de los clientes desde el principio.

Parte 2

Servidor.js

```
var temp_maxima = 30;
var lum_maxima = 50;
var temp_actual = 0;
var lum_actual = 0;
var estado_persiana = 'abierta';
var estado_ac = 'apagado';

const httpServer = http
  .createServer((request, response) => {
    let {url} = request;
    if(url == '/') {
      url = '/client.html';
      const filename = join(process.cwd(), url);

      readFile(filename, (err, data) => {
        if(!err) {
          response.writeHead(200, {'Content-Type': 'text/html; charset=utf-8'});
          response.write(data);
        } else {
          response.writeHead(500, {"Content-Type": "text/plain"});
          response.write(`Error en la lectura del fichero: ${url}`);
        }
        response.end();
      });
    }
    else {
      console.log('Petición inválida: ' + url);
      response.writeHead(404, {'Content-Type': 'text/plain'});
      response.write('404 Not Found\n');
      response.end();
    }
  });
```

Como se puede ver tengo los valores necesarios para operar y también los inicializo. También podemos ver como creo el servidor igual que en mongo-test.js y aparte lo pongo el client.html para la pagina del servidor.

```

MongoClient.connect("mongodb://localhost:27017").then((db) => {

  const dbo = db.db("practica4");
  const collection = dbo.collection("datos");

  const io = new Server(httpServer);
  //Funcion para insertar un mensaje con la fecha actual en la base de datos
  function insertarRegistro(mensaje) {
    // Obtener fecha y hora actual
    const fechaActual = new Date().toLocaleString();

    // Insertar documento en la colección de datos
    collection.insertOne({
      mensaje: mensaje,
      fecha: fechaActual
    })
    .then(() => console.log("Registro insertado en la base de datos"))
    .catch((error) => console.error("Error al insertar registro:", error));
  }

  //Funcion para devolver los registros
  function obtenerRegistros() {
    return new Promise((resolve, reject) => {
      collection.find({}).toArray()
        .then((registros) => {
          //console.log("Registros en la colección 'datos':", registros);
          // Crear un nuevo array con el mensaje y la fecha de cada registro
          const registrosFormateados = registros.map(registro => {
            return {
              mensaje: registro.mensaje,
              fecha: registro.fecha
            };
          });
          resolve(registrosFormateados); // Resuelve la promesa con los registros formateados
        })
        .catch((error) => {
          console.error("Error al obtener registros:", error);
          reject(error); // Rechaza la promesa en caso de error
        });
    });
  }
});
}

```

Ahora a continuación me conecto al mongo.client y me conecto a la base de datos practica4 pero tengo una colección distinta llamada datos. tengo dos funciones auxiliares, una de insertarRegistro donde con la fecha actual te mete en la collection antes inicializada, el mensaje que le pasamos con la fecha y lo dice por consola. ObtenerRegistro tiene una promesa para devolver lo que hay en la base de datos, lo formateo para que no me diera problemas y mostrarlo como quiero, da error en ambas funciones si no se puede realizar la acción.

```

io.on('connection', async (client) => {
  // lo siguiente guarda los usuarios cuando se conectan
  const cAddress = client.request.socket.remoteAddress;
  const cPort = client.request.socket.remotePort;
  // Guardo en la base de datos que se ha conectado el usuario X
  insertarRegistro(`El usuario con address: ${cAddress}:${cPort} se ha conectado`);
  const registros = await obtenerRegistros();
  io.emit('tomaregistros', registros);

  // Cuando se desconecta lo registro en la base de datos
  client.on('disconnect', async () => {
    //console.log(`El usuario ${cAddress}:${cPort} se ha desconectado`);
    insertarRegistro(`El usuario con address: ${cAddress}:${cPort} se ha desconectado`);
    const registros = await obtenerRegistros();
    io.emit('tomaregistros', registros);
  });
});

```

empezamos con el connection, cuando se conecta un usuario guardamos su address y su puerto para meter un registro y avisamos a todos los clientes de que hay un nuevo registro, ha de ser await para que espere y no mande registro vacíos como un problema que tuve.

