Práctica 4. NodeJS

Desarrollo de Sistemas Distribuidos

Manuel Guerrero Mesías

UNIVERSIDAD
DE GRANADA

Curso 2021-2022

Índice

1. Parte 1

1.1. Ejercicio 1 - Hola Mundo

Para este ejercicio necesitaremos el archivo helloworld.js.

Este programa realiza el siguiente funcionamiento:

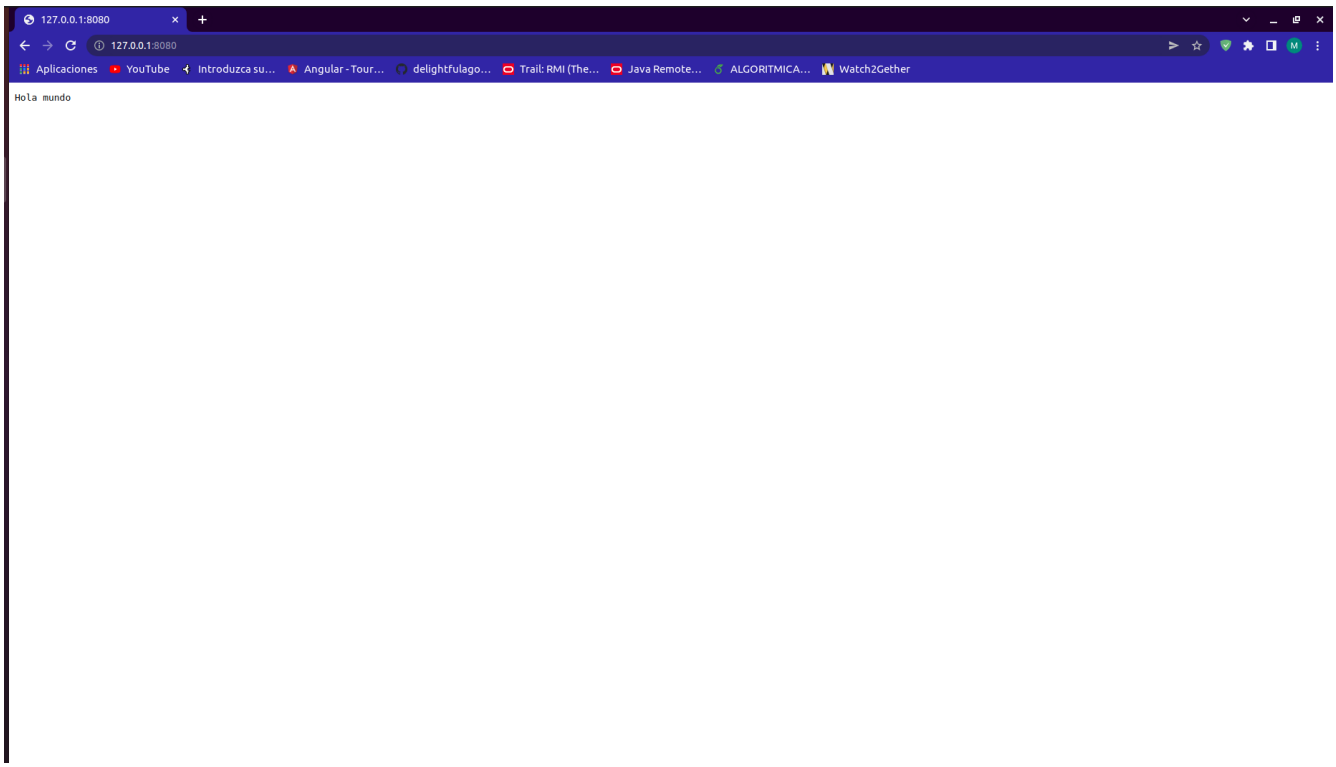
En este programa se realiza el siguiente funcionamiento:

- Obtenemos el modulo "http", el cual nos permite implementar servicios usando el protocolo http.
- Creamos el servidor http, al constructor de este es una función a la que se le pasan los parámetros request y response. Esta función se llamara cada vez que se reciba una petición. El código de la linea 4 muestra en la terminal la información referente a las diferentes peticiones realizadas al servidor. En las lineas 5-7 se genera la respuesta del servidor.
- En la linea 10 se establece el puerto en el cual va a estar escuchando el servidor.
- El código de la linea 11 muestra en la terminal el mensaje "Servicio HTTP iniciado"

