Desarrollo de Sistemas Distribuidos

- Práctica 4. NodeJS -

Índice

Ignacio Yuste López. 3ºA1.

54141533Q

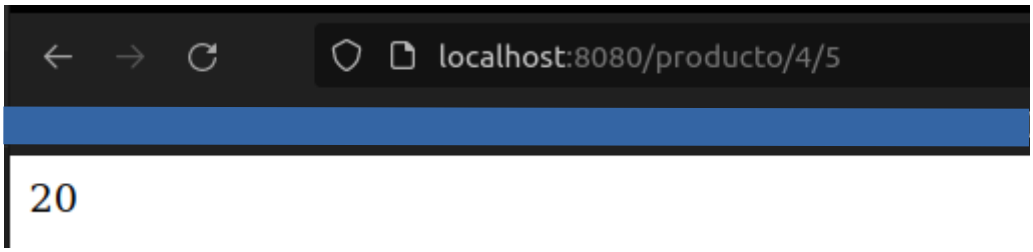
1. Ejecución de ejemplos.

1.1. Ejemplo 1 – Calculadora simple:

Este ejemplo consiste en una calculadora distribuida usando una interfaz de tipo REST. Las peticiones han de escribirse en la propia URL ya que no tiene ningún tipo de interfaz.

Consta de una función calcular a la cual se le pasa la operación a realizar junto con los parámetros de la misma. Para ello es necesario separar la entrada, lo cual hacemos en la función de petición al servidor.

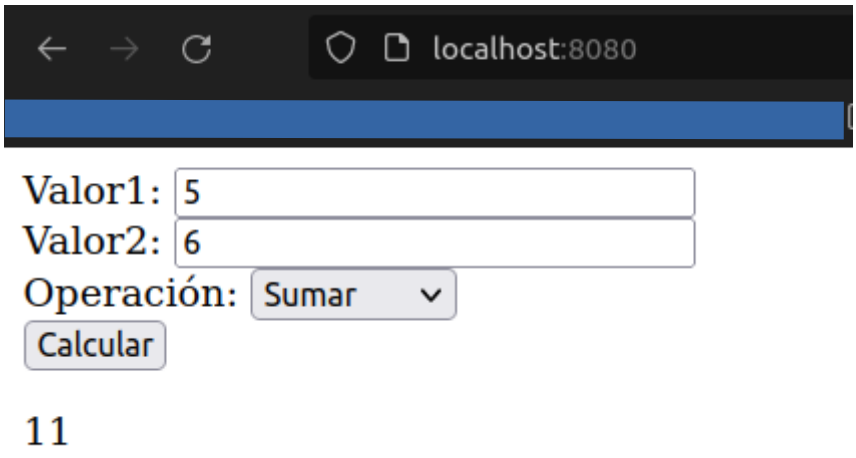
Cuerpo de texto



1.2. Ejemplo 2 – Calculadora con interfaz:

La diferencia de este ejemplo con el anterior es la implementación de una interfaz para usar la calculadora.

Para ello se hace uso de un archivo *html* que implementa un formulario, a través del cual podremos usar la calculadora. Este archivo incluye un script a través del cual hacemos las peticiones REST, de manera que el funcionamiento real de la calculadora es muy similar. Realmente, si escribimos la petición en la URL de la misma manera que antes el funcionamiento será igual.