y cuando se desconecta hace lo mismo, lo mete en el registro y lo manda a los clientes.


```

// funcion que se ejecuta cuando se reciben datos nuevos de los sensores
client.on('datos_actualizados', async (data) => {
  console.log("recibo los datos de los sensores");

  temp_actual = data.temperatura;
  lum_actual = data.luminosidad;

  //Meto los cambio de temperatura en los registros
  insertarRegistro('La temperatura ha cambiado a ' + temp_actual);
  insertarRegistro('La luminosidad ha cambiado a ' + lum_actual);
  const registros = await obtenerRegistros();
  io.emit('tomaregistros', registros);

  // Compruebo si el agente cambia algo de los actuadores y en
  // caso positivo se lo notifico al usuario y lo registro
  if (temp_actual > temp_maxima && estado_ac == "apagado"){
    io.emit('encender_ac');
    estado_ac = 'encendido';
    //Meto en el registro y envio a todos los clientes
    insertarRegistro('El agente ha encendido el AC');
    const registros = await obtenerRegistros();
    io.emit('tomaregistros', registros);
  }
  else if(temp_actual <= temp_maxima && estado_ac == "encendido"){
    io.emit('apagar_ac');
    estado_ac = 'apagado';

    //Meto en el registro y envio a todos los clientes
    insertarRegistro('El agente ha apagado el AC');
    const registros = await obtenerRegistros();
    io.emit('tomaregistros', registros);
  }
  if (lum_actual > lum_maxima && estado_persiana == "abierta"){
    io.emit('cerrar_persiana');
    estado_persiana = 'cerrada';
    insertarRegistro('El agente ha cerrado la persiana');
    const registros = await obtenerRegistros();
    io.emit('tomaregistros', registros);
  }
  // Envio para que actualiza la advertencia siempre aunque no se
  // muestra si no sube del umbral
  io.emit('actualizar_advertencia_temp',temp_actual);
  io.emit('actualizar_advertencia_lum',lum_actual);

  // Actualizo las temperaturas de los clientes
  io.emit('actualizar',{
    temperatura: temp_actual,
    luminosidad: lum_actual,
  });
});

```

datos actualizados lo emite cuando se pasa por el form datos de los sensores, los guardamos, creamos registro de cambio de temperatura y emitimos a todos los clientes. Ahora comprobamos como si fuera el agente si

ha de cambiar el estado de los actuadores si es así, insertamos registros y se lo mandamos a los clientes. emitimos un actualizar para que los clientes tengan siempre actualizados los valores.

```
// funcion que abre/cierra la persiana cuando lo solicita el cliente
client.on('cambiar_estado_persiana', async () => {

  if (estado_persiana == 'abierta')
    estado_persiana = 'cerrada';
  else if(estado_persiana == 'cerrada')
    estado_persiana = 'abierta';

  insertarRegistro("Servidor: la persiana ha cambiado de estado");
  const registros = await obtenerRegistros();
  io.emit('tomaregistros', registros);

  io.emit('obtener_estado_persiana', estado_persiana);
});

// funcion que enciende/apaga el aire acondicionado cuando lo solicita el cliente
client.on('cambiar_estado_ac', async () => {

  if (estado_ac == 'encendido')
    estado_ac = 'apagado';
  else
    estado_ac = 'encendido';

  insertarRegistro("Servidor: el aire acondicionado ha cambiado de estado");
  const registros = await obtenerRegistros();
  io.emit('tomaregistros', registros);

  io.emit('obtener_estado_ac', estado_ac);
});
```

Esto se hace cuando el cliente le da él mismo para cambiar los actuadores, se cambia el estado y mete un registro de lo mismo y como siempre lo emite a todos los clientes.

Client.html

```
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
    <title>Client</title>
  </head>
  <body>
    <div id="estados">
      <p> Temperatura: <span id="temperatural"></span></p>
      <p> Luminosidad: <span id="luminosidad1"></span></p>
      <p> Estado de la persiana: <span id="estado_persiana"></span></p>
      <p> Estado del A/C: <span id="estado_ac"></span></p>
      <button id="cambiar_estado_persiana">Cambiar estado de la persiana</button>
      <button id="cambiar_estado_ac">Cambiar estado del A/C</button>
      <p id="advertencia_lum" style="color: red"></p>
      <p id="advertencia_temp" style="color: red"></p>
    </div>
    <form action="javascript:void(0);" onsubmit="enviarSensores();">
      <h1>SENSORES</h1>
      <label for="temperatura">Temperatura:</label>
      <input type="text" id="temperatura"><br><br>
      <label for="luminosidad">Luminosidad:</label>
      <input type="text" id="luminosidad"><br><br>
      <input type="submit" value="Enviar datos de los sensores">
    </form>
    <h1> REGISTROS DE EVENTOS</h1>
    <div id="registros">
    </div>
  </body>
```