Este seria el resultado en la terminal:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P4/ejemplos$ nodejs helloworld.js
Servicio HTTP iniciado
{ host: '127.0.0.1:8080',
  connection: 'keep-alive',
  'sec-ch-ua':
   '" Not A;Brand";v="99", "Chromium";v="101", "Google Chrome";v="101"',
  'sec-ch-ua-mobile': '?0',
  'sec-ch-ua-platform': '"Linux"',
  'upgrade-insecure-requests': '1',
  'user-agent':
   'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36',
  accept:
   'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
  'sec-fetch-site': 'none',
  'sec-fetch-mode': 'navigate',
  'sec-fetch-user': '?1',
  'sec-fetch-dest': 'document',
  'accept-encoding': 'gzip, deflate, br',
  'accept-language': 'es-ES,es;q=0.9' }
{ host: '127.0.0.1:8080',
  connection: 'keep-alive',
  'sec-ch-ua':
   '" Not A;Brand";v="99", "Chromium";v="101", "Google Chrome";v="101"',
  'sec-ch-ua-mobile': '?0',
  'user-agent':
   'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/101.0.4951.64 Safari/537.36',
  'sec-ch-ua-platform': '"Linux"',
  accept:
   'image/avif,image/webp,image/apng,image/svg+xml,image/*,*/*;q=0.8',
  'sec-fetch-site': 'same-origin',
  'sec-fetch-mode': 'no-cors',
  'sec-fetch-dest': 'image',
  referer: 'http://127.0.0.1:8080/',
  'accept-encoding': 'gzip, deflate, br',
  'accept-language': 'es-ES,es;q=0.9' }
```

Este seria el resultado en el navegador



Como podemos apreciar la ejecucion del programa no termina aunque se haga una respuesta, ya que la maoyria de las operaciones de Nodejs son no bloqueantes, pero se impide la finalización de un programa mientras existan puertos en uso, lo que facilita la creación de servicios con una arquitectura dirigida por eventos.

1.2. Ejercicio 2 - Calculadora REST

Este ejemplo es una versión mas avanzada del ejemplo anterior, este se basa en una calculadora online el cual va a sacar los datos de la url del navegador.

Las partes que podemos destacar con respecto al codigo del ejemplo anterior son las siguientes:

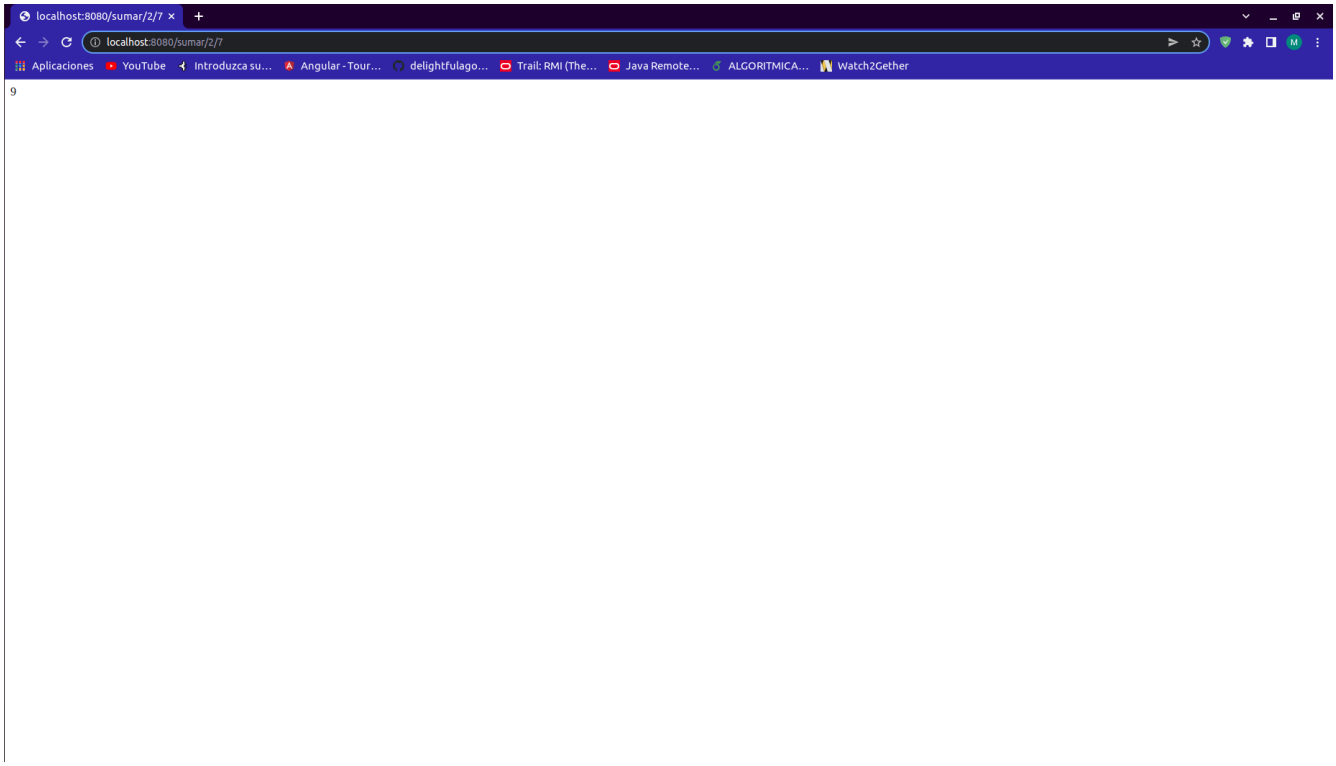
- Obtenemos el modulo "url", necesario para sacar la informacion de la url dada por el usuario, la cual seria aquella que no comprende la direccion donde esta el servidor.
- En la funcion del servidor, podemos destacar la linea 14 donde se obtiene la url dada por el usuario.
- En las lineas 16 a 22 se saca la informacion de la url, separando por "/" los elementos de la url, si no hay "/" en la url se devuelve un elemnto de una unidad, asi a la hora de comprobar la condicion del if dara fallo y saltara al mensaje de error del else. Si la url posee barras, se crea un array con los elementos que hay entre estas "/" (linea 17), tras esto se comprueba que haya 3 o