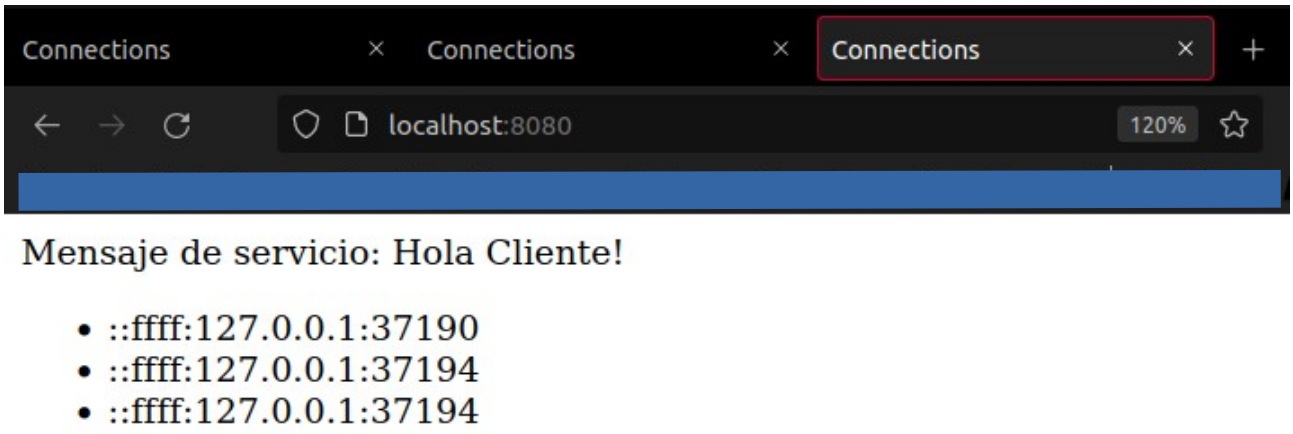


1.3. Ejemplo 3 – Conexiones SocketIO:

Este ejemplo sirve para demostrar cómo SocketIO puede enviar mensajes sin necesidad de que un cliente se lo solicite a través del uso de eventos.

En concreto, cuando un usuario se conecta el servidor envía a todos los usuarios conectados todas las IPs conectadas al mismo. Igual cuando un usuario se desconecta, su IP desaparece de la información del resto de cliente.

Para ello, el servidor emite y recibe varios eventos con los cuales gestiona el número de usuarios y se comunica con la interfaz para actualizar la información. De esta forma, si una de las dos partes llama a un evento de tipo “emit”, el mismo evento de tipo “on”, esté donde esté, se ejecutará.

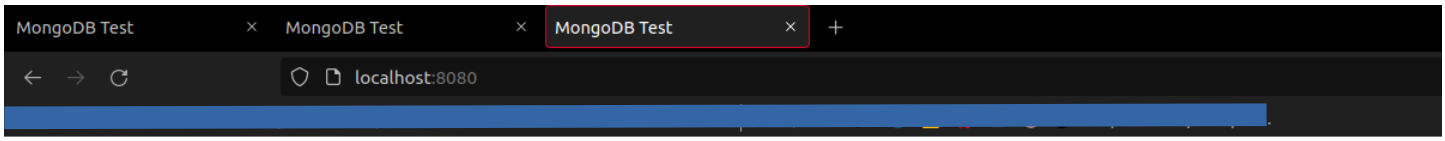


1.4.Ejemplo 4 – MongoDB:

Este ejemplo demuestra cómo podemos usar MongoDB desde NodeJS de una manera muy sencilla usando el módulo correspondiente. El ejemplo muestra la implementación de un servicio que recibe dos tipos de notificaciones: “poner” y “obtener”.

- “poner”: cuando el servidor recibe esta notificación introduce la información recibida en la base de datos
- “obtener”: cuando recibe esta notificación, hace una consulta a la base de datos y devuelve el contenido al cliente.

Por último, cuando un cliente se conecta, el servicio le devuelve su dirección de conexión.



- {"_id":"628c917a6aa1089750f25128","host":"::ffff:127.0.0.1","port":37294,"time":"2022-05-24T08:04:10.425Z"}
- {"_id":"628c91816aa1089750f25129","host":"::ffff:127.0.0.1","port":37302,"time":"2022-05-24T08:04:17.480Z"}
- {"_id":"628c91836aa1089750f2512a","host":"::ffff:127.0.0.1","port":37302,"time":"2022-05-24T08:04:19.428Z"}

2. Ejercicio – Implementación de sistema domótico con NodeJS.

Implementación de un sistema domótico a través de un servidor implementado con NodeJS. Incluye una interfaz de usuario web a través de la cual se puede modificar el estado de los distintos actuadores y modificar el valor de los sensores.

2.1. Sensores y actuadores.

Lista de sensores y actuadores implementados.