una interfaz muy básica para mostrar los parámetros y poder luego cambiar los sensores por un form, al final en el último div se añadirán los registros.

```
<script src="/socket.io/socket.io.js"></script>
<script type="text/javascript">

  const serviceURL = document.URL;
  const socket = io(serviceURL);

  // cuando se actualizan los sensores
  socket.on('actualizar', (data) => {
    //actualizarLista(data);
    document.getElementById('temperatural').innerHTML = data.temperatura;
    document.getElementById('luminosidad1').innerHTML = data.luminosidad;
  });

  // Cuando se actualizan los actuadores por el agente
  socket.on('cerrar_persiana', () => {
    var estado_persiana = document.getElementById('estado_persiana');
    estado_persiana.innerHTML = "cerrada";
  });

  socket.on('encender_ac', () => {
    var estado_persiana = document.getElementById('estado_ac');
    estado_persiana.innerHTML = "encendido";
  });

  socket.on('apagar_ac', () => {
    var estado_persiana = document.getElementById('estado_ac');
    estado_persiana.innerHTML = "apagado";
  });
```

Actualizamos los datos cuando nos lo mandan poniéndolos en el html.
También cambiamos el innerhtml de los actuadores cuando nos mandan que debemos hacerlo.

```
// Cuando pulsan lo botones el cliente manda para cambiar estado
var boton_persiana = document.getElementById('cambiar_estado_persiana');
boton_persiana.onclick = function(){
    socket.emit('cambiar_estado_persiana');
}

var boton_ac = document.getElementById('cambiar_estado_ac');
boton_ac.onclick = function(){
    socket.emit('cambiar_estado_ac');
}

//cuando se cambia recibe el estado de los actuadores
socket.on('obtener_estado_persiana', (data) => {
    document.getElementById('estado_persiana').innerHTML = data;
});

socket.on('obtener_estado_ac', (data) => {
    document.getElementById('estado_ac').innerHTML = data;
})

// cuando le mandan que ha superado el umbral para la advertencia
socket.on('actualizar_advertencia_lum', (data) => {
    if(data > 50){
        document.getElementById('advertencia_lum').innerHTML = "Advertencia: " +
        " La luminosidad ha sobrepasado el valor máximo. Ahora es: " + data;
    }else{
        document.getElementById('advertencia_lum').innerHTML = " ";
    }
});

socket.on('actualizar_advertencia_temp', (data) => {
    if(data > 30){
        document.getElementById('advertencia_temp').innerHTML = "Advertencia: " +
        " La temperatura ha sobrepasado el valor máximo. Ahora es: " + data;
    }else{
        document.getElementById('advertencia_temp').innerHTML = " ";
    }
});
```

Los botones para cambiar el estado, mandan al servidor que queremos cambiar el estado y los .on de obtener es para que el servidor nos mande el estado.

```

socket.on('tomaregistros', (data) => {
  const divRegistros = document.getElementById('registros');
  divRegistros.innerHTML = '';

  data.forEach(registro => {
    const p = document.createElement('p');
    p.innerHTML = `${registro.mensaje} - Fecha: ${registro.fecha}`;
    divRegistros.appendChild(p);
  });
});

```

Cuando hay un registro nuevo este lo pone en el innerhtml del div anteriormente comentado.

```

// Cambio en lo sensores
function enviarSensores() {
  const temperatura = document.getElementById('temperatura').value;
  const luminosidad = document.getElementById('luminosidad').value;

  socket.emit('datos_actualizados', {
    temperatura: temperatura,
    luminosidad: luminosidad,
    time: new Date().toLocaleString()
  });
}

```

Cambio en los sensores, no actualiza instantáneamente el html del cliente ya que quiero que se parezca lo máximo al diagrama de la práctica, es decir, los sensores mandan los valores al servidor y el servidor avisa al usuario del cambio.

Funcionamiento y capturas de pantalla

```
antonio@antonio-Modern-15-A5M:~/Escritorio/DSD/Practica4$ node servidor.js
Servicio MongoDB iniciado
Registro insertado en la base de datos
Registro insertado en la base de datos
Registro insertado en la base de datos
```

iniciamos el servidor y nos conectamos y ya vemos que se hacen registros. Ahora iremos dando a los botones para cambiar el estado también cambiar los sensores y veremos cómo se añaden cuando se conectan y como están sincronizados al hacer io.emit.