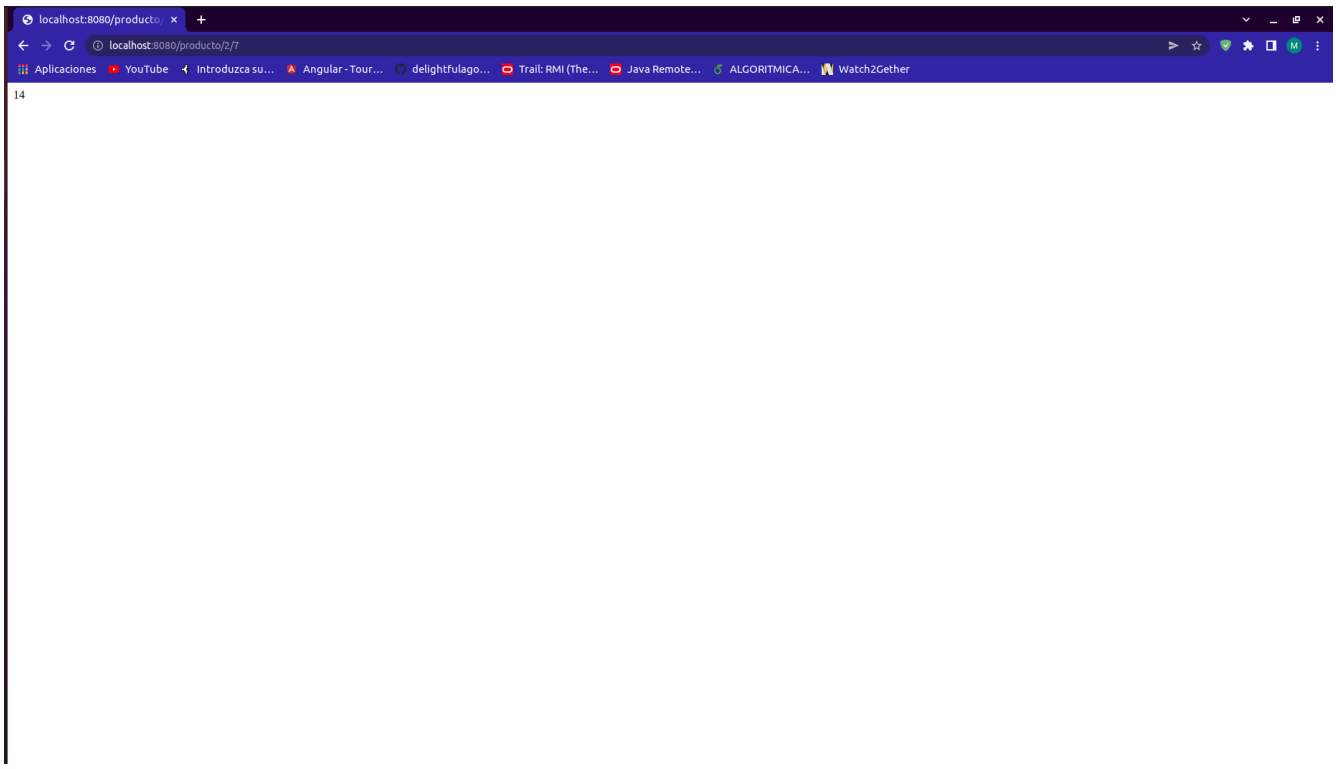
mas parametros de donde sacaremos el tipo de operacion (parametro 0), y los numeros de la operacion (parametros 1 y 2).

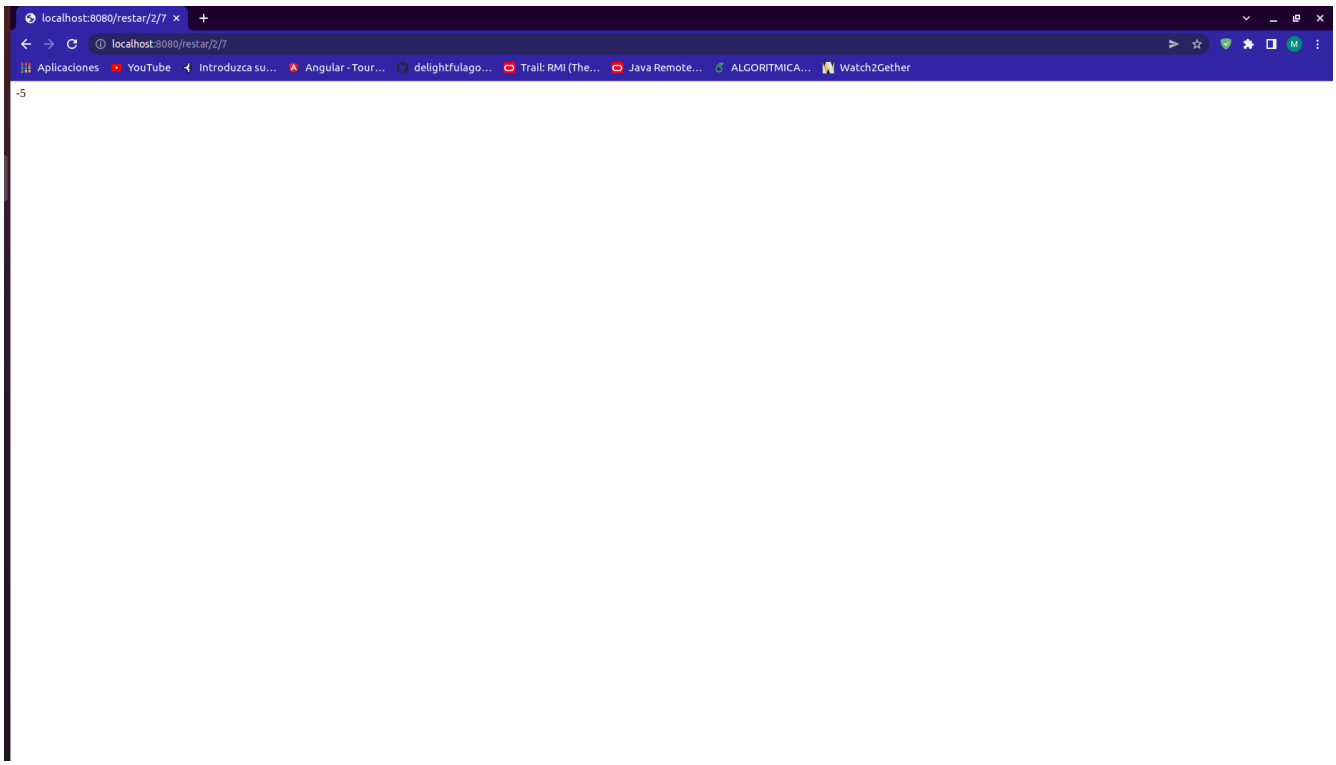
- Cuando tenemos los parametros llamamos a la funcion que calculara la operacion (funcion calcular). Esta funcion realiza la operacion (si es correcto el parametro, si no devuelve un error) y devuelve el resultado.

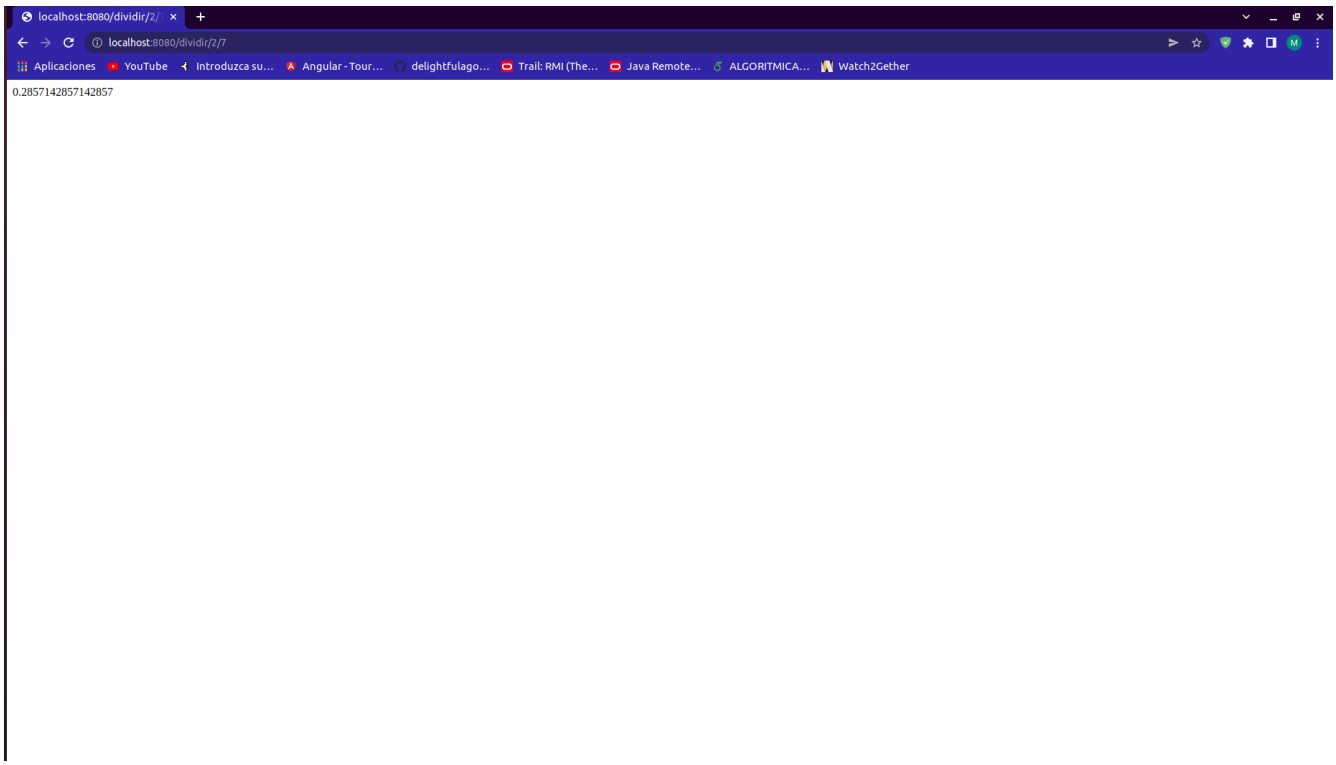
Este seria el resultado de la ejecucion del ejemplo en la terminal:

Este seria el resultado de las diferentes operaciones realizadas en el navegador:




localhost:8080/producto
localhost:8080/producto/2/7
Aplicaciones
YouTube
Introduzca su...
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
14


localhost:8080/restar/2/7
localhost:8080/restar/2/7
Aplicaciones
YouTube
Introduzca su...
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
-5



1.3. Ejercicio 3 - Calculadora con interfaz

Nuevamente este ejemplo es una versión mejorada de los anteriores, en concreto es una versión actualizada del ejemplo anterior, donde esta vez se añadirá una interfaz gráfica a la calculadora.

Las partes a destacar del nuevo código son las siguientes:

- Obtenemos el modulo "fs" el cual permite realiza operaciones de E/S sobre el sistema de ficheros del servidor.
- Obtenemos el modulo "path" el cual permite extraer información de una ruta a partir de cadenas de caracteres.
- En la linea 18 se añade el nombre del fichero que contiene el código de html de la interfaz. En este fichero podemos destacar la función "enviar", la cual recoge los valores introducidos por el usuario los manda al servidor por medio de una petición REST por medio del objeto creado de tipo XMLHttpRequest y espera la respuesta para mostrarla en pantalla con la funcion de la linea 30 la cual detecta cuando se recibe una respuesta y la muestra por pantalla.
- En la linea 19 utilizamos path para seleccionar el archivo html.
- En la linea 20 comprobamos si la llamada recibida se refiere a un archivo correcto en el sistema. Si es así se ejecutara el archivo en cuestión, en este caso un html, si se produce algún error se

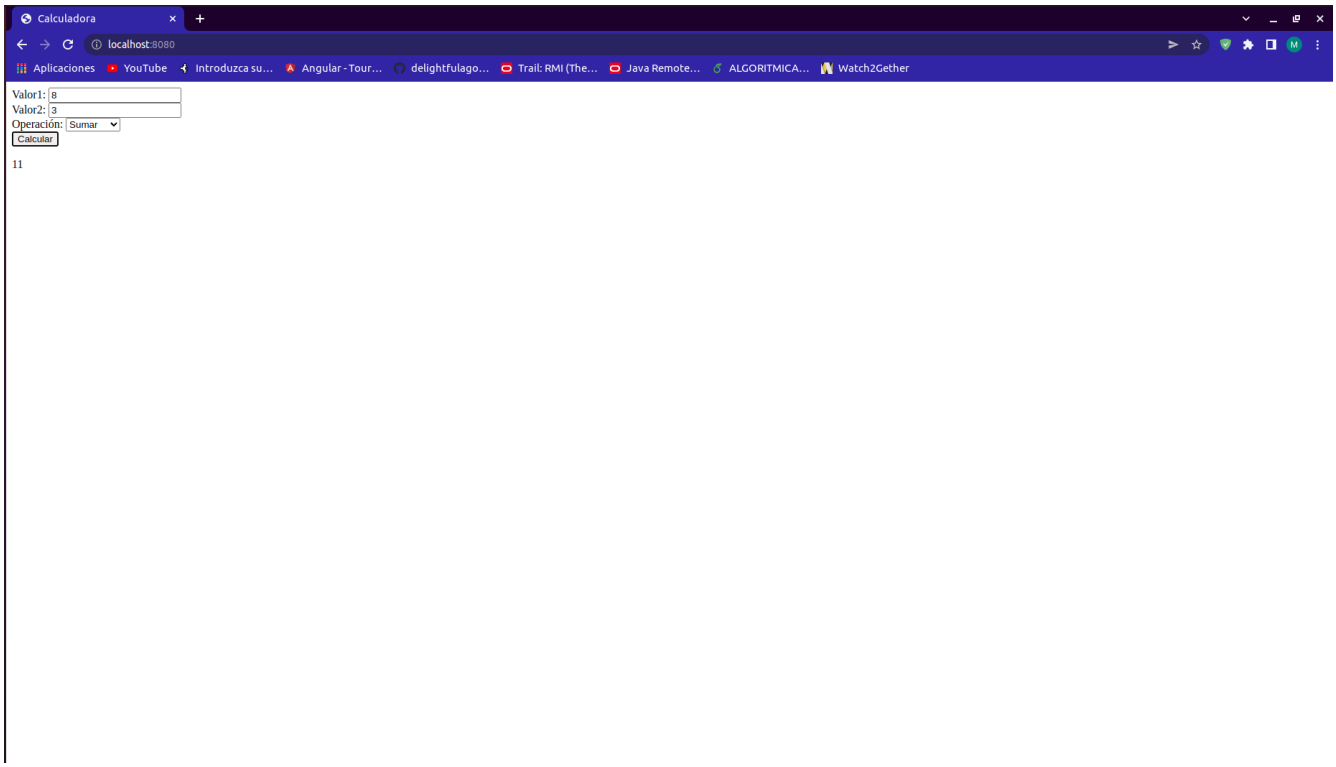
notificara esto. Si el fichero no existe se realizara lo mismo que en el ejemplo anterior.

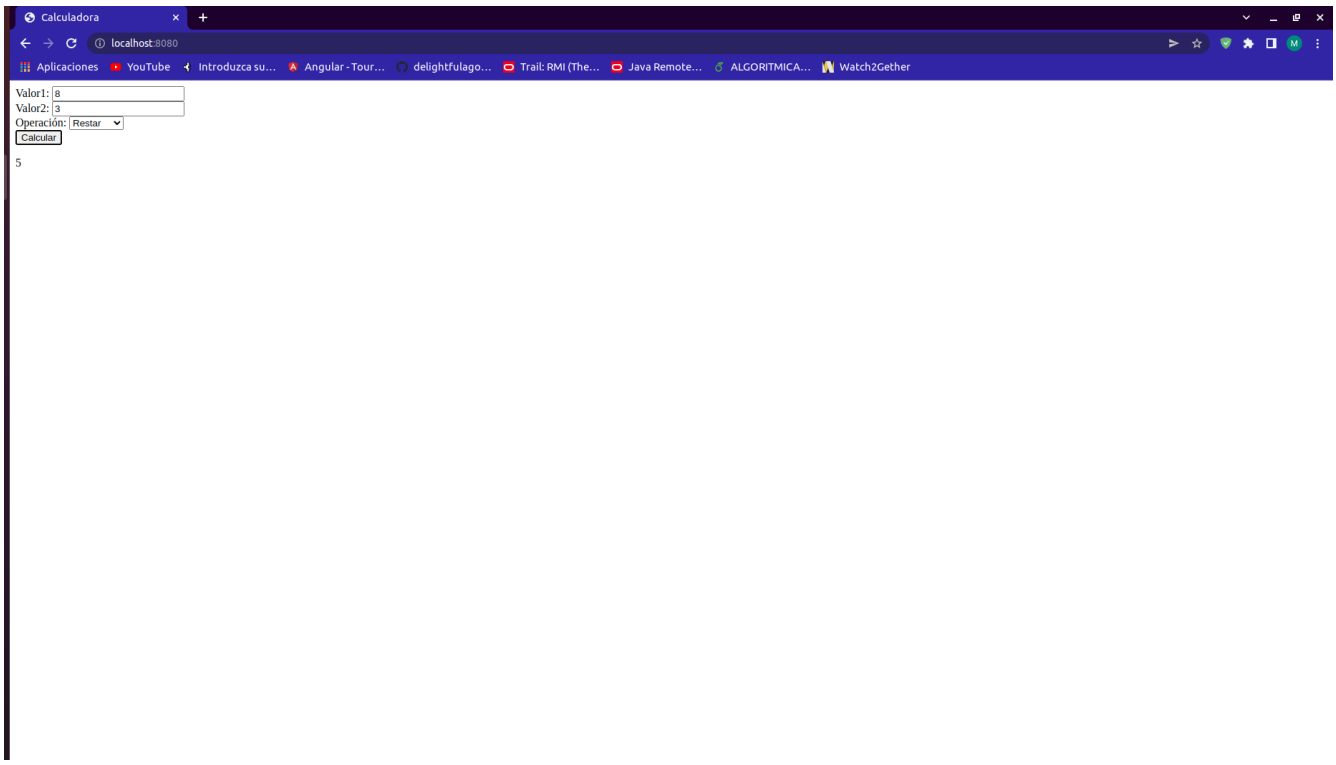
-

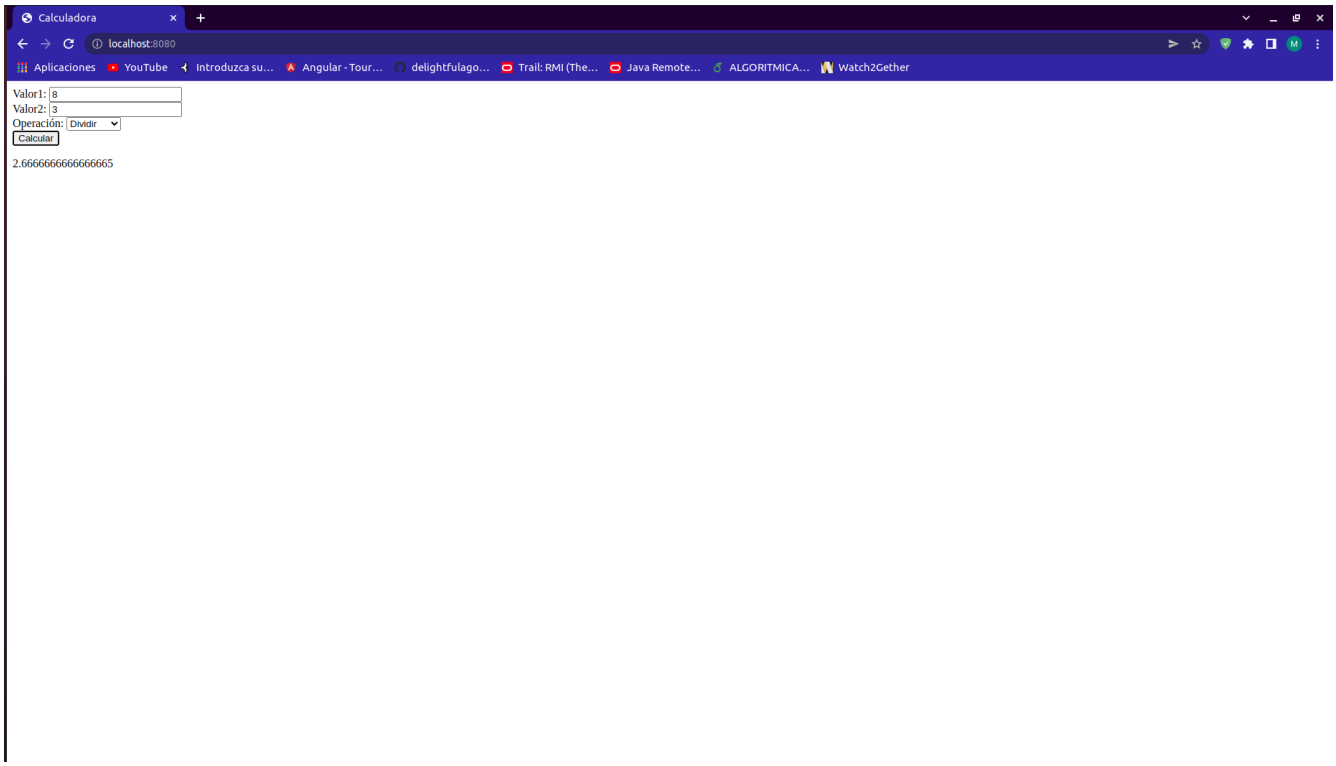
Este es el resultado de la ejecución en el terminal:

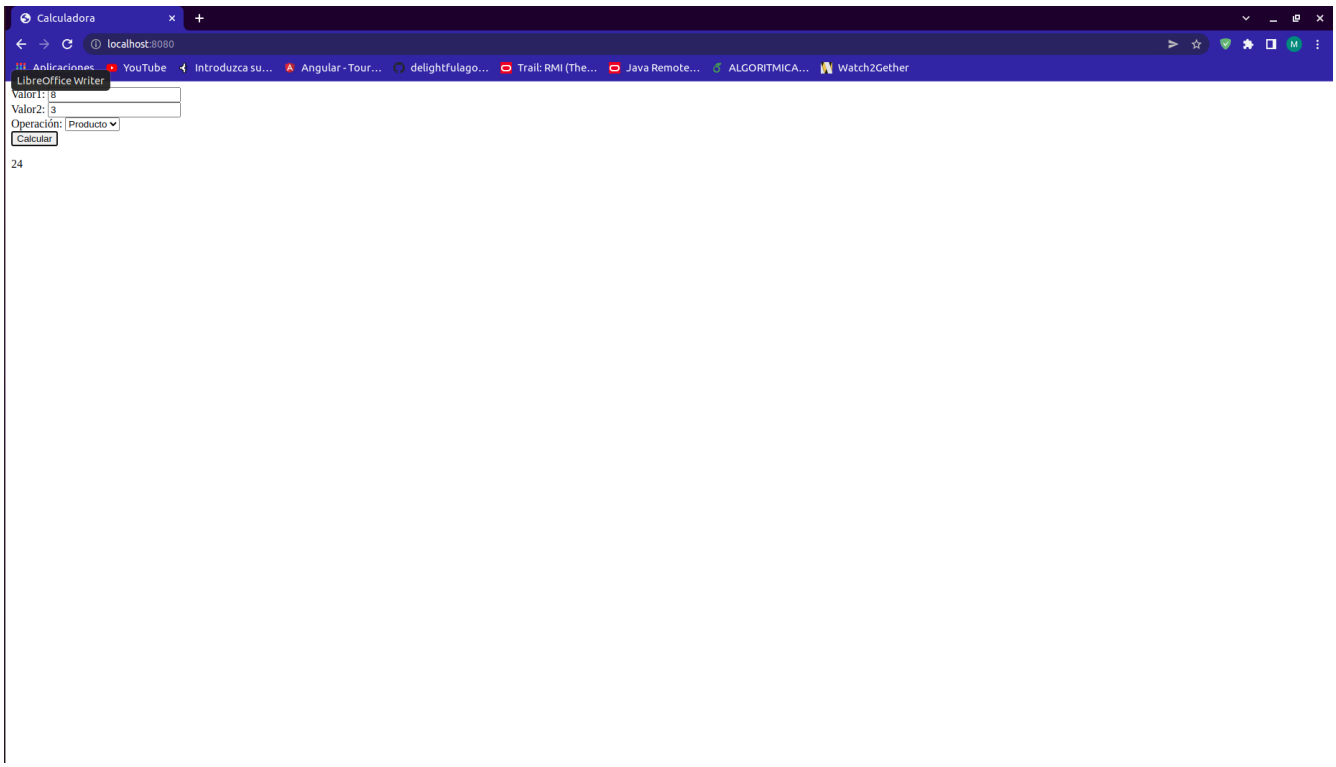
```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P4/ejemplos$ nodejs calculadora-web.js
Servicio HTTP iniciado
