

Práctica 4

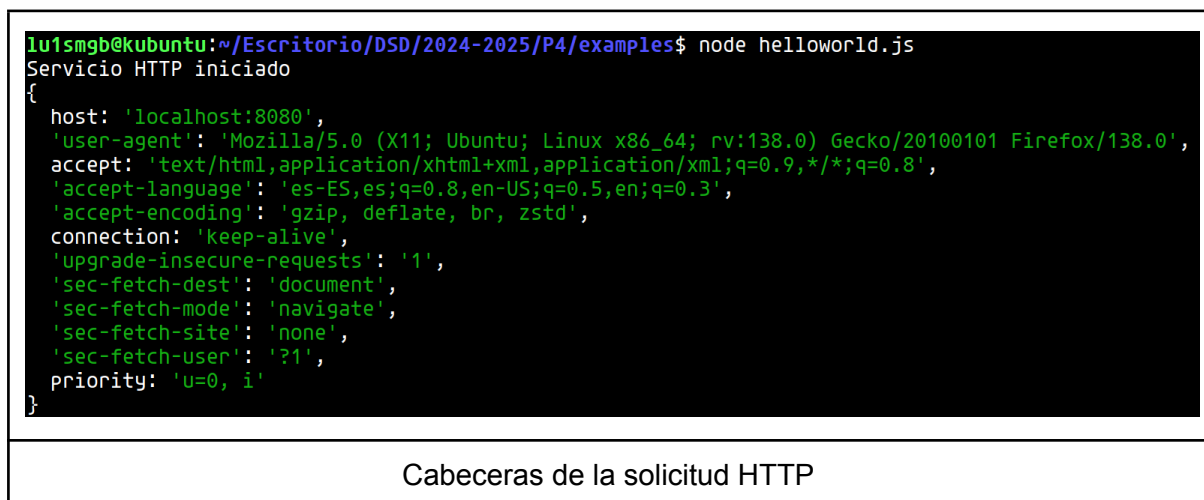
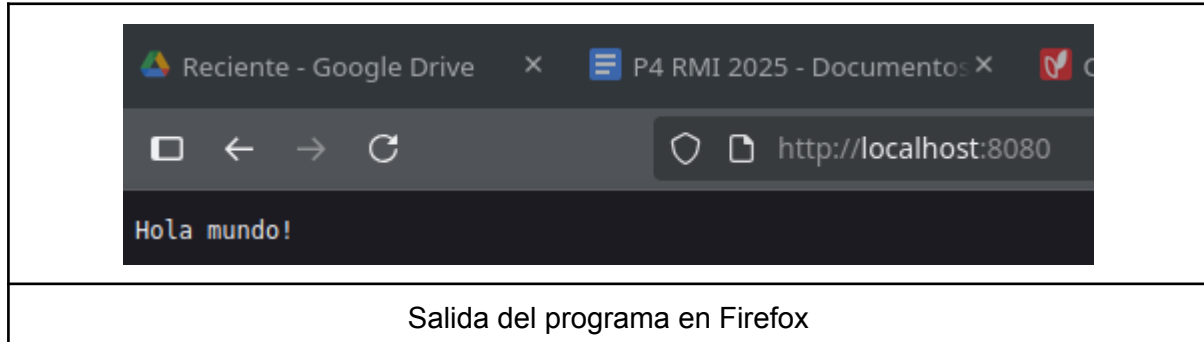
Servicio web con Node.js, Socket.io y MongoDB

Luis Miguel Guirado Bautista
Curso 2024-2025
Universidad de Granada

| | |
|--|----------|
| Ejemplos | 2 |
| Hola mundo | 2 |
| Calculadora simple | 3 |
| Calculadora con interfaz | 4 |
| Conexiones con Socket.io | 5 |
| Prueba de la base de datos | 6 |
| Sistema domótico | 7 |
| Valor añadido | 7 |
| Estructura del proyecto | 9 |
| Instalación de dependencias y ejecución del servicio | 9 |
| Base de datos | 10 |
| Servicio web | 11 |
| Comandos del bot de Telegram | 13 |
| Cliente | 14 |
| Capturas | 16 |
| Utilizar el sistema como usuario | 18 |
| Posibles mejoras | 18 |

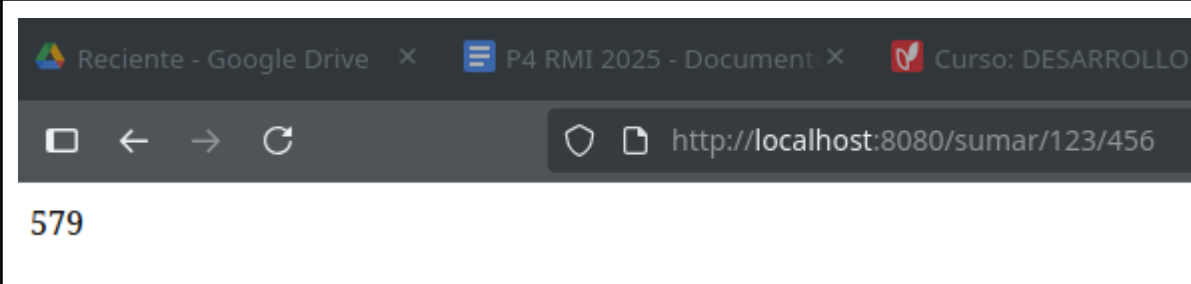
Ejemplos