El usuario con address: ::ffff:127.0.0.1:47894 se ha conectado - Fecha: 24/5/2024, 12:02:47

El usuario con address: ::ffff:127.0.0.1:37818 se ha conectado - Fecha: 24/5/2024, 12:04:49

El usuario con address: ::ffff:127.0.0.1:47894 se ha conectado - Fecha: 24/5/2024, 12:02:47

El usuario con address: ::ffff:127.0.0.1:37818 se ha conectado - Fecha: 24/5/2024, 12:04:49

ya vemos que en ambos salen que se han conectado dos usuarios.



Cambiamos los valores por encima de los umbrales para que cambien los actuadores , etc y ahora mostraré el registro para ver que solo se han hecho los cambios desde un cliente y ha cambiado en ambos.

El usuario con address: ::ffff:127.0.0.1:47894 se ha conectado - Fecha: 24/5/2024, 12:02:47

El usuario con address: ::ffff:127.0.0.1:37818 se ha conectado - Fecha: 24/5/2024, 12:04:49

La luminosidad ha cambiado a 90 - Fecha: 24/5/2024, 12:05:51

La temperatura ha cambiado a 90 - Fecha: 24/5/2024, 12:05:51

El agente ha encendido el AC - Fecha: 24/5/2024, 12:05:51

El agente ha cerrado la persiana - Fecha: 24/5/2024, 12:05:51

El usuario con address: ::ffff:127.0.0.1:47894 se ha conectado - Fecha: 24/5/2024, 12:02:47

El usuario con address: ::ffff:127.0.0.1:37818 se ha conectado - Fecha: 24/5/2024, 12:04:49

La luminosidad ha cambiado a 90 - Fecha: 24/5/2024, 12:05:51

La temperatura ha cambiado a 90 - Fecha: 24/5/2024, 12:05:51

El agente ha encendido el AC - Fecha: 24/5/2024, 12:05:51

El agente ha cerrado la persiana - Fecha: 24/5/2024, 12:05:51

como vemos lo anterior es la conexión de los usuarios y vemos que solo he rellenado el form en uno, y los agentes han cambiado los actuadores por superar los valores.



Cambiamos el valor temperatura por debajo del umbral y vemos que el agente cambia el A/C.

luminosidad ha cambiado a 90 - Fecha: 24/5/2024, 12:09:28
temperatura ha cambiado a 23 - Fecha: 24/5/2024, 12:09:28
agente ha apagado el AC - Fecha: 24/5/2024, 12:09:28

La luminosidad ha cambiado a 90 - Fecha: 24/5/2024, 12:09:28
La temperatura ha cambiado a 23 - Fecha: 24/5/2024, 12:09:28
El agente ha apagado el AC - Fecha: 24/5/2024, 12:09:28

Ahora probaremos los botones del usuario en un solo cliente.

Temperatura: 23
Luminosidad: 90
Estado de la persiana: abierta
Estado del A/C: encendido

Cambiar estado de la persianaCambiar estado del A/C

Advertencia: La luminosidad ha sobrepasado el valor máximo. Ahora es: 90

SENSORES

Temperatura:
Luminosidad:

Enviar datos de los sensores

Temperatura: 23
Luminosidad: 90
Estado de la persiana: abierta
Estado del A/C: encendido

Cambiar estado de la persianaCambiar estado del A/C

Advertencia: La luminosidad ha sobrepasado el valor máximo. Ahora es: 90

SENSORES

Temperatura:
Luminosidad:

Enviar datos de los sensores

se cambian en ambos y vemos el registro.

Servidor: el aire acondicionado ha cambiado de estado - Fecha: 24/5/2024, 12:10:58
Servidor: la persiana ha cambiado de estado - Fecha: 24/5/2024, 12:10:59

Servidor: el aire acondicionado ha cambiado de estado - Fecha: 24/5/2024, 12:10:58
Servidor: la persiana ha cambiado de estado - Fecha: 24/5/2024, 12:10:59

finalmente vemos como se desconecta y sale en el registro.

Servidor: la persiana ha cambiado de estado - Fecha: 24/5/2024, 12:10:59
El usuario con address: ::ffff:127.0.0.1:37818 se ha desconectado - Fecha: 24/5/2024, 12:20:23