Peticion invalida: favicon.ico
Peticion REST: sumar/8/3
Peticion REST: restar/8/3
Peticion REST: dividir/8/3
Peticion REST: producto/8/3
```

Este seria el resultado en el navegador:




Calculadora
localhost:8080
Aplicaciones
YouTube
Introduzca su...
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
Valor1: 8
Valor2: 3
Operación: Restar
Calcular
5


Calculadora
localhost:8080
Aplicaciones
YouTube
Introduzca su...
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
Valor1: 8
Valor2: 3
Operación: Dividir
Calcular
2.6666666666666665



1.4. Ejercicio 4 - Connections

Este ejemplo utiliza el modulo socket.io para Node.js, que permite implementar aplicaciones web en tiempo real mediante WebSockets. El funcionamiento de este ejemplo es muy simple, a cada cliente que se conecta le dice que clientes estan conectados y cual es su IP y puerto correspondientes.

Las partes a destacar de este código serian las siguientes:

- En la linea 5 obtenemos el modulo "socket.io".
- En la linea 39 se llama a la funcion httpServer.
- En la linea 40 se crea el objeto "socket.io", este objeto registra las conexiones al servidor por parte de los clientes en las lineas 43 a 72. En la linea 45 se van guardando los diferentes clientes que van entrando en el servidor en un array, la cual luego se actualiza en la linea 47 donde se refresca la interfaz con los diferentes clientes. En las lineas 48-50 se muestra un mensaje al cliente que acaba de llegar al servidor. En las lineas 51-72 se avisa de que el cliente se acaba de desconectar, se elimina de la lista de clientes y se llama a todos los clientes para que actualicen su interfaz.

En el archivo "connections.html" podemos destacar lo siguiente:

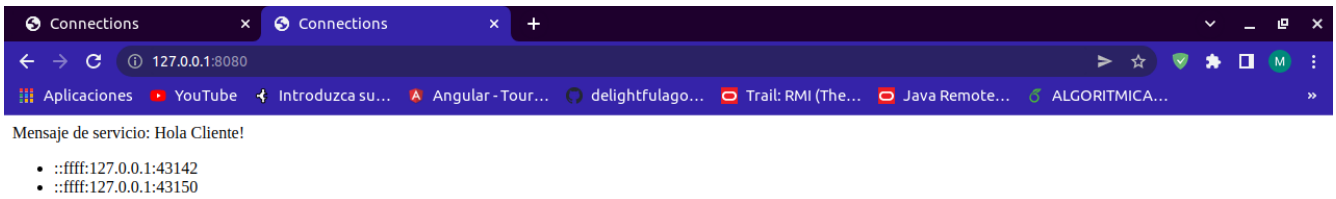
- La función "mostrar$_{m}ensaje"cambiaelcontenidodelcontenedorspan.La función"actualizarLista"actualiz$

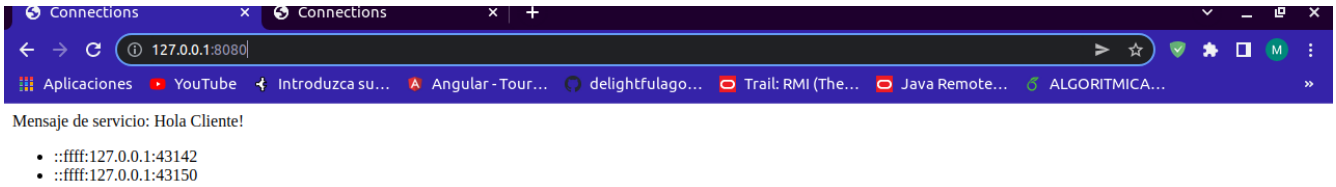
- Las cuatro funciones de las lineas 32 a 43 son funciones que hacen de intermediario entre el cliente y el servidor.

  Este seria el resultado de la ejecución en la terminal:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P4/ejemplos$ nodejs connections.js
Servicio Socket.io iniciado
Peticion invalida: /favicon.ico
New connection from ::ffff:127.0.0.1:43142
Peticion invalida: /favicon.ico
New connection from ::ffff:127.0.0.1:43150
El cliente ::ffff:127.0.0.1 se va a desconectar
[ { address: '::ffff:127.0.0.1', port: 43142 },
  { address: '::ffff:127.0.0.1', port: 43150 } ]
El usuario ::ffff:127.0.0.1 se ha desconectado
El cliente ::ffff:127.0.0.1 se va a desconectar
[ { address: '::ffff:127.0.0.1', port: 43142 } ]
El usuario ::ffff:127.0.0.1 se ha desconectado
```

  Este seria el resultado en el navegador:


Connections
Connections
127.0.0.1:8080
Aplicaciones
YouTube
Introduzca su...
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Mensaje de servicio: Hola Cliente!
::ffff:127.0.0.1:43142
::ffff:127.0.0.1:43150



1.5. Ejercicio 5 - MongoDB

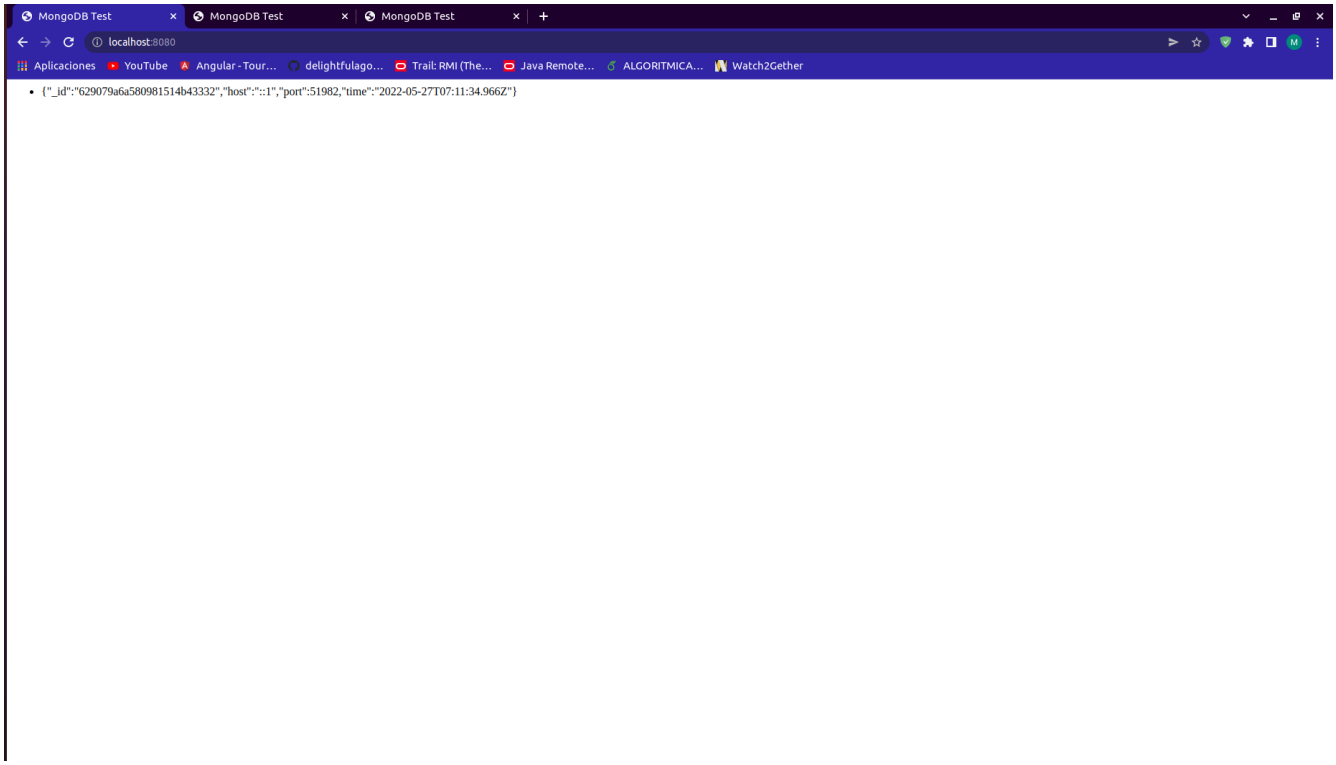
Este ejemplo muestra un servicio que recibe dos tipos de notificaciones mediante Socket.io: "poner" y "obtener". Podemos destacar de este código las lineas de código de la 45 a la 64:

- Lineas 45-47: conectamos la base de datos de mongo con el servidor.
- Lineas 49-64: creamos la base de datos.
- Lineas 52-60: creamos la función que va a realizarse cada vez que haya una llamada. Esta gracias al código html y al modulo socket, va a recibir la información de cuando un usuario se conecta y a que hora. Esta información se va a guardar en la BD con la llamada a "poner", y con la llamada a "obtener" se van a devolver todos los datos.

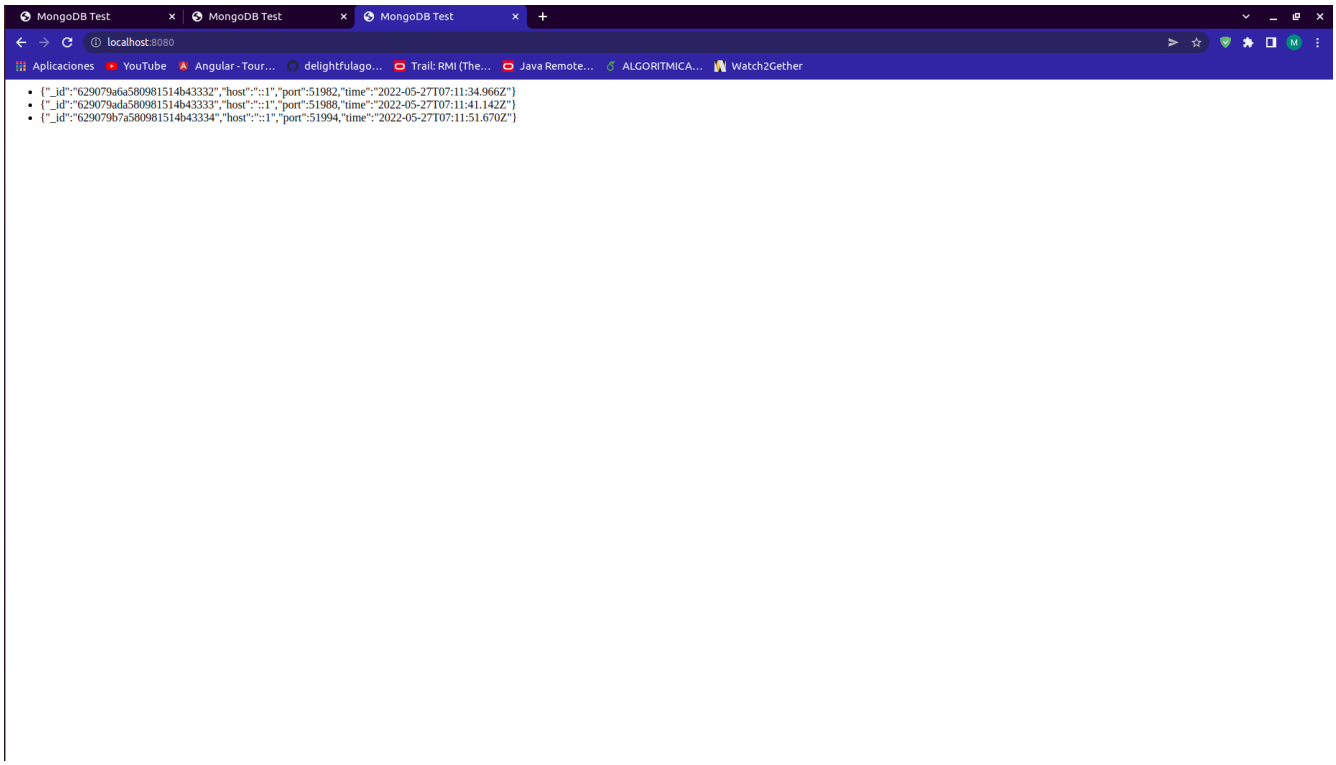
Este seria el resultado de la ejecución en la terminal:

```
manuel@manuel-ZenBook-UX425IA-UM425IA:~/Escritorio/DSD/P4/ejemplos$ node mongo-test.js
Servicio MongoDB iniciado
Peticion invalida: /favicon.ico
(node:8356) [MONGODB DRIVER] Warning: collection.insert is deprecated. Use insertOne, insertMany or bulkWrite instead.
(Use `node --trace-warnings ...` to show where the warning was created)
Peticion invalida: /favicon.ico
Peticion invalida: /favicon.ico
```

Este seria el resultado en el navegador web:




MongoDB Test
MongoDB Test
MongoDB Test
localhost:8080
Aplicaciones
YouTube
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
{"_id":"629079a6a580981514b43332","host":"::1","port":51982,"time":"2022-05-27T07:11:34.966Z"}
{"_id":"629079ada580981514b43333","host":"::1","port":51988,"time":"2022-05-27T07:11:41.142Z"}


MongoDB Test
MongoDB Test
MongoDB Test
localhost:8080
Aplicaciones
YouTube
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
{"_id":"629079a6a580981514b43332","host":"::1","port":51982,"time":"2022-05-27T07:11:34.966Z"}
{"_id":"629079ada580981514b43333","host":"::1","port":51988,"time":"2022-05-27T07:11:41.142Z"}
{"_id":"629079b7a580981514b43334","host":"::1","port":51994,"time":"2022-05-27T07:11:51.670Z"}

2. Parte 2

2.1. Ejecución del código.

Para la ejecución de este ejercicio se ha creado un simplemente se debe abrir una terminal, situarse en la carpeta que contiene los ficheros y ejecutar el comando "node servidor.js" tras esto el servidor se pondrá en funcionamiento. Hecho esto abriremos el navegador con las tres pestañas siguientes: http://localhost:8080/cliente.html http://localhost:8080/agente.html http://localhost:8080/sensores.html

2.2. Explicación del desarrollo.

Las tres pestañas que abriremos en el navegador tienen la siguiente funcionalidad:

- Cliente: corresponde al apartado del cliente, este se encarga de ver los valores tomados por los sensores y mostrarlos, además el cliente puede subir o bajar la persiana y apagar o encender el aire acondicionado.
- Agente: Esta pestaña no tiene interfaz, pero se encarga de controlar si los valores exceden un limite o no.
- Sensores: esta pestaña nos permite introducir los valores que queramos en los sensores los cuales se alteraran en el servidor.

2.3. Ficheros.

2.3.1. Agente.html:

Aquí se realizan las operaciones de consulta de los datos introducidos por los sensores y comprueba si son o no correctos. Las funciones de este fichero serian las siguientes:

- **usarPersiana()**: Esta función simplemente llama a la función 'usarPersiana' que se encarga de avisar al servidor de que se va a usar la persiana.
- **getEstadoPersiana**: Esta función le pide al servidor el valor actual del estado de la persiana y lo guarda.
- **actualizarLuz**: Esta función analiza si el valor de la luz es el correcto, si no es así llama a la función 'luzAlert' y a la función 'usarPersiana()' para corregir el valor. La función 'luzAlert' se encarga de enviar el aviso de que la luz se ha salido del rango y la corrige.

- **usarAC()**: Esta función simplemente llama a la función 'usarAC' que se encarga de avisar al servidor de que se va a usar el aire acondicionado.
- **getEstadoAC**: Esta función le pide al servidor el valor actual del estado del aire acondicionado y lo guarda.
- **actualizarTemp**: Esta función analiza si el valor de la temperatura es el correcto, si no es así llama a la funcion 'tempAlert' y a la función 'usarAC()' para corregir el valor. La función 'tempAlert' se encarga de enviar el aviso de que la temperatura se ha salido del rango y la corrige.

2.3.2. Sensores.html:

En esta interfaz podremos cambiar los valores de la temperatura y la luz.

- **enviar()**: Esta función recoge los valores introducidos por el usuario y los manda al servidor por medio de la orden 'poner', que se encargara de introducirlos en el servidor.

2.3.3. Cliente.html:

El cliente nos permite cambiar los estados de la persiana y del aire acondicionado, además de consultar los valores de estos y de la luz y temperatura.

- **actualizar**: Actualiza los diferentes datos que ve el cliente.
- **setEstadoPersiana**: Establece el valor del estado de la persiana.
- **updateEstadoPersiana**: Cuando se pulsa el botón que cambia el estado de la persiana esta función avisa al servidor de dicho cambio.
- **updateAlertLuz**: cambiar el mensaje de alerta de la luz, avisando de que se ha salido el valor del rango permitido.
- **setEstadoAC**: Establece el valor del estado del aire acondicionado.
- **updateEstadoAC**: Cuando se pulsa el botón que cambia el estado del aire acondicionado esta función avisa al servidor de dicho cambio.
- **updateAlertTemp**: cambiar el mensaje de alerta de la temperatura, avisando de que se ha salido el valor del rango permitido.

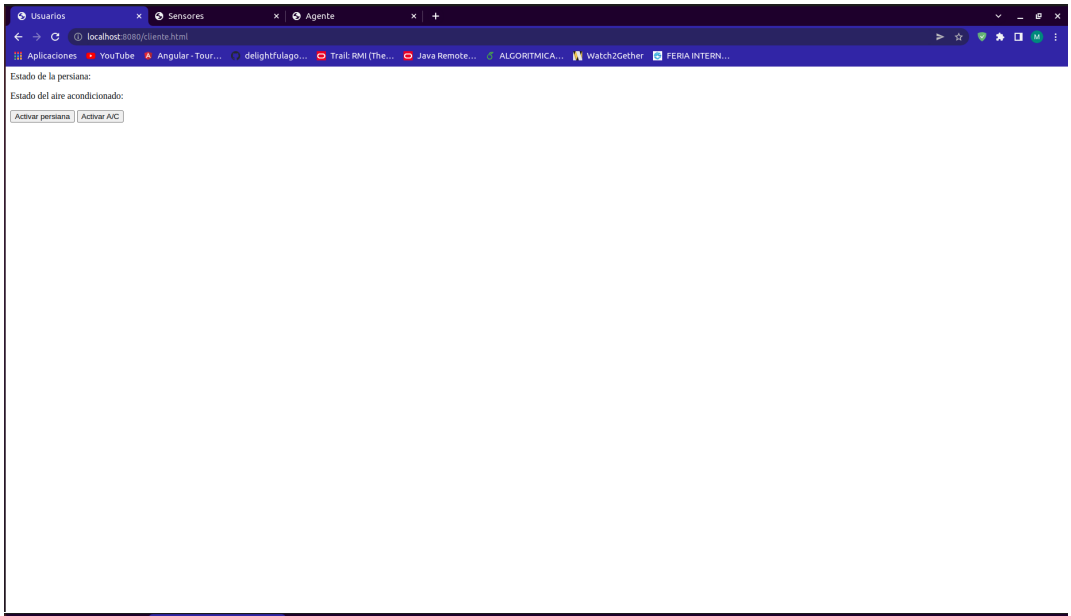
2.3.4. Servidor.js:

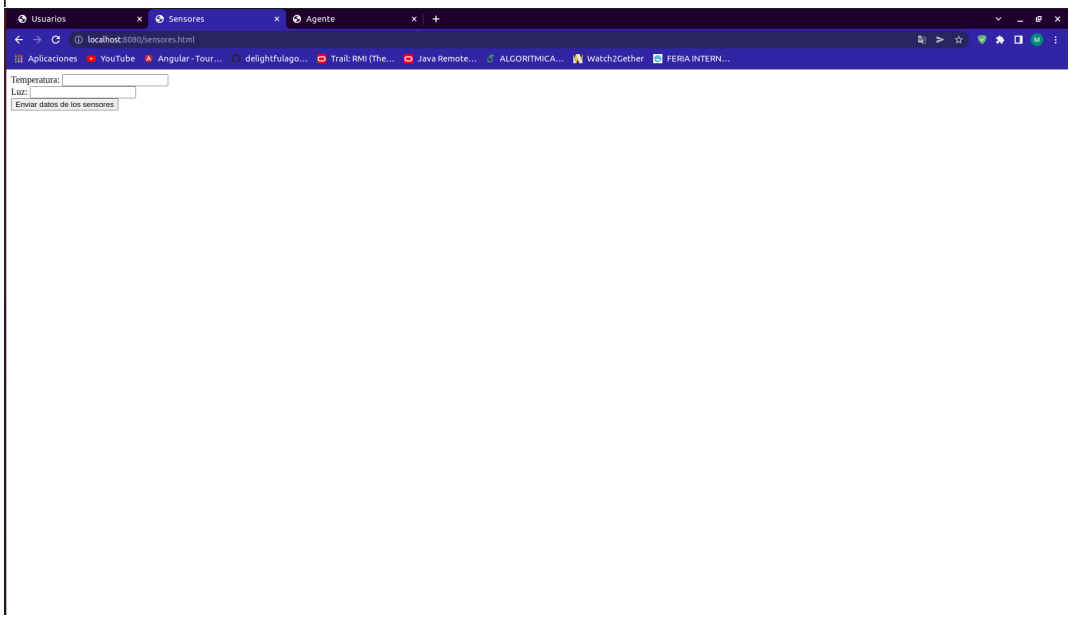
Este es el servidor que va a permitir la comunicación entre los anteriores componentes del sistema domótico, además de esto implementa las funciones antes descritas para que realicen lo que se ha

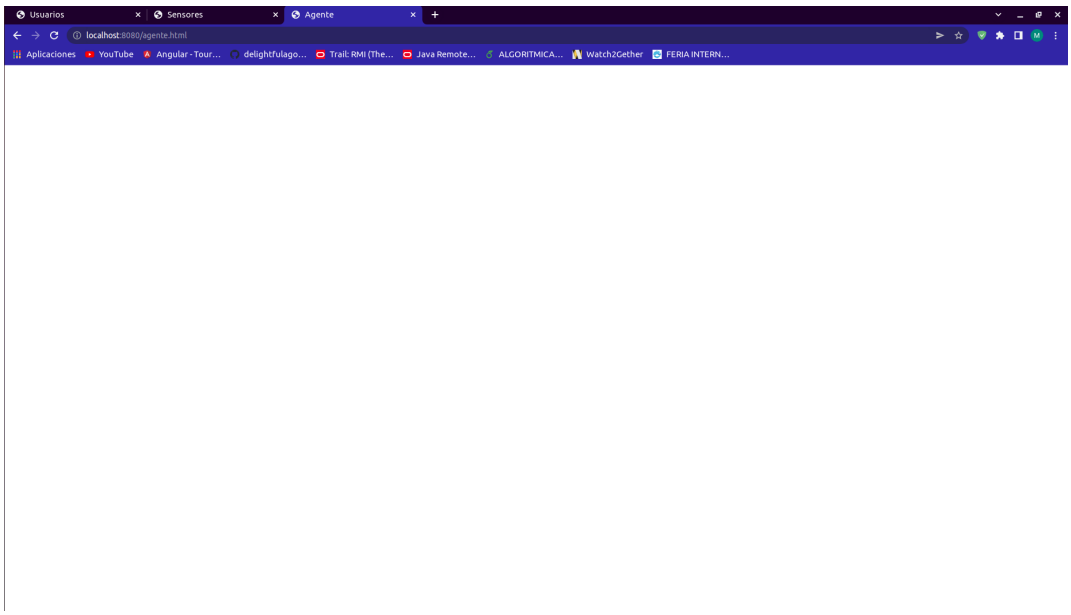
explicado.

2.4. Ejemplo de uso.

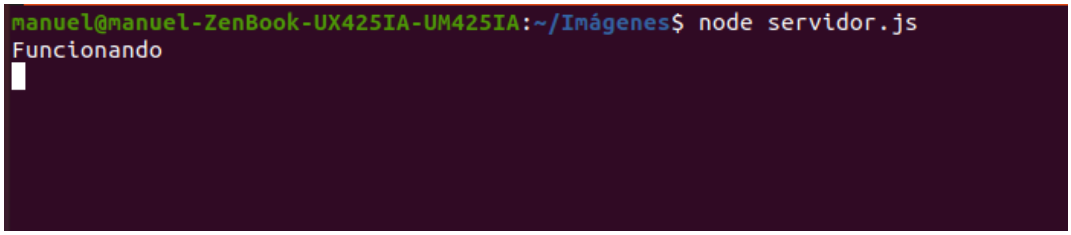
Vamos a seguir la ejecución de un ejemplo, para ello seguiremos los pasos descritos anteriormente, por lo cual al comienzo tendremos las siguientes pestañas:



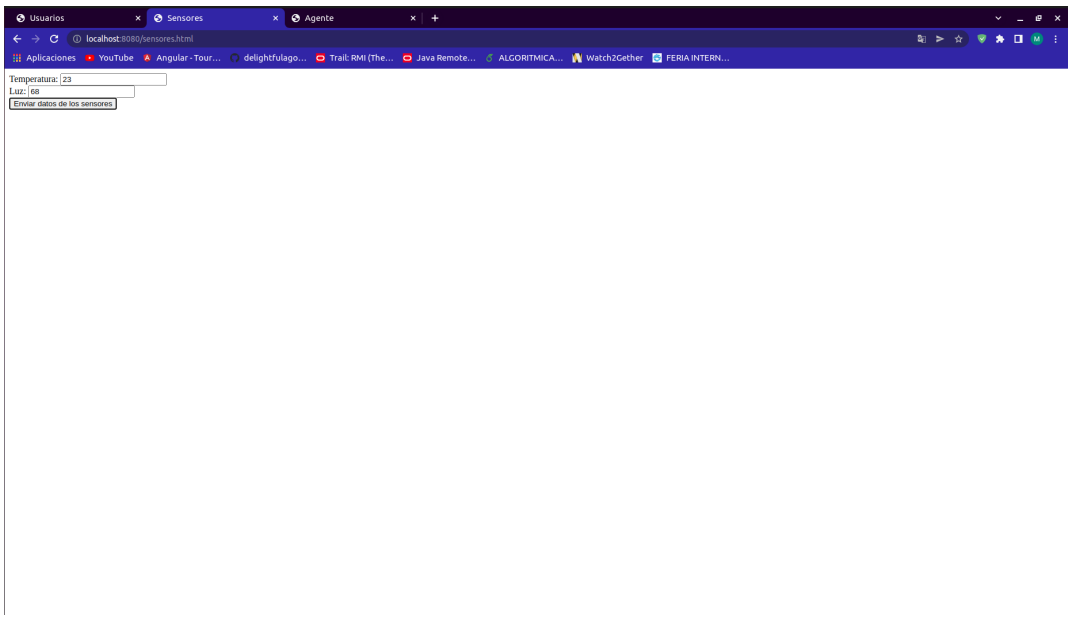




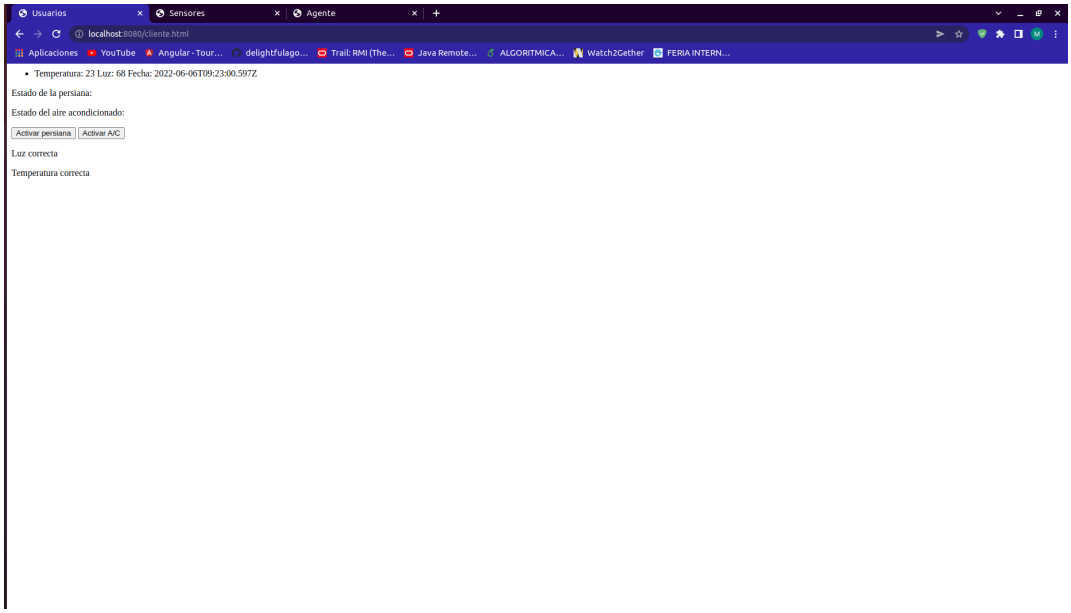
Y el siguiente estado en la terminal:



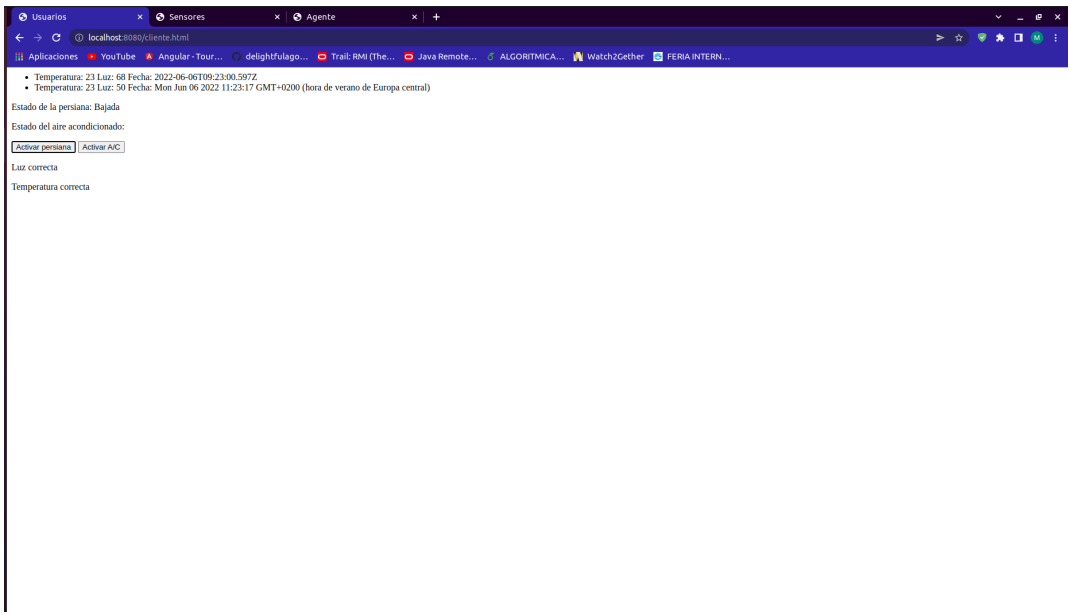
Ahora introduciremos en la pestaña de sensores los siguientes valores:



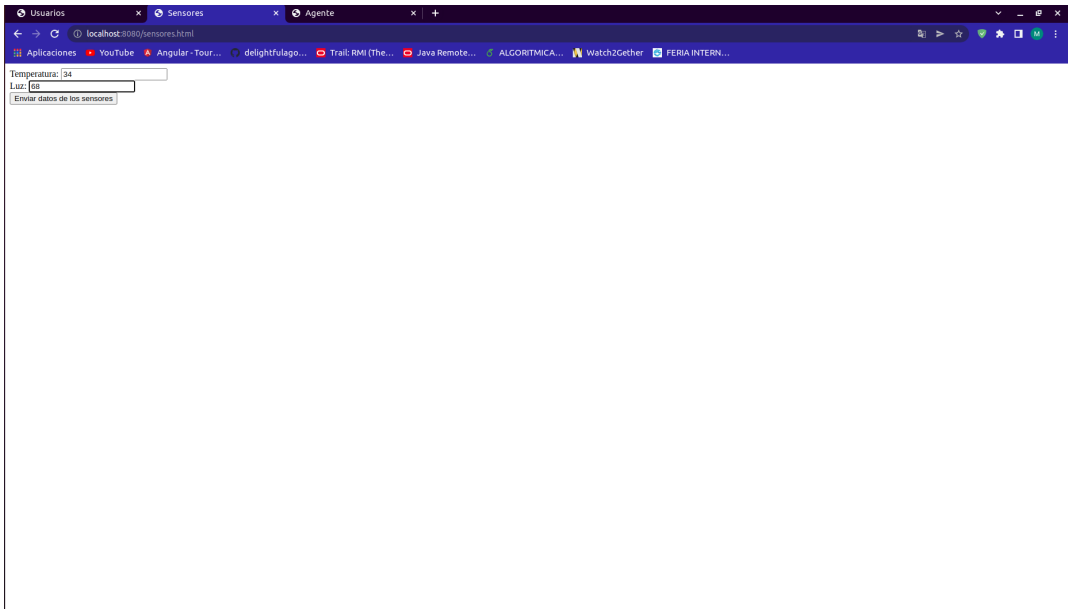
Podemos ver reflejados dichos valores en la pestaña del cliente de la siguiente manera:



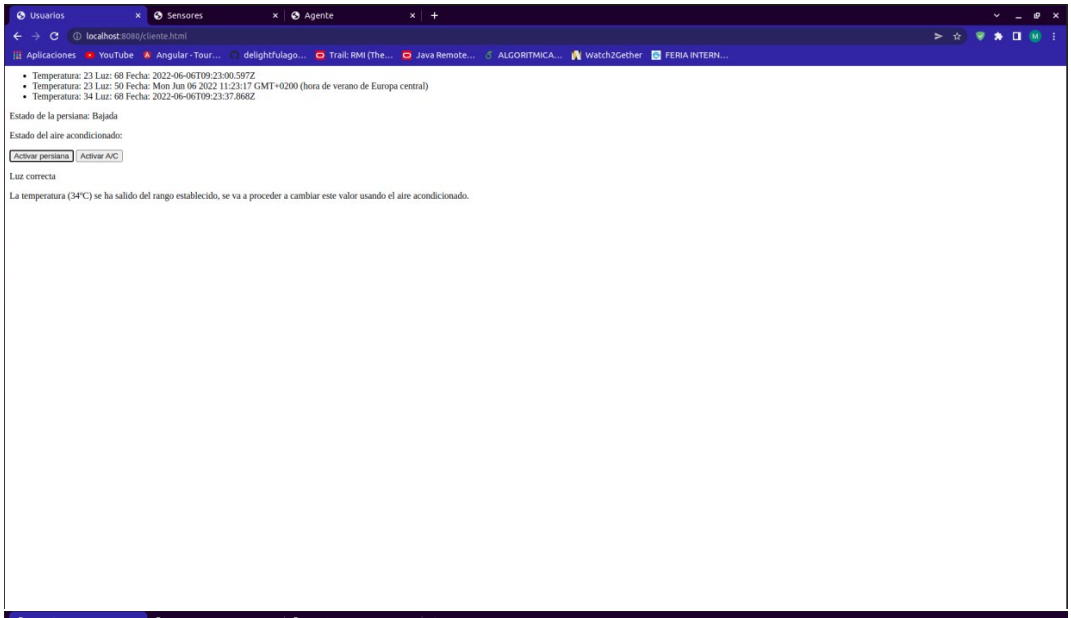
Ahora pasamos a bajar la persiana para bajar así el valor de la luz:

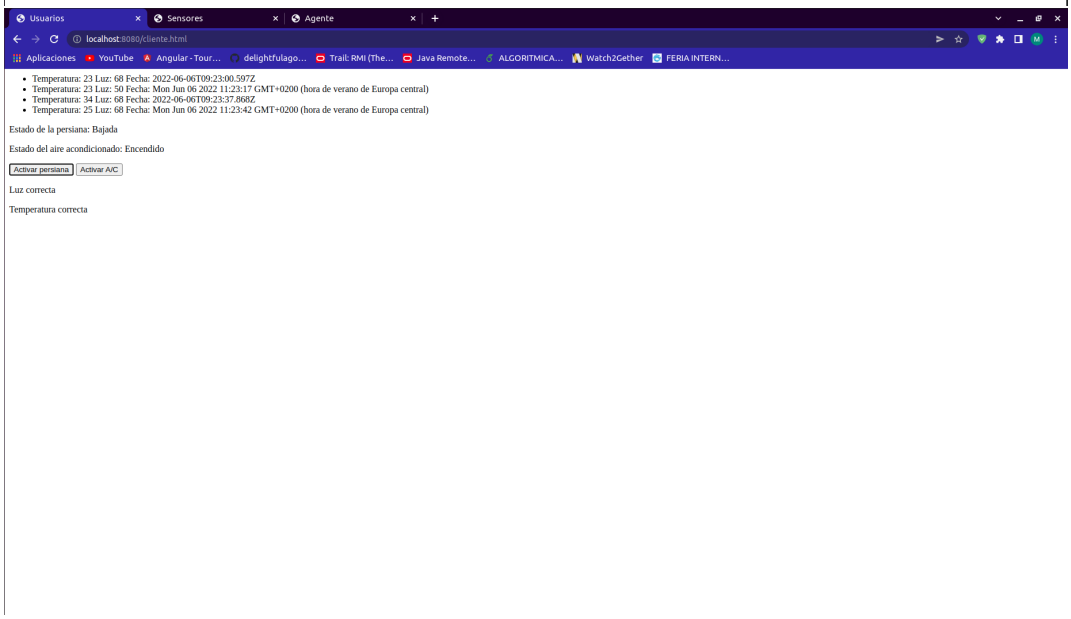


Tras esto vamos a introducir valores que se escapan del rango para que salte el agente:



Ahora en el cliente podemos ver los anuncios hechos:


Usuarios
Sensores
Agente
localhost:8080/cliente.html
Aplicaciones
YouTube
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
FERIA INTERN...
Temperatura: 23 Luz: 68 Fecha: 2022-06-06T09:23:00.597Z
Temperatura: 23 Luz: 50 Fecha: Mon Jun 06 2022 11:23:17 GMT+0200 (hora de verano de Europa central)
Temperatura: 34 Luz: 68 Fecha: 2022-06-06T09:23:37.868Z
Estado de la persiana: Bajada
Estado del aire acondicionado:
Activar persiana
Activar A/C
Luz correcta
La temperatura (34ºC) se ha salido del rango establecido, se va a proceder a cambiar este valor usando el aire acondicionado.


Usuarios
Sensores
Agente
localhost:8080/cliente.html
Aplicaciones
YouTube
Angular - Tour...
delightfulago...
Trail: RMI (The...
Java Remote...
ALGORITMICA...
Watch2Gether
FERIA INTERN...
Temperatura: 23 Luz: 68 Fecha: 2022-06-06T09:23:00.597Z
Temperatura: 23 Luz: 50 Fecha: Mon Jun 06 2022 11:23:17 GMT+0200 (hora de verano de Europa central)
Temperatura: 34 Luz: 68 Fecha: 2022-06-06T09:23:37.868Z
Temperatura: 25 Luz: 68 Fecha: Mon Jun 06 2022 11:23:42 GMT+0200 (hora de verano de Europa central)
Estado de la persiana: Bajada
Estado del aire acondicionado: Encendido
Activar persiana
Activar A/C
Luz correcta
Temperatura correcta