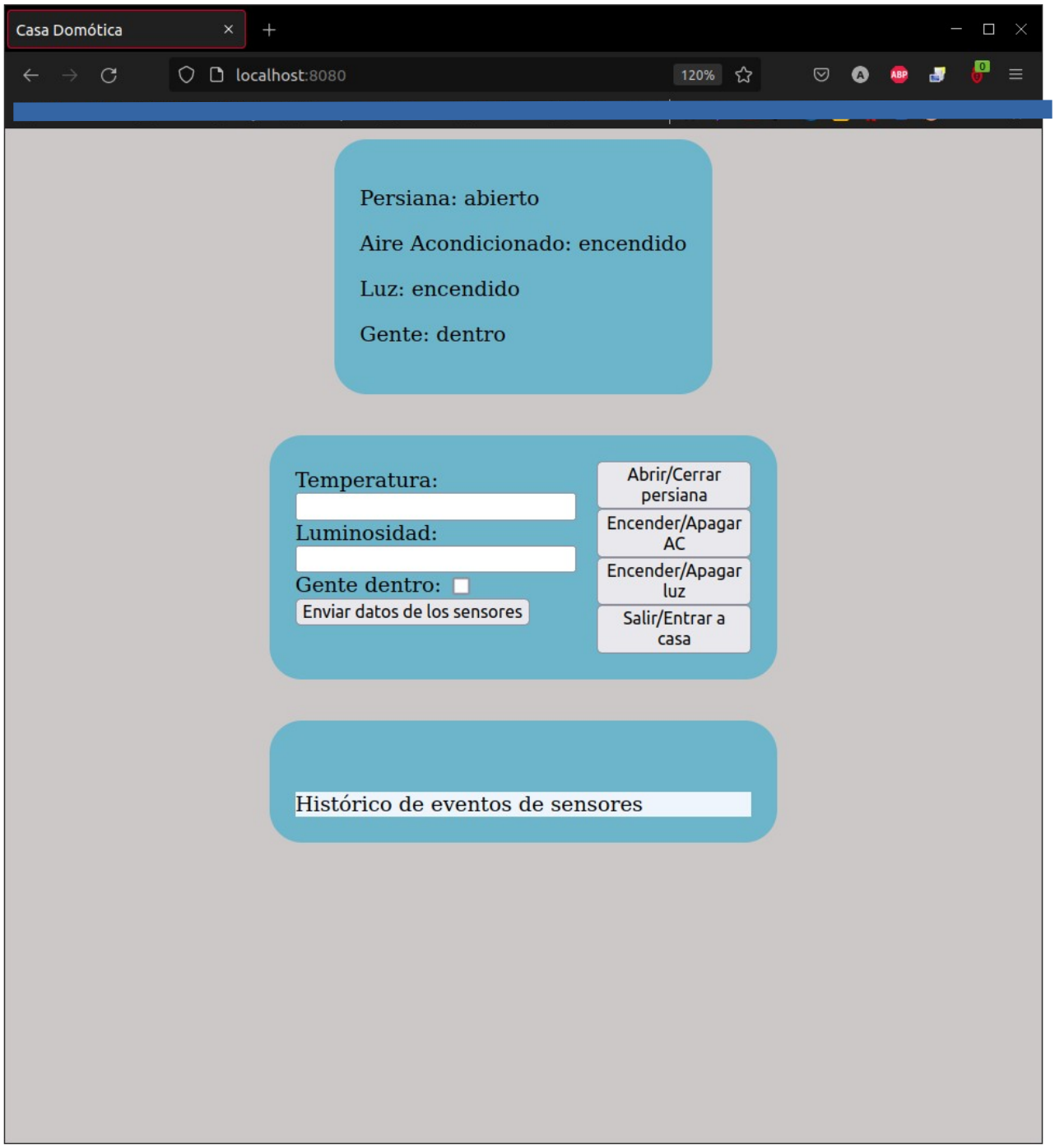
- **Sensores:**
  - *Temperatura*: Sensor de temperatura. Dispara dos alerta, una por temperatura demasiado alta y otra por temperatura demasiado baja.
  - *Luminosidad*: Sensor de luz. Dispara dos alerta, una por luminosidad demasiado alta y otra por luminosidad demasiado baja.
  - *Gente dentro:* Detecta si hay gente en la vivienda. Dispara una alerta cada vez que alguien entra o sale de casa.
- **Actuadores:**
  - Persiana: Motor que abre y cierra la persiana
  - Aire acondicionado: Enciende y apaga el AC
  - Luz: Enciende y apaga las luces
  - Puerta: No es realmente un actuador pero cambia el estado de la casa. Indicar que hay gente dentro de casa o no.

