Práctica 4. Uso de Node.js y Mongo DB, y su interconexión a través de Sockets.

4.1 Revisando los ejemplos

4.1.1 Ejemplo de calculadora

El primer ejemplo, para probar que Node funcionaba bien, era ejecutar una calculadora. Si bien había dos versiones, tenían un punto clave, que era cómo se definía su estructura. Pongamos un ejemplo:

`http://localhost:8080/sumar/3/2`

donde:

`http://localhost:8080` es la dirección donde se sirve el archivo HTML asociado a ese archivo Node de la calculadora, en este caso **`calc.html`**

`/sumar/` es uno de los parámetros que trata en la dirección. Para extraer ese parámtro, se extrae el path de la dirección, de divide en cadenas según el caracter **`/`** y se coge el primer fragmento, el cual se compara con las palabaras clave sumar, restar, producto y dividir, para hacer la operación pertinente.

Con la segunda mitad de la URI (**`/3/2`**), se vuelve a separar para obtener los operandos, y así poder hacer el cálculo y entregárselo al HTML asociado.

La diferencia entre las dos versiones es que mientras la primera sólo tiene el manejo con las URIs, la segunda añade en su dirección base (`http://localhost:8080`) un formulario para añadir los datos y la operación a realizar.

4.1.2 Ejemplo de conexiones a través de Socket.io

Para poder hacer uso de las conexiones se requiere instalar el paquete socket.io:

```
npm install socket.io
```

Al parecer, los sockets son una especie de Message-Oriented Middlewares, donde los mensajes son etiquetados según el socket yvan realizando operaciones con los datos que tienen esa etiqueta.

En el programa puedo detectar que hay como dos ámbitos de envío de mensajes a través de los sockets:

- **`socket`**/**`client`**, que reciben y envían mensajes entre ellos y además reciben mensajes globales.
- **`io.sockets,`** que hace un broadcast a todos quienes reciben mensajes por ese socket.

A partir de esos fundamentos, se trata de levantar una página HTML que permite ver el estado del servidor (con **`connect`**/**`disconnect`**).

Además, se da un socket para ir añadiendo las direcciones en esa página (de forma que si se entra a esa misma página en otra pestaña, te indica dinámicamente su dirección).

Además, en el servidor se queda a modo de log con las conexiones y desconexiones, incluso la lista de conexiones.

4.1.3 Ejemplo con MongoDB

Para poder usar las operaciones de la Base de Datos en Node, se ha instalado el paquete correspondiente:

```
npm install mongodb
```

El ejemplo es una extensión del ejemplo anterior, donde se registran las conexiones en esta Base de Datos NoSQL.

Sin embargo, tuve problemas a la hora de ejecutar este código, ya que si intentaba crear de nuevo la colección, me saltaba una excepción, así que fui pensando en alternativas. La primera que pensé fue ver si existía la colección en la base de datos, y si no crearla. Pero por consecuencia no podía evitar la repetición de código, lo que podía causar dolores de cabeza en el desarrollo. Así que opté por otra vía, que era capturar el error en caso de que me saltara ese error por ese motivo, y asiganr la variable a la colección ya creada.

4.2 El sistema Domótico

4.2.1 El servidor

El servidor se encarga de realizar las conexiones con Socket.io y MongoDB. De hecho, el servidor es el único que tiene acceso a la Base de Datos, y para compartir los resultados u otros datos, o para introducir nuevos regsitros; se transmiten desde las vistas hacia el servidor a través de esos sockets.

Además, desde el servidor, se almacenan los estados de los actuadores, los cuales se han de modificar desde ahí (si bien otras vistas, a través de sockets, podrán solicitar la modificación).

Nótese que tiene de base el código del segundo ejemplo, ya que durante el desarrollo de esta actividad decidí primero hacer el funcionamiento entre los componentes, y luego registrar los parámetros en MongoDB.

