Miguel Medina Martínez

DOMOTIC - DSD

Mi sistema demótico consta de 3 archivos HTML (usuario, historial, sensores) y un archivo JavaScript que actúa como servidor.

Servidor:

```
var httpServer = http.createServer(
    function(request, response) {
        var uri = url.parse(request.url).pathname;
        if (uri=="/") uri = "/usuario.html";
        var fname = path.join(process.cwd(), uri);
        fs.exists(fname, function(exists) {
            if (exists) {
                fs.readFile(fname, function(err, data){
                    if (!err) {
                        var extension = path.extname(fname).split(".")[1];
                        var mimeType = mimeTypes[extension];
                        response.writeHead(200, mimeType);
                        response.write(data);
                        response.end();
                    }
                    else {
                        response.writeHead(200, {"Content-Type": "text/plain"});
                        response.write('Error de lectura en el fichero: '+uri);
                        response.end();
                    }
                });
            }
            else{
                console.log("Peticion invalida: "+uri);
                response.writeHead(200, {"Content-Type": "text/plain"});
                response.write('404 Not Found\n');
                response.end();
            }
        });
    }
);
```

Creamos el servidor HTML y lo redirigimos a /usuario.html

En el servidor inicializamos las variables de nuestro sistema además de los valores limites que pueden tomar.

```
var temperatura = 20;
var temperaturaMax = 60;
var temperaturaMin = 0;
var luminosidad = 50;
var luminosidadMax = 80;
var luminosidadMin = 20;
var aire = 0;
var persiana = 1;
```

Lo siguiente que encontramos es la conexión a mongodb. Nos conectamos a la base de datos sensoresDB y creamos la colección sensores. Al producirse la conexión comprobamos si hay más de dos entradas en la base de datos. Si no hay las añadimos para evitar errores más adelante (Estas entradas no se ven en el historial ni se tweetean). Contamos con 4 funciones de socket.io:

```
client.emit('conexion', {});

client.on('poner', function (data) {
    update(data);
    if (agente()){
        collection.insertOne(data, {safe:true}, function(err, result) {});
        tweet(data);
    }

});

client.on('obtener', function (data) {
    collection.find().toArray(function(err, results){
        client.emit('obtener', results);
    });
});

client.on('getTemperaturaGranada', function (data) {
    var url = 'http://api.openweathermap.org/data/2.5/weather?q=Granada,es&APPID=bc1d8a6611b1a3a35f4c8b898ec7eb19';
    request(url,function(err,response,body){
        if(err){
            console.log('error:', error);
          } else {
            var api = JSON.parse(body);
            var temperatura = parseInt(api.main.temp) - 273.15;
            temperatura = temperatura.toFixed(2);
            client.emit('getTemperaturaGranada',temperatura);
          }
    });
});
```

- Conexión: Avisa al cliente que acaba de conectarse al servidor
- Poner: actualiza los parámetros locales del sistema y comprueba llamando al agente si se puede hacer una inserción con los datos sin modificar. En ese caso lo hace y llama a la función de tweet.
- Obtener: devuelve todos los datos de la colección.
- getTemperaturaGranada: Devuelve la temperatura actual que hace en granada haciendo un request a la API de openweathermap.org

También contamos con otras 3 funciones:

```
function update(data){
    persiana = data.pers;
    aireAcondicionado = data.ac;
    temperatura = data.temp;
    luminosidad = data.lumi;
}

function agente(){
    var result = true;
    var alerta = '';

    if ((luminosidad > luminosidadMax) && persiana == 1){
        persiana = 0;
        result = false;
        alerta+= "\nLuminosidad muy Alta! Subiendo Persiana";
    }
    if ((luminosidad <= luminosidadMin) && persiana == 0){
        persiana = 1;
        result = false;
        alerta+= "\nLuminosidad muy Baja! Bajando Persiana";
    }

    if ((temperatura > temperaturaMax) && aireAcondicionado==0){
        aireAcondicionado = 1;
        result = false;
        alerta+= "\nTemperatura muy alta! Encendiendo el Aire";
    }
    if ((temperatura < temperaturaMin) && aireAcondicionado==1){
        aireAcondicionado = 0;
        result = false;
        alerta+= "\nTemperatura muy baja! Apagando el Aire";
    }

    if (result == false){
        var d = new Date();
        collection.insertOne({time:d, temp:temperatura, lumi:luminosidad, ac:aireAcondicionado, pers:persiana}, {safe:true}, function(err, result) {});
        tweet({time:d, temp:temperatura, lumi:luminosidad, ac:aireAcondicionado, pers:persiana});
        io.sockets.emit('alerta', alerta);
    }

    return result;
}
```

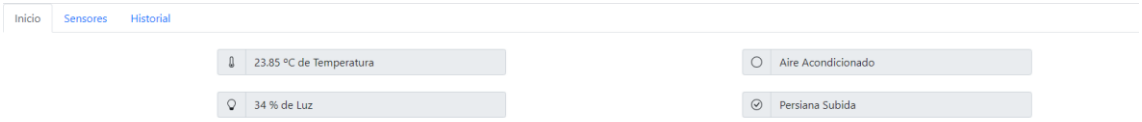
- Update: Actualiza los parámetros locales del servidor usando los que se le pasan como argumento.
- Agente: Es el encargado de comprobar si la temperatura o luminosidad activan algún actuador. Con distintos ifs controlamos que si la luminosidad o temperatura pasan de sus máximos se baja la persiana y se activa el aire acondicionado respectivamente o que si la luminosidad o temperatura pasan de sus mínimos se sube la persiana y se apaga el aire acondicionado respectivamente. Además se encarga de insertar en la base de datos los datos si necesitan ser modificados y mandar un emit con una alerta que explica que activador está siendo activado.
- Tweet: usando node js y twitter lite he creado una cuenta que refleja todos los cambios en el servidor twitteándolos de forma automática. He seguido este tutorial (https://dev.to/ahmed_mahallawy/tweeting-using-node-js-5986) y tras instalar y configurar el modulo he importado lo necesario y he hecho una función que escribe un tweet con todo lo necesario.

```
const config = require('./config');
const twitter = require('twitter-lite');
const clientTWT = new twitter(config);
```

```
function tweet(data){

    var auxaire;
    if (data.ac == 0){
        auxaire = "OFF";
    }
    else{
        auxaire = "ON";
    }
    var auxpers;
    if (data.pers == 0){
        auxpers = "Bajada";
    }
    else{
        auxpers = "Subida";
    }

    var msg = "Actualizacion en el Servidor ---> Temperatura: "+data.temp+"  |  Luminosidad: "+data.lumi + "  |  Aire Acondicionado: "+ auxaire+ "  |  Persiana: "+ auxpers;

    clientTWT.post('statuses/update', { status: msg }).then(result => {

    }).catch(console.error);
}
```

A continuación voy a documentar las 3 distintas páginas de las que dispongo. La interfaz la he hecho usando bootstrap y mirando en su extensa documentación.

<u>Usuario/Inicio:</u>



Esta página nos permite visualizar la temperatura y la luminosidad además de activar o desactivar el aire acondicionado y bajar o subir la persiana.

```
var persiana = 1; //0 Abajo, 1 Arriba
var aireAcondicionado = 0; //0 Apagado, 1 Encendido
var temperatura = 20;
var luminosidad = 50;

document.getElementById('aireAcondicionado').onclick = function(e){
    if (aireAcondicionado == 1){
        aireAcondicionado = 0;
    }
    else{
        aireAcondicionado = 1;
    }

    actualizarIconos();
    enviarDatos();
}

document.getElementById('persiana').onclick = function(e){
    if (persiana == 1){
        persiana = 0;
    }
    else{
        persiana = 1;
    }

    actualizarIconos();
    enviarDatos();
}
```

Además de declarar las variables locales, cuenta con dos funciones que controlan si se marcan las casillas de la persiana o el aire acondicionado y que envían estos datos al servidor.

```
function actualizarDatos(){
    if (persiana == 0){
        document.getElementById('persiana').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" cl
    }else{
        document.getElementById('persiana').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" cl
    }

    if (aireAcondicionado == 0){
        document.getElementById('aireAcondicionado').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="current
    }else{
        document.getElementById('aireAcondicionado').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="current
    }

    document.getElementById('temperatura').value = temperatura + " ºC de Temperatura";
    document.getElementById('luminosidad').value = luminosidad + " % de Luz";
}

function actualizarIconos(){
    if (persiana == 0){
        document.getElementById('persiana').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" cl
    }else{
        document.getElementById('persiana').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="currentColor" cl
    }

    if (aireAcondicionado == 0){
        document.getElementById('aireAcondicionado').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="current
    }else{
        document.getElementById('aireAcondicionado').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="current
    }
}

function enviarDatos(){
    var d = new Date();
    socket.emit('poner', {time:d, temp:temperatura, lumi:luminosidad, ac:aireAcondicionado, pers:persiana});
}

function update(data){
    persiana = data.pers;
    aireAcondicionado = data.ac;
    temperatura = data.temp;
    luminosidad = data.lumi;
}
```

- actualizarDatos: Se encarga tanto de actualizar los iconos como la temperatura y luminosidad.
- actualizariconos: Se encarga de actualizar los iconos.
- enviarDatos: Envía los nuevos datos al servidor usando “poner”.
- Update: Actualiza los parámetros locales del servidor usando los que se le pasan como argumento.

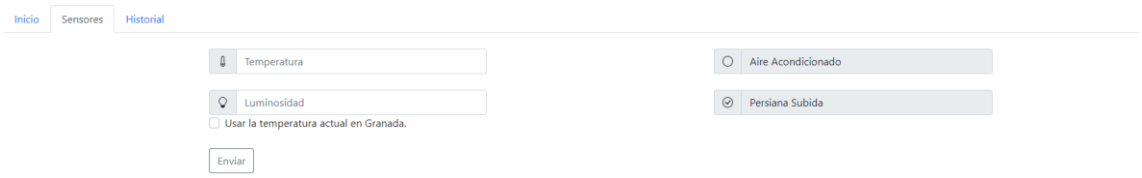
Por ultimo en cuanto a operaciones con socket.io. Cuando recibe las distintas funciones

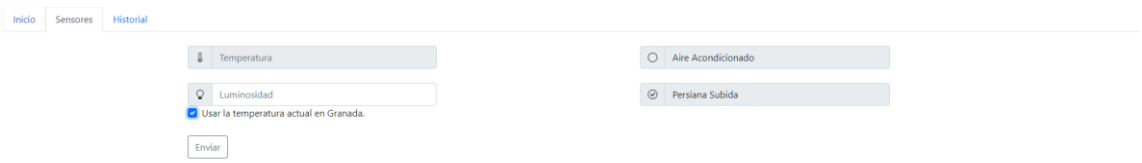
```
var serviceURL = "localhost:8080";
var socket = io.connect(serviceURL);

socket.on('conexion', function(data) {
    socket.emit('obtener',{});
});

socket.on('obtener', function(data) {
    update(data[data.length-1]);
    actualizarDatos();
});

socket.on('alerta', function(data) {
    socket.emit('obtener',{});
    alert(data);
});
```

- Conexión: hace una llamada “obtener” al servidor.

- Obtener: Selecciona la última medición de los datos del servidor y actualiza los datos de la interfaz.
- Alerta: hace una llamada “obtener” al servidor y muestra una alerta en pantalla que se le pasa por argumentos.

Sensores:





Esta página nos permite visualizar el estado de la persiana y el aire acondicionado además de modificar tanto la temperatura como la luminosidad y enviárselo al servidor. También cuenta con una checkbox que nos permite usar la temperatura actual en granada en vez de una introducida por nosotros.

```
var persiana = 1; //0 Abajo, 1 Arriba
var aireAcondicionado = 0; //0 Apagado, 1 Encendido
var temperatura = 20;
var luminosidad = 50;

document.getElementById('usarTemperatura').onclick = function(e){
    if (document.getElementById('usarTemperatura').checked == true){
        document.getElementById('temperatura').readOnly = true;
    }else{
        document.getElementById('temperatura').readOnly = false;
    }
}

document.getElementById('enviar').onclick = function(e){
    enviarDatos();
}
```

Además de declarar las variables locales, cuenta con dos funciones que controlan si se marca la casilla de usar la temperatura de granada y otra que se encarga de enviar los datos cuando se pulsa el botón de enviar.

```
function actualizarDatos(){
    if (persiana == 0){
        document.getElementById('persiana').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="
    }else{
        document.getElementById('persiana').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="16" fill="
    }

    if (aireAcondicionado == 0){
        document.getElementById('aireAcondicionado').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="1
    }else{
        document.getElementById('aireAcondicionado').innerHTML = '<svg xmlns="http://www.w3.org/2000/svg" width="16" height="1
    }
}

function enviarDatos(){
    var d = new Date();
    var aux = document.getElementById('temperatura').value;
    if (aux != ''){
        temperatura = aux;
        document.getElementById('temperatura').value = '';
    }
    aux = document.getElementById('luminosidad').value;
    if (aux != ''){
        luminosidad = aux;
        document.getElementById('luminosidad').value = '';
    }
    if (document.getElementById("usarTemperatura").checked == 1){
        socket.emit('getTemperaturaGranada', {});
    }
    else{
       socket.emit('poner', {time:d, temp:temperatura, lumi:luminosidad, ac:aireAcondicionado, pers:persiana});
    }

}

function update(data){
    persiana = data.pers;
    aireAcondicionado = data.ac;
    temperatura = data.temp;
    luminosidad = data.lumi;
}
```

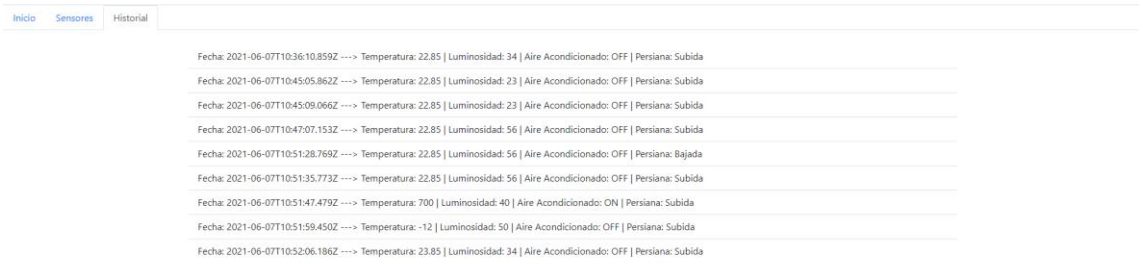
- actualizarDatos: Se encarga de actualizar los iconos.
- enviarDatos: Comprueba si los inputs para temperatura y luminosidad están vacíos. Si están vacíos usa los valores que tenía previamente. En caso de que si hayan sido rellenados, los actualiza. Si la casilla de usar el tiempo actual de granada está marcada llama a “getTemperaturaGranada” y NO envía los datos al servidor. En caso contrario los envía al servidor usando “poner”.
- Update: Actualiza los parámetros locales del servidor usando los que se le pasan como argumento.

Por ultimo en cuanto a operaciones con socket.io. Cuando recibe las distintas funciones

```
var serviceURL = "localhost:8080";
var socket = io.connect(serviceURL);

socket.on('conexion', function(data) {
    socket.emit('obtener',{});
});

socket.on('obtener', function(data) {
    update(data[data.length-1]);
    actualizarDatos();
});

socket.on('alerta', function(data) {
    socket.emit('obtener',{});
    alert(data);
});

socket.on('getTemperaturaGranada', function(data) {
    var d = new Date();
    socket.emit('poner', {time:d, temp:data, lumi:luminosidad, ac:aireAcondicionado, pers:persiana});
});
```

- Conexión: hace una llamada “obtener” al servidor.
- Obtener: Selecciona la última medición de los datos del servidor y actualiza los datos de la interfaz.
- Alerta: hace una llamada “obtener” al servidor y muestra una alerta en pantalla que se le pasa por argumentos.
- getTemperaturaGranada: recibe la temperatura actual de Granada y mediante una llamada “poner” envía al servidor los nuevos datos.

Historial:



Esta página simplemente nos muestra un historial legible de los cambios que se han producido en nuestro sistema.

```
function actualizarLista(cambios){
    var listContainer = document.getElementById('historial');
    listContainer.innerHTML = '';
    var listElement = document.createElement('ul');
    listContainer.appendChild(listElement);
    var num = cambios.length;
    for(var i=2; i<num; i++) {
        var listItem = document.createElement('li');
        var auxaire;
        if (cambios[i].ac == 0){
            auxaire = "OFF";
        }
        else{
            auxaire = "ON";
        }
        var auxpers;
        if (cambios[i].pers == 0){
            auxpers = "Bajada";
        }
        else{
            auxpers = "Subida";
        }
        listItem.innerHTML = "Fecha: "+cambios[i].time+" ---> Temperatura: "+cambios[i].temp+"  |  Luminosidad: "+cambios[i].lumi + "  |  Aire Acondicionado: "+ auxaire+ "  |  Persiana: "+ auxpers;
        listElement.appendChild(listItem);
    }
}
```

Cuenta con una función (Que he adaptado del ejemplo de mongo db para que muestre información más legible) que añade a una lista cada una de las entradas que están almacenadas en la base de datos.

En cuanto a operaciones con socket.io. Cuando recibe las distintas funciones

```
var serviceURL = "localhost:8080";
var socket = io.connect(serviceURL);

socket.on('conexion', function(data) {
    socket.emit('obtener',{});
});

socket.on('obtener', function(data) {
    actualizarLista(data);
});
```

- Conexión: hace una llamada “obtener” al servidor.
- Obtener: Llama a la función actualizarLista con los datos que ha recibido por parte del servidor.