2.2. Interfaz de usuario.

La interfaz de usuario consiste en un cliente web el cual muestra:

- Estado de los actuadores.
- Formulario para establecer el valor de los sensores
- Botones para cambiar manualmente el estado de los actuadores.
- Alertas de temperatura y luminosidad
- Histórico de eventos de sensores.

A través de esta interfaz, el cliente puede ver el estado de la casa y cambiar manualmente el mismo. Los sensores son simulados y por tanto puede introducir los valores de los mismo manualmente. También puede consultar todos los cambios que han generado los sensores, incluyendo los valores obtenidos en cada cambio.

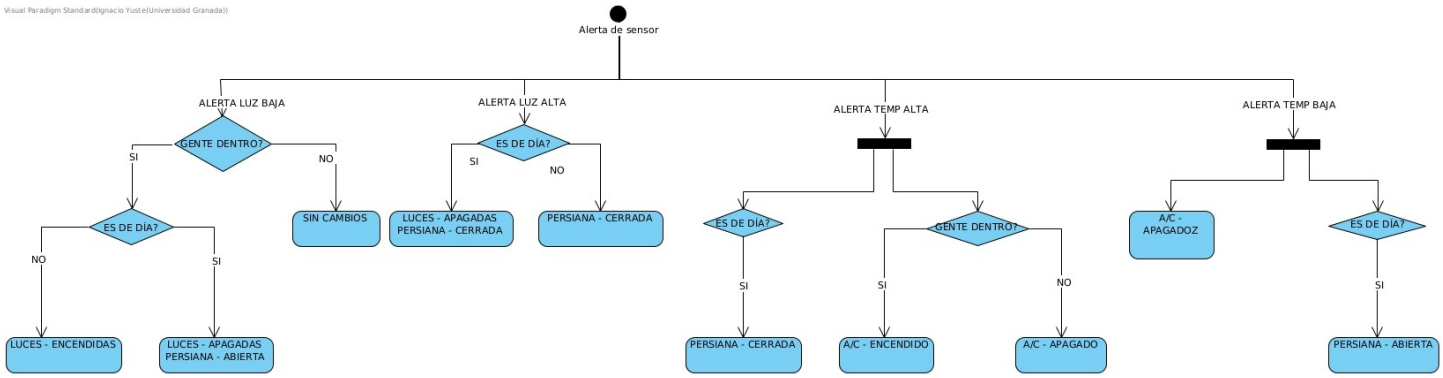


Se ha usado CSS para maquetar de manera sencilla la interfaz, dotándola de más claridad ya que agrupa por bloques los distintos elementos de la misma.

2.3. Diagrama de estados.

El servidor incorpora eventos con una complejidad un poco más elevada. A la hora de cambiar el estado de la casa, este tiene en cuenta si hay gente dentro de la casa, si es de día o de noche, etc.

Para explicarlo de manera más clara muestro un diagrama de estados:



(Imagen adjuntada en archivo de entrega para poder leer con mayor claridad)

En resumen, el sistema tiene en cuenta si hay gente dentro de la casa y si es de día o no para tomar decisiones. Por ejemplo, si la temperatura es muy alta pero no hay nadie en la casa no se enciende el aire acondicionado, pero sí se cierran las persianas para intentar bajar la temperatura. O, si detecta que hay mucha luz pero no hay nadie en la casa no se encienden las luces.

En el diagrama he usado un elemento “fork” para las alertas de temperatura. Esto no quiere decir que se ejecutan dos hebras en paralelo, si no que se ejecutan ambas partes del “fork” en serie, de izquierda a derecha. Por tanto, el estado final puede ser una combinación de ambos lados del “fork”.

Todos estas acciones se dan con los eventos generados por los sensores, ya que si el usuario cambia manualmente un actuador el sistema no debe volver a cambiarlo instantáneamente, solo lo hará si ocurre otro evento de sensor.

Para comprobar si es de día o no el servidor incluye un par de variables las cuales establecen que la noche empieza a las 20:00 y termina a las 7:00. De esta manera, y con el uso de la hora actual, el sistema hace un mejor uso de la electricidad.

2.4. Métodos Cliente-Servidor

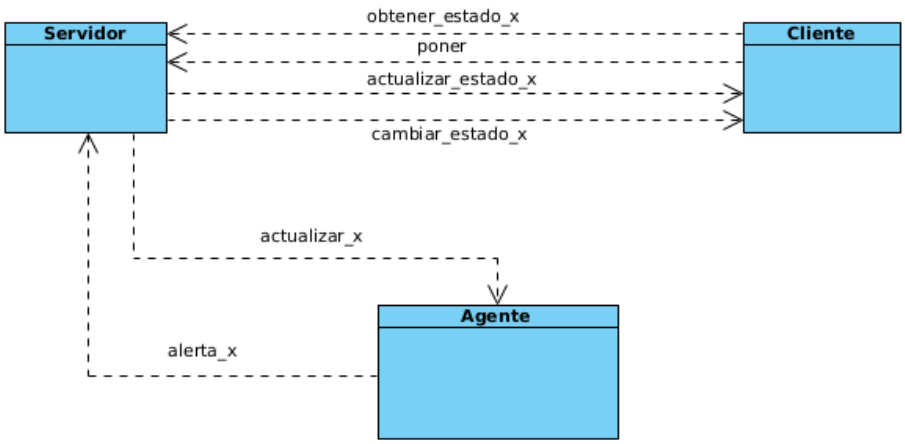
Al igual que en el ejemplo de MongoDB, en primer lugar tenemos dos notificaciones 'poner' y 'obtener'. Su cometido es similar al ejemplo, 'poner' introduce información en la base de datos y 'obtener' la devuelve. Sin embargo, en este caso la información introducida consiste en los valores de los sensores más la hora actual del sistema. Además, 'poner' invoca tres notificaciones que sirven para invocar la actualización de la información del cliente sobre la temperatura, la luz y si hay gente.

Podemos agrupar el resto de notificaciones de ambas partes en dos grupos: notificaciones de acción manual y notificaciones de acción por eventos de sensor.

Los eventos manuales son 'obtener_estado_x' y 'cambiar_estado_x', donde las x son los estados de los distintos actuadores. De esta manera, el cliente puede comprobar el estado de, por ejemplo, la persiana desde la interfaz mediante 'obtener_estado_persiana' y cambiar su estado manualmente con el botón, que emite la notificación 'cambiar_estado_persiana'. Estas notificaciones también invocan las notificaciones para actualizar la información en el cliente.

Por otro lado están los sensores. Ya que estos funcionan de manera "autónoma" es necesario implementar un tercer individuo llamado "agente". Este agente, el cual es ejecutado por el cliente, se encarga de estar atento a si los valores de los sensores pasan de los valores máximo y mínimos fijados para las alertas. Si se da el caso, el agente avisará al servidor de que hay una alerta de un sensor y el servidor invocará las notificaciones necesarias para cambiar los estados de los actuadores. Esta posibilidad de estados está descrita en el apartado anterior. Para esto, el servidor dispone de los métodos 'alerta_x' para recibir las alertas; el agente dispone de los métodos 'actualizar_x', los cuales se llaman cada vez que los sensores emiten datos, invocan las alertas si procede; el cliente tiene el método enviar_sensores(), el cual introduce la información en la base de datos mediante 'poner', que a si vez llama a los métodos del agente.

A continuación incluyo un breve diagrama para resumir la situación:



Para la cooperación el sistema usa la técnica de actualización. El agente se encarga de almacenar los valores de los sensores, además conocer los límites que disparan las alertas. Todos los cambios en los sensores pasan por él y actualizan estos valores, llamando al servidor y por tanto indicando si es necesario hacer algún cambio en los actuadores.