4.2.2 La vista de usuario

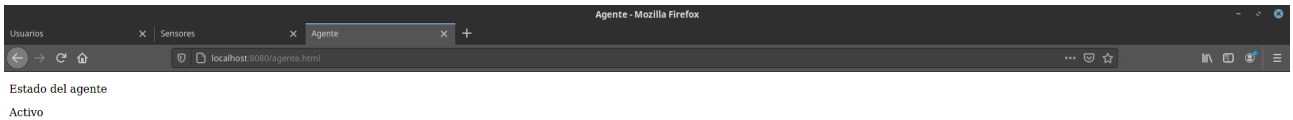
La vista de usuario básica se compone de un visor de los sensores junto con su marca de tiempo, y los estados de los actuadores (AC y motor de las persianas)., que se pueden modificar en los botones (que llaman desde ahí a los sockets pertinentes para cambiar los estados y retornen su estado alterado)

4.2.3 Recogiendo datos desde los sensores

El formulario de recogida de sensores coge los datos de temperatura y luminosidad y se envían al pulsar un botón. Para poder recoger los valores, una vez ejecutado el programa Node, se consulta la web `http://localhost:8080/sensores.html`
Al pulsar un botón,se llamará a una función denominada `sendInfo()`, donde se cogen los valores de los campos y se envían a través de una llamada al socket para que el servidor actualice los valores en la vista de usuario.
También se vacían los campos para que se note un feedback.

4.2.4 El agente y las alertas

El agente se encarga de observar los valores y emitir alertas de ser necesario. Para poder recoger los valores, una vez ejecutado el programa Node, se consulta la web **http://localhost:8080/agente.html** , viendo la siguiente pantalla:



En él se monitoriza el estado del servidor (se comprueba si está levantado con los tags '**connect**' y '**disconnect**'), e internamente gestiona los cambios en la temperatura y la luminosidad. Sus definiciones serían:

```
const TEMP_LIMITS={
MAX: 30,
MIN: 10
};

const LUMENS_LIMITS={
MAX: 70,
MIN: 20
};
```

Según se superen esos límites al alza o a la baja, se realizarán varias acciones:

- Si la temperatura es muy baja, se activaría el Aire Acondicionado para calentar el ambiente.
- Si la temperatura es muy alta, las persianas se bajarán para que no haya mucho sol; y se activaría el Aire Acondicionado para enfriar como se pueda el hogar.
- Si hay mucha luz, se bajan las persianas.

- Si hay poquísima luz, se suben las persianas. Es cierto que no se valora el tema de si es de noche o de día, por lo que puede pasar que en plena noche se suban las persianas y la luminosidad siga baja.

4.2.5 Extra: API meteorológica

Una vez hecha la funcionalidad básica, toca hacer algunas funcionalidades más.
La primera que pensé fue poner una API para el tiempo. A la hora de buscar, pensé en evitar aquellos con un token, ya que mi intención es que se pueda mostrar con independencia del usuario.
La API escogida la ofrece 7timer, que ofrece varias APIs con varias opciones, sin necesidad de token ni instalar nuevos paquetes via **npm**. La que he usado se denomina "CIVIL", y permite consultar la previsión meteorológica por 7 días, convirtiendo el resultado en una imagen que se incrusta en la página de vista del usuario.

El formato sería `http://www.7timer.info/bin/civil.php?lon={LONGITUD}&lat={LATITUD}&ac=0&lang=en&unit=metric&output=internal&tzshift=0`

Una vez se pasen la longitud y latitud en **`long`** y **`lat`** respectivamente, y si están en estos rangos:

- Longitud:[-180,180]
- Latitud:[-90,90]

El sistema generaría la previsión para esas coordenadas. Por defecto, estarían puestas las de Granada, pero se pueden editar desde la vista de recogida de datos de los sensores.

