**GRADO EN INGENIERÍA INFORMÁTICA**

DESARROLLO DE SISTEMAS DISTRIBUIDOS


UNIVERSIDAD
DE GRANADA

# P4 : NodeJS

David Serrano Domínguez

27 de mayo de 2024

# Índice

# 1. Ejemplos

En esta primera parte voy a ejecutar los distintos ejemplos que se nos han proporcionado e intentar comprender su funcionamiento para en base a esto poder realizar el ejercicio de una aplicación web de domotica que se nos pide.

## 1.1. Ejemplo 1

El primer ejemplo simplemente es simplemente la ejecución de un servidor web que lo único que hace es mostrar por pantalla un hola mundo, mostrando como crear un servidor web con NodeJS.
Aquí una prueba de ejecución:

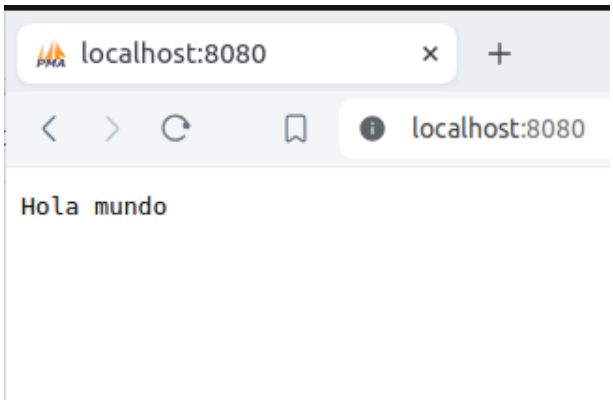


Figura 1: Ejecución Ejemplo 1

## 1.2. Ejemplo 2

El segundo ejemplo se trata de una calculadora la cual hay que pasarle los distintos parámetros por la url siendo primero el tipo de operación, segundo el primer valor de dicha operación y tercero el segundo valor de la misma. Aquí una prueba de ejecución haciendo que sume dos mas dos:

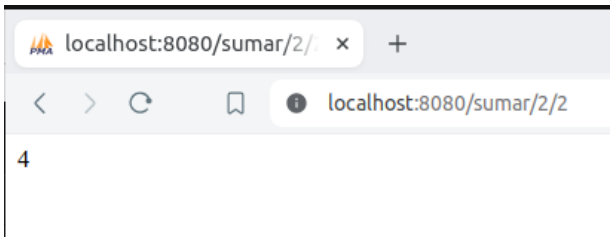


Figura 2: Ejecución Ejemplo 2.

## 1.3. Ejemplo 3

Este tercer ejemplo se trata de una calculadora web, pero esta vez a través de un formulario en el cual tenemos dos input en los que ingresamos los valores y luego un menú desplegable en el que seleccionamos entre las cuatro operaciones básicas.En este caso el servidor será igual que el anterior pero teniendo la posibilidad de sumar pasando los parámetros por la url o mediante el formulario.Para el formulario se añade de código nuevo al servidor la petición GET del archivo raiz y cuando llegue dicha petición entonces se devollvera al cliente el calc.html junto con el status 200 si todo fue bien y en caso contrario con el status 500 el cual es para errores en el servidor, en este caso sería al intentar leer el fichero calc.html

En el caso de que se quiera hacer mediante el formulario, veamos que hace el javascript del calc.html. Lo primero que hará, es que cuando se produzca el evento de submit del botón calcular se recogeran los valores de cada input del formulario y con dichos valores se formara una url que se pasará mediante la llamada asincrona fetch al servidor para que calcule el resultado como si se hubieran pasado mediante la url. Una vez calculado el fetch recoge el resultado y lo mete en en el campo con dicho id. Aquí una prueba de ejecución de dicho ejemplo haciendo que sume seis mas cinco:

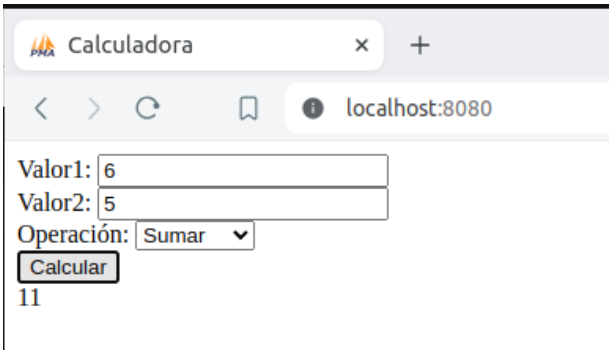


Figura 3: Ejecución Ejemplo 3.

## 1.4. Ejemplo 4

Vamos a pasara a ver el ejemplo cuatro que es diferente a los anteriores ya que introduce el uso del socket.io el cual nos va a permitir que el cliente y servidor se puedan comunicar mediante eventos de manera asincrona, es decir sin tener que recargar la página para cambiar algo en la misma.
En este caso lo que vamos a tener que hacer es una vez creado el servidor HTTP con NodeJS crear un nuevo servidor Socket.io pasandole dicho servidor HTTP y obteniendo el objeto io que nos permitira comunicarnos con el cliente para suscribirnos a eventos del mismo y poder emitirle también.La linea en concreto para hacer esto es la siguiente: **const io = new Server(httpServer);**

Una vez visto esto vamos a ver como se van a gestionar los distintos eventos. Primero vamos a tener el evento connection que va a englobar al resto ya que este evento sucedera cuando un cliente se conecte al servidor haciendole una petición de alguno de sus recursos.Por tanto cuando un cliente se conecte lo que haremos es obtener su dirección IP y su PUERTO mediante el objeto cliente que recibimos cuando este se conecta y una vez hecho esto lo almacenaremos en un array el cual contendra un JSON para cada cliente con su IP y PUERTO y una vez almacenada en dicho array emitiremos a todos los clientes un nuevo array con todos los clientes conectados incluyendo el útimo que se acabe de conectar y esto lo haremos mediante el evento siguiente: **io.sockets.emit('all-connections', allClients);**. Después mandaremos otro evento en el cual saludamos al cliente que se acabe de conectar.
El último evento que tendremos será el evento 'disconnect' el cual sucedera cuando un cliente se desconecte de y lo que hará el servidor cuando esto ocurra será buscar en el array el cliente que se ha desconectado por su ID y PUERTO, y si lo encuentra de manera correcta lo sacamos del dicho array y si no lo encontramos hacemos un

console log avisando de esto. En el lado del cliete simplemente nos vamos a conectar haciendo lo siguiente: **const serviceURL = document.URL; const socket = io(serviceURL);**Y con socket.on nos vamos a suscribir a todos los eventos dicho anteriormente para asu poder mantener los usuarios actualizados cunado el servidor no haga el emit de la lista de usuarios cuando se conecte o desconecte un usuario y avisando po consola cuando nos conectemos o desconectemos del servidor.

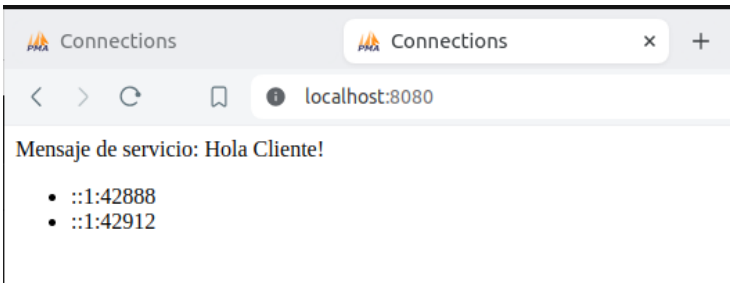


Figura 4: Ejecución Ejemplo 4.

## 1.5. Ejemplo 5

Vamos con el quinto y útimo ejemplo, que nos introduce lo último que vamos a necesitar para realizar la apliación de domótica y es la base de datos de mongodb.
En este caso vamos a replicar un servidor parecido al explicado en el ejemplo anterior pero esta vez cada quez que un cliente se conecte vamos a insertarlo en una collection de una base de datos de mongodb junto con la fecha que dicho cliente se ha conectado.Para mostrarlo por pantalla extraeremos de dicha base de daros todas las filas que contiene y se lo mandaremos al cliente mediante el evento obtener para que este pueda mostrarlo en el html. Aquí un ejemplo de cuando realizamos varias conexiones al navegador desde distintas ventanas, lo que se nos va a mostrar.

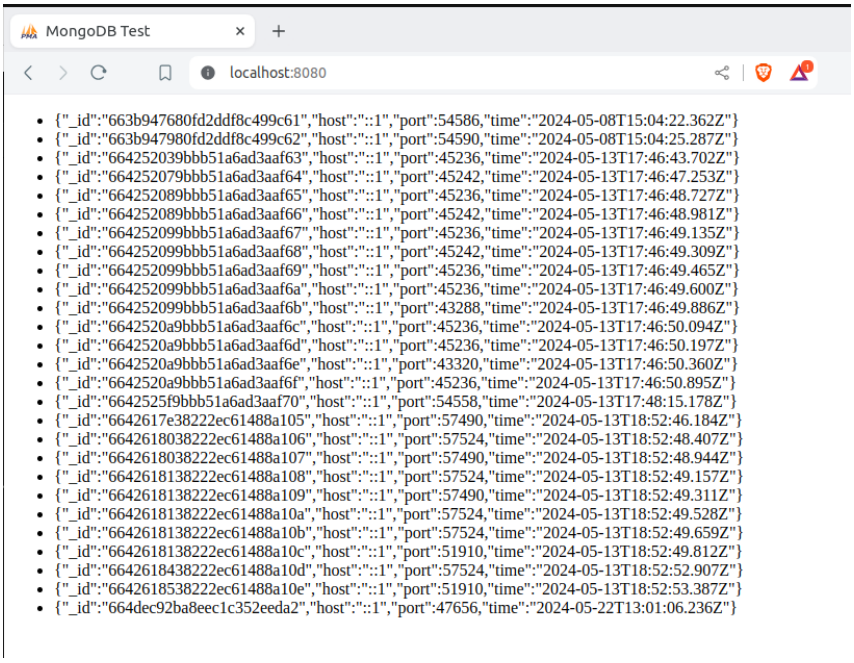


Figura 5: Ejecución Ejemplo 5.

# 2. Ejercicio aplicación web domótica

## 2.1. Esquema de lo que se nos pide

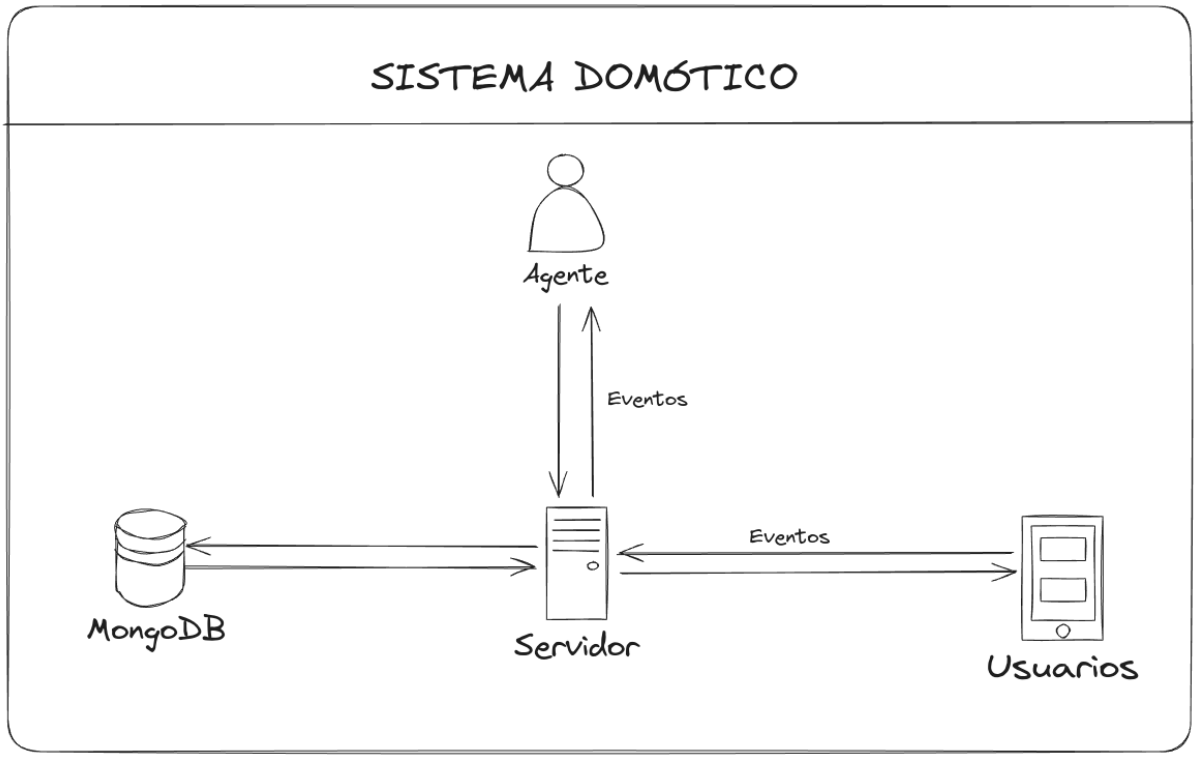


Figura 6: Esquema.

Se nos pide que el usuario pueda acceder a los valores de temperatura de distintos sensores, en nuestro caso son tres: Sensor Temperatura, Sensor Luminosidad y Sensor Dioxido de Carbono. Los cuales son controlados por tres actuadores que son Aire Acondicionado, Persianas y Filtro Dioxido de Carbono respectivamente. Ahora la forma de implementarlo es que los usuarios puedan modificar los distintos sensores mediante peticiones al servidor y que este vaya almacenando a su vez en un base de datos de mongoDB los distintos valores que vamos teniendo y mandandoselos al agente para que este controle los valores y cuando dichos valores alcancen un máximo o un minímo le diga al servidor que actuadores activar o desactivar y si debe avisar a los usuarios cuando se superen dichos máximos.

## 2.2. Método GET del servidor

Primero vamos a ver como nuestro servidor de NodeJS va a procesar los distintos archivos cuando estos se le soliciten

```
const processRequest = (req, res) => {
    let {method, url} = req;
    //console.log(method, url);
    switch(method){
        case 'GET':
            switch(url){
                case '/':
                    url = '/templates/index.html';
```

```
                    leerArchivoHtml(url,res);
                    break;
                case '/luminosidad.html':
                    url='/templates/luminosidad.html';
                    leerArchivoHtml(url,res);
                    break;
                case '/temperatura.html':
                    url='/templates/temperatura.html';
                    leerArchivoHtml(url,res);
                    break;
                case '/dioxido.html':
                    url='/templates/dioxido.html';
                    leerArchivoHtml(url,res);
                    break;
                case '/style.css':
                    url='/css/style.css';
                    leerArchivoCss(url,res);
                    break;
                case '/css/sensores.css':
                    leerArchivoCss(url,res);
                    break;
                case '/js/informacionGeneral.js':
                    leerArchivoJs(url,res);
                    break;
                case '/js/temperatura.js':
                    leerArchivoJs(url,res);
                    break;
                case '/js/funciones.js':
                    leerArchivoJs(url,res);
                    break;
                case '/js/luminosidad.js':
                    leerArchivoJs(url,res);
                    break;
                case '/js/dioxido.js':
                    leerArchivoJs(url,res);
                    break;
                case '/agente.js':
                    leerArchivoJs(url,res);
                    break;
                default://Si la ruta no existe pasamos un error 404
                    res.writeHead(404, {'Content-Type': 'text/html; charset=utf-8'});
                    res.write('<h1>404 Not Found<h1>');
                    res.end();
                    break;
            }
            break;
}
```

Listing 1: Procesamiento de peticiones GET.

Como podemos observar vamos a recibir archivos tanto HTML, como CSS y Javascript y en función del que sea se lo mandaremos a un función para leer archivos diferente y procesarlo según les corresponde.
Por otro lado en caso de que un cliente nos pida un recurso que no tenemos entonces le mandaremos un error 404 avisandole que dicho recurso no existe.

## 2.3. Eventos principales del servidor

En el servidor vamos a tener distintos eventos para cada sensor pero al final van a ser iguales para los tres sensores, por tanto vamos a centrarnos y explicar solo uno, en este caso va a ser el sensor temperatura.
Lo primero es saber que vamos a almacenar en una base de datos de mongoDB llamada sensores en la collection temperatura toda la información que tenga que ver con esta, por tanto cada uno de las filas de esta collection va a contener la temperatura máxima con la que se activa el aire acondicionado, la temperatura minima con la que se desactiva dicho aire, la temperatura actual, el estado del aire acondicionado, es decir, si esta encendido o apagado y por último la fecha y hora en la que se produce una modificación en alguno de los cuatro valores anteriores.

Ahora vamos a ver el evento mediante el cual sacamos de la base de datos los último valores introducidos para mandarselo a todos los usuarios.

```
        collectionTemperatura.find().sort({ $natural: -1 }).limit(1).toArray().then((data) =>
    {
            if (data.length > 0) {
                // Emitir los datos mas recientes de la base de datos (ultimo registro)
                client.emit('obtenerTemperatura', data[0]);
                client.emit('obtenerActuadorTemperatura', data[0]);
                client.emit('temperaturaMaximaAviso', data[0].actual >= data[0].maxima);
            }
        }).catch((error) => {
            console.error("Error obteniendo la temperatura: ", error);
        });
```

Listing 2: Evento obtener temperatura.

Como vemos vamos a obtener de la collection temperatura la última fila y para ello vamos a hacer uso de una promesa y en el caso de que esta se cumpla vamos entonces a comprobar que hay algo en la base de datos, es decir que data contiene algo y en caso de ser así vamos a mandarselo a todos los usuarios para que puedan obtener en cada momoento el ultimo valor insertado en la base de datos, también vamos a emitir el evento obtener actuador para actualizar también el estado del actuador y por último emitimos el evento de si hay temperatura máxima.Este evento básicamente se encarga de actualizar esas tres cosas cada vez que un cliente se conecta a nuestra página.
Para la luminosidad y dioxido de carbono seran eventos similares.

Ahora vamos a ver el evento encargado de insertar la temperatura en la base de datos, el cual recibira el servidor cuando el cliente haya modificado algún valor relacionado con el sensor temperatura, ya sea la máxima,minima o el aire acondicionado.

```
        client.on('insertarTemperatura', (data) => {
            collectionTemperatura.insertOne(data).then(() => {
                //console.log(data);
                io.emit('obtenerTemperatura', data);
            }).catch((error) => {
                console.error("Error insertando la temperatura en la base de datos: ", error);
            });
        });
```

Listing 3: Evento insertar temperatura.

Basicamente lo que hacemos es insertar en la colección de temperatura la data que nos han mandado del cliente o del servidor, ya sea porque se ha modifico un sensor, el cliente ha apagado o encendido algún actuador o el propio agente ha modificado algún actuador.

Ahora vamos a ver los eventos relacionados con la colección que guarda los eventos reciente, primero vamos a ver como los obtenemos de la base de datos y después como los insertamos.

```
        collectionLastsEvents.find().sort({ $natural: -1 }).limit(10).toArray().then((data) =>
     {
            if (data.length > 0) {
                // Emitir los datos mas recientes de la base de datos (ultimo registro)
                client.emit('obtenerEventosRecientes', data);
```

```
            }
        }).catch((error) => {
            console.error("Error obteniendo eventos recientes: ", error);
        });
```

Listing 4: Evento obtener eventos reciente.

Como vemos lo que hacemos es sacar los diez eventos mas recientes de la colección y emitirselo a todos los clientes para que los tengan.

Para insertarlos vamos a tener basicamente lo mismo que para insertar la temperatura solo que esta vez lo haremos en la colección de eventos recientes y tendremos que volver a sacar de la colección los diez eventos más recientes, cosa que no era necesaria hacer con las temperatura ya que podiamos emitir solo el último evento que se acababa de insertar ya que era el que nos interesaba, en este caso necesitamos lo diez últimos.

```
        client.on('insertarEventoReciente', (data) => {
            collectionLastsEvents.insertOne(data).then(() => {
                //console.log(data);
                collectionLastsEvents.find().sort({ $natural: -1 }).limit(10).toArray().then((
    data) => {
                    if (data.length > 0) {
                        //Emitir los datos mas recientes de la base de datos (ultimo registro)
                        io.emit('obtenerEventosRecientes', data);
                    }
                }).catch((error) => {
                    console.error("Error obteniendo eventos recientes: ", error);
                });
            }).catch((error) => {
                console.error("Error insertando evento reciente en la base de datos: ", error)
    ;
            });
        });
```

Listing 5: Evento obtener eventos recientes.

## 2.4. Eventos relacionados con el agente

Vamos a pasar a ver como gestionamos los eventos de los actuadores. Para ello vamos a meternos con el agente ya que es el encargado de gestionar esto.
Para el agente lo que se ha hecho es instalar mediante npm la libre socket-io-client que permite que un archivo javascript se nos conecte al servidor in necesidad de que este este incrustado en un archivo HTML.
La función principal de nuestro agente va a ser que siempre que se modifique algun sensor recibirlo mediante los eventos de obtenerTemperatura, obtenerLuminosidad,etc.. y va a gestionarlo de la siguiente manera.

```
socket.on('obtenerTemperatura', (data) => {
    let resultado = comprobarActuadorAire(data);
    //Comprobamos si se ha modificado el estado del aire
    if(resultado.modificado){
        let newData = resultado.data;
        socket.emit('insertarTemperatura', newData);
        socket.emit('actuadorTemperaturaModificado',newData);
        if(newData.estadoAire){
            let fecha = new Date();
            let fechaFormateada = formatearFecha(fecha);
            socket.emit('insertarEventoReciente', {evento: 'Se ha encendido el aire por alta
    temperatura',fecha: fechaFormateada});
        }else{
            let fecha = new Date();
            let fechaFormateada = formatearFecha(fecha);
            socket.emit('insertarEventoReciente', {evento: 'Se ha apagado el aire por baja
    temperatura',fecha: fechaFormateada});
        }
```

```
    }else{//Si no se ha modificado el estado del aire, comprobamos si la temperatura ha bajado
     de la maxima y en caso de ser asi avisamos
        console.log('Comprobamos...');
        if(data.actual < MAX_TEMPERATURA) socket.emit('temperaturaMaxima',false);
        else socket.emit('temperaturaMaxima',true);
    }
});
```

Listing 6: Evento obtener temperatura en el agente.

Lo primero que va a hacer es llamar a una función que se llama comprobarActuadorAire y le vamos a pasar la fila que se acaba de insertar en la base de datos.La función es la siguiente.

```
function comprobarActuadorAire(data){
    var modificado = false;
    MAX_TEMPERATURA = data.maxima;
    MIN_TEMPERATURA = data.minima;
    if(data.actual >= MAX_TEMPERATURA && data.estadoAire == false){
        data.estadoAire = true;
        modificado = true;
    }else if(data.actual <= MIN_TEMPERATURA && data.estadoAire == true){
        data.estadoAire = false;
        modificado = true;
    }
    return {data: data,modificado: modificado};
}
```

Listing 7: Función comprobarActuadorTemperatura.

Lo que va a hacer este método es actualizar los valores de maxima temperatura y mínima por si el cliente las ha modificado, y posteriormente comparar el valor actual del sensor con el máximo y comprobando que este apagado, ya que en caso de ser así entonces estamos en el límite de temperatura máxima y el agente va a poner a true l estado del aire acondiconado, además de una variable que controla si el agente a modificado dicho aire.Luegos vamos a tener otra comprobación pero para el limite anterior, ya que en dicho caso el agente debera apagar el aire acondicionado.
Una vez hecho esto se devolvera el data recibido con el valor del aire modificado si fue necesario y con la variable modificado para saber si se modifico algo o no.
LO siguiente que se hara es comprobar si se ha modificado algo y en caso de ser asi insertar primero una nueva fila de temperatura ya que se modifico el aire acondicionado e insertar el evento en la colección de eventos recientes, mediante el evento de insertarEventos.Tambiñen hacer un emir de actuadorTemperaturaModificado para que el servidor sepa que el agente a modificado el aire y en consecuencia vaya a avisar a los clientes.
En caso de que no se haya modificado el aire el agente va a comprobar si estamos en máxima temperatura para decirselo al servidor y que este a su vez puede decirselo al cliente y se vea el mensaje de aviso de máxima temperatura en el cliente.

Ahora vamos a ver como el servidor gestiona los diferentes eventos que le va a mandar el agente en función de lo que este haga.
El primer evento es el actuadorTemperaturaModificado que es el evento que recibe el servidor cuando el agente a modificado el estado del aire acondicionado.

```
        client.on('actuadorTemperaturaModificado', (data) => {
            io.emit('obtenerActuadorTemperatura', data);
            if(data.estadoAire){
                console.log('Estado aire: ', data.estadoAire);
                console.log('Enviando mensaje a: ', userId);
                bot.telegram.sendMessage(userId, 'El aire acondicionado ha sido encendido');
            }
        });
```

Listing 8: Evento actuadorTemperaturaModificado.

Este evento lo que hara es avisar a todos los clientes mediante obtenerActuadorTemperatura de que el agente ha modificado el aire y mandarle dicho valor y después si el aire esta encendido hacer que se el bot mande un mensaje,

porque como explicaremos después se ha implementado un bot que avisa de la modificación de actuadores ya sea por parte del cliente o del propio agente.

El último evento que vamos a ver en el servidor es el de temperaturaMaxima que es el evento en el que el agente manda true si estamos en temperatura máxima y false en caso contrario y el agente simplemente lo que hará es reenviarselo al cliente para mostrar o dejar de mostrar el aviso de temperatura máxima.

```
        client.on('temperaturaMaxima', (data) => {
            if(data) io.emit('temperaturaMaximaAviso', data);
            else io.emit('temperaturaMaximaAviso', data);
        });
```

Listing 9: Evento temperatura máxima.

## 2.5. Eventos relacionados con el cliente

Vamos a pasar a ver como el cliente recibe y trata los distintos eventos que le manda el servidor.
Pero antes de explicarlos, tenemos que saber que hay una página principal con información general de cada sensor(temperatura actual, temperatura máxima, temperatura mínima y estado del actuador) y luego los diez eventos más recientes.Y por otro lado para cada sensor tenemos una página en la que podemos modificar la temperatura máxima, mínima y la actual, además de poder encender y apagar dicho actuador con el que esta relacionado.

Aquí dos capturas de cada tipo de página: Este seria la página principal mencionada anteriormente. Y esta la


Figura 7: Página Principal.

página para modificar los distintos valores relacionados con la temperatura.

Una vez explicado y visto lo anterior vamos a ver de la página principal como mostramos los eventos cuando nos lo manda el servidor y de la de modificar temperatura como modificamos el valor del sensor y como apagamos y encemos el aire.
Vamos a empezar por los eventos de la principal.

```
socket.on('obtenerEventosRecientes', (data) => {
    actualizarEventosRecientes(data);
```

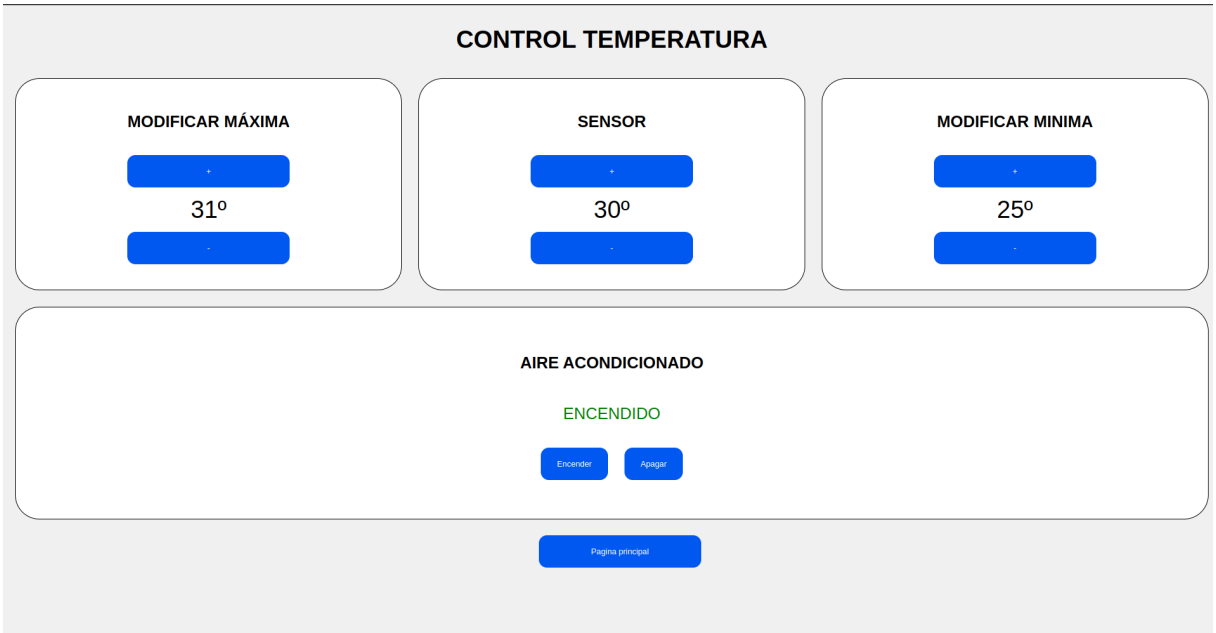


Figura 8: Modificar Temperatura.

```
});
```

Listing 10: Evento obtenerEventosRecientes.

Cuando el servidor hace el emit de obtener eventos recientes lo que hacemos en el cliente es suscribirnos mediante el on y mandarlo a una función actualizarEventosRecientes, mandandole el data que nos mando el servidor. Esta función básicamente añade de manera dinámica a la página los diez últimos eventos.

```
function actualizarEventosRecientes(data){
    //console.log(data);
    let contenedorEventos = document.getElementById('events');
    contenedorEventos.innerHTML = '';

    data.forEach(eventoData => {
        let divEvento = document.createElement('div');
        divEvento.className = 'event';

        let pNombre = document.createElement('p');
        pNombre.className = 'name-event';
        pNombre.textContent = eventoData.evento;

        let pHora = document.createElement('p');
        pHora.className = 'hour-event';
        pHora.textContent = eventoData.fecha;

        divEvento.appendChild(pNombre);
        divEvento.appendChild(pHora);

        contenedorEventos.appendChild(divEvento);
    });
}
```

Listing 11: Evento obtenerEventosRecientes.

Ahora vamos a pasar a ver como modificamos el sensor temperatura en la pagina modificar temperatura y el estado del aire acondicionado.
Vamos a obtener del DOM los botones con ID aumentarTemperatura y disminuirTemperatura que sen el botón con el + y el - respectivamente. Una vez hecho esto les añadimos un event listener de cuando se les haga click y

cuando est suceda le quitaremos el evento por defecto y llamaremos a una dunción que se llama modificarTemperaturaActual y le mandaremosa dicha fucnión true, cuando queramos estemos subiendo la temperatura y false en caso contrario.

```
document.getElementById('aumentar-temperatura').addEventListener('click', function(event){
    event.preventDefault();
    modificarTemperaturaActual(true);
});
document.getElementById('disminuir-temperatura').addEventListener('click', function(event){
    event.preventDefault();
    modificarTemperaturaActual(false);
});
```

Listing 12: Listener cuando pulsamos el botón + o -.

Ahora veamos que hace modificarTemperaturaActual.

```
function modificarTemperaturaActual(valor){
    let temperaturaA = parseInt(temperaturaActual.textContent);
    let fecha = new Date();
    let fechaFormateada = formatearFecha(fecha);
    if(valor){
        let temperaturaNueva = temperaturaA + variacion;
        socket.emit('insertarTemperatura', {actual: temperaturaNueva, maxima: parseInt(
    temperaturaMaxima.textContent), minima: parseInt(temperaturaMinima.textContent),fecha:
    fechaFormateada, estadoAire: aireAcondicionado.textContent == 'Encendido' ? true : false})
    ;
    }else{
        let temperaturaNueva = temperaturaA - variacion;
        socket.emit('insertarTemperatura', {actual: temperaturaNueva, maxima: parseInt(
    temperaturaMaxima.textContent), minima: parseInt(temperaturaMinima.textContent),fecha:
    fechaFormateada, estadoAire: aireAcondicionado.textContent == 'Encendido' ? true : false})
    ;
    }
};
```

Listing 13: Función modificarTemperaturaActual.

Esta función obtiene el valor del elemento del DOM con ID temperaturaActual y le saca el valor mediante el textContent para posteriormente pasarlo a entero, obtiene la fecha actual y comprueba si se le ha pasado true o false. En caso de true es que queremos aumentar la temperatura y por tanto le añadimos a temperaturaA que es la actual la variación que es una variable constante para modificar el valor del sensor.Una vez hecho esto emitimos el evento de insertarTemperatura rescatando los valores que ya teniamos y mandando el nuevo de temperaturaActual al servidor para que este lo inserte en la base de datos.
Para modificar la temperatura máxima o minima seria exactamante lo mismo.

Vamos por último a ver como modificamos el estado del aire acondicionado.
Para esto vamos a tener dos botones una para encender y otro para apagar, y para cada uno vamos a añadirle también un event listener para cuando se le haga click y en función del que sea haremos una cosa u otra.

```
encenderAire.addEventListener('click', function(event){
    event.preventDefault();
    let fecha = new Date();
    let fechaFormateada = formatearFecha(fecha);
    socket.emit('insertarTemperatura', {actual: parseInt(temperaturaActual.textContent),
    maxima: parseInt(temperaturaMaxima.textContent), minima: parseInt(temperaturaMinima.
    textContent),fecha: fechaFormateada, estadoAire: true});
    socket.emit('insertarEventoReciente', {evento: 'El usuario ha encendido el aire',fecha:
    fechaFormateada});
    socket.emit('actuadorTemperaturaModificado', {actual: parseInt(temperaturaActual.
    textContent), maxima: parseInt(temperaturaMaxima.textContent), minima: parseInt(
    temperaturaMinima.textContent),fecha: fechaFormateada, estadoAire: true});
});
```

Listing 14: Función modificarTemperaturaActual.

Este es el ejemplo de encender el aire pero el de apagar será similir y solo cambiara el valor del aire que insertamos en la base de datos y que emitimos.
Lo que hacemos simplemente es quitar evento por defecto, luego obtener fecha actual y decirle al servidor mediante el emit que inserte nuevo valor en la colección de temperatura y de eventos recientes ya que se ha modifcado el estado del aire y por último emitir que hemos modificado el estado del aire mediante el evento explicado anteriormente de actuadorTemperaturaModificado que lo que hacia es avisar a todos los clientes que el estado del aire acondicionado ha sido modificado.

## 2.6. Configuración Bot Telegram

Lo último que queda por explicar es como hemos configurado el bot de telegram que nos manda avisos de de cuando se enciende o apaga alguno de los tres actuadores que controlamos, ya sea por que lo enciende o apaga el agent, o la hace el propio usuario.
Para hacer esto he usado el paquete telegraf de npm y para su instalación y configuración se ha seguido la documentación oficial en esta página: **Telegraf.js**.
Ahora vamos a ver que configuración hemos hecho en el servidor para mandar los mensajes y recibir los distintos comandos en el servidor, cuando se lo mande al bot el cliente.

```
const bot = new Telegraf('7103064786:AAFf7K9J9tZsyTJsEPESKeI6-VevRZDKnBA');
bot.start((ctx) => {
    console.log('/start command received');
    ctx.reply('Bienvenido al bot de domotica use el comando /help para ver los comandos
    disponibles');
});

bot.help((ctx) => {
    console.log('/help command received');
    ctx.reply('Comandos disponibles:\n\n/start - Iniciar el bot\n/help - Mostrar ayuda\n/
    temperatura - Obtener la temperatura actual\n/luminosidad - Obtener la luminosidad actual\
    n/dioxido - Obtener el nivel de dioxido de carbono actual\n/avisos - Obtener los avisos de
     aire acondicionado, persianas y filtro de dioxido de carbono');
});

bot.command('avisos', (ctx) => {
    console.log('/avisos command received');
    userId = ctx.message.from.id;
    ctx.reply('Suscripcion a avisos activada, cuando se modifique un actuador se le enviara un
     mensaje');
    console.log('User ID: ', userId);
});


// Lanzar el bot
bot.launch().then(() => {
    console.log('Bot launched');
});
```

Listing 15: Comandos Básicos.

La primera linea es el token del bot, luegos tenemos recibimos el comando start y el comando help cuando nos lo mande el cliente.El comando start le pide al cliente que mande /help para conocer los distintos comandos disponibles, el help le muestra cada uno de estos comandos que puede usar el cliente y por último avisos hace permite la suscripción ha recibir un mensaje de aviso, cuando se modifica alguno de los tres actuadores.
Por último tenemos el launch para lanzar el bot.

Ahora vamos a ver que hacemos cuando el cliente nos manda el comando de /temperatura.

```
    bot.command('temperatura', (ctx) => {
        collectionTemperatura.find().sort({ $natural: -1 }).limit(1).toArray().then((data) =>
    {
            if (data.length > 0) {
                    console.log('/temperatura command received');
```

```
                    let aire = 'APAGADO';
                    if(data[0].estadoAire)  aire = 'ENCENDIDO';
                    ctx.reply(
                        'La temperatura maxima es de ' + data[0].maxima + 'C' +
                        '\nLa temperatura actual es de ' + data[0].actual + 'C' +
                        '\nLa temperatura minima es de ' + data[0].minima + 'C' +
                        '\nEl estado del aire acondicionado es ' + aire
                    );
            }
        }).catch((error) => {
            console.error("Error obteniendo la temperatura: ", error);
        });
    });
```

Listing 16: Comando Temperatura.

Cuando el cliente mande el comando temperatura lo que haremos es sacar de la base de datos la última linea y mediante ctx.reply(), responderle al cliente mostrandole el valor máximo,minimo y actual detemperatura como el estado del aire.
Para los comando de luminosidad y el dioxido será exactamente igual.

Lo último es ver los mensajes que mandamos al cliente cuando se modifica el estado del aire acondicionado.

```
        client.on('actuadorTemperaturaModificado', (data) => {
            io.emit('obtenerActuadorTemperatura', data);
            if(data.estadoAire){
                console.log('Estado aire: ', data.estadoAire);
                if(userId!=null){
                    console.log('Enviando mensaje a: ', userId);
                    bot.telegram.sendMessage(userId, 'El aire acondicionado ha sido encendido'
    );
                }
            }else{
                console.log('Estado aire: ', data.estadoAire);
                if(userId!=null){
                    console.log('Enviando mensaje a: ', userId);
                    bot.telegram.sendMessage(userId, 'El aire acondicionado ha sido apagado');
                }
            }
        });
```

Listing 17: Mensaje de aviso aire modificado.

Cuando recibamos el evento actuadorAireModificado lo que haremos es comprobar si esta encendido o apagado y si el ID de telegram del usuario no es nulo,ya que esto significaria que este se ha suscrito al evento avisos y por tanto debemos mandarle un mensaje de aviso con la modificación del aire acondicionado.

A continuación dejo el enlace del bot para su uso: **Bot Domótica**.