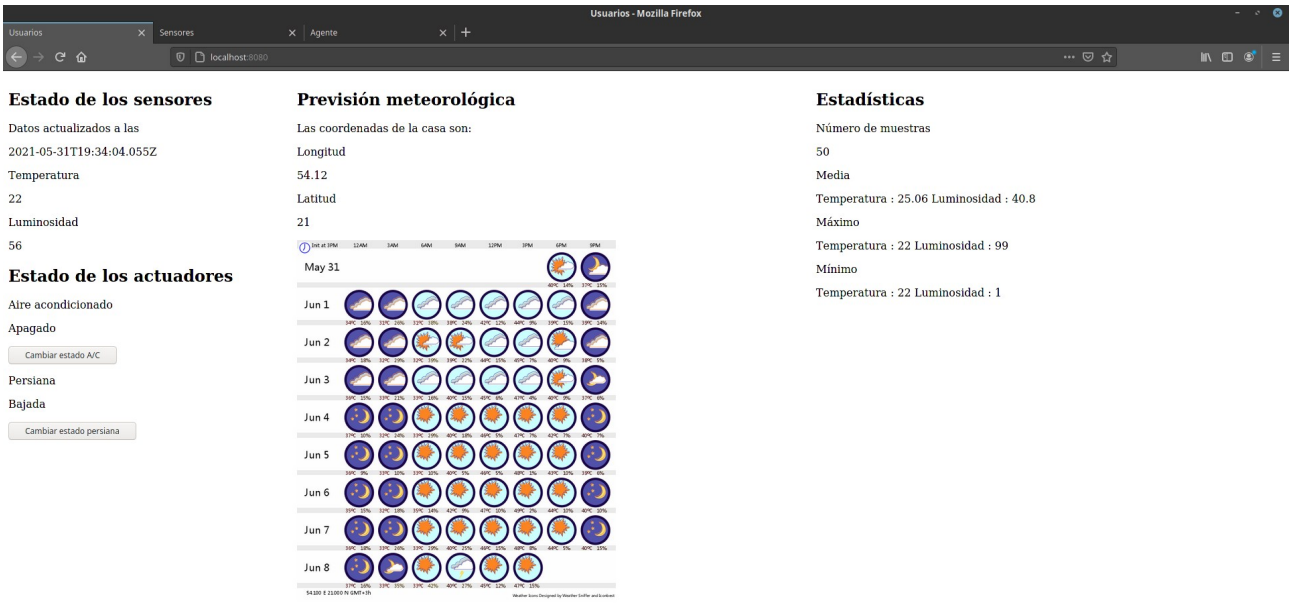
De hecho, como consecuencia, se pueden rellenar o no los campos de longitud y latitud para modificar las coordenadas y ver la previsión en una imagen (si bien esa imagen tarda un tiempo en salir por la vista de usuario) . Para cambiar la imagen, se tiene un string con la forma de la URI donde falta situar la latitud y longitud a cambiar (cuando se haya disparado el socket y existan tales datos).

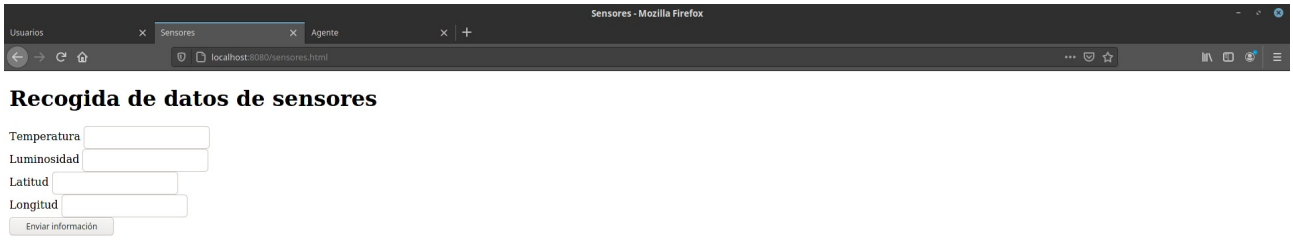
4.2.6 Extra: Muestra de estadísticas

Otra operación extra que quería realizar, para darle uso a la base de datos, es poder tomar los valores máximos, mínimos y medias de las temperaturas y luminosidades, además de la cuenta de registros.

Una vez se recogen los valores por la recogida de datos de los sensores, el socket se encarga de introducirlo en la Base de Datos y se llama desde el mismo socket al cálculo de estadísticas, donde se hace una llamada a toda la colección, se convierte a un array, y se trata de buscar esos parámetros recorriendo los registros.

Dejo anexados un par de imágenes con las vistas de recogida de datos y de usuario donde se visualizan todos los cambios realizados más la funcionalidad Extra.


Usuarios - Mozilla Firefox
Usuarios
Sensores
Agente
localhost:8080
Estado de los sensores
Datos actualizados a las
2021-05-31T19:34:04.055Z
Temperatura
22
Luminosidad
56
Estado de los actuadores
Aire acondicionado
Apagado
Cambiar estado A/C
Persiana
Bajada
Cambiar estado persiana
Previsión meteorológica
Las coordenadas de la casa son:
Longitud
54.12
Latitud
21
May 31
Jun 1
Jun 2
Jun 3
Jun 4
Jun 5
Jun 6
Jun 7
Jun 8
Estadísticas
Número de muestras
50
Media
Temperatura : 25.06 Luminosidad : 40.8
Máximo
Temperatura : 22 Luminosidad : 99
Mínimo
Temperatura : 22 Luminosidad : 1


Sensores - Mozilla Firefox
Usuarios
Sensores
Agente
localhost:8080/sensores.html
Recogida de datos de sensores
Temperatura
Luminosidad
Latitud
Longitud
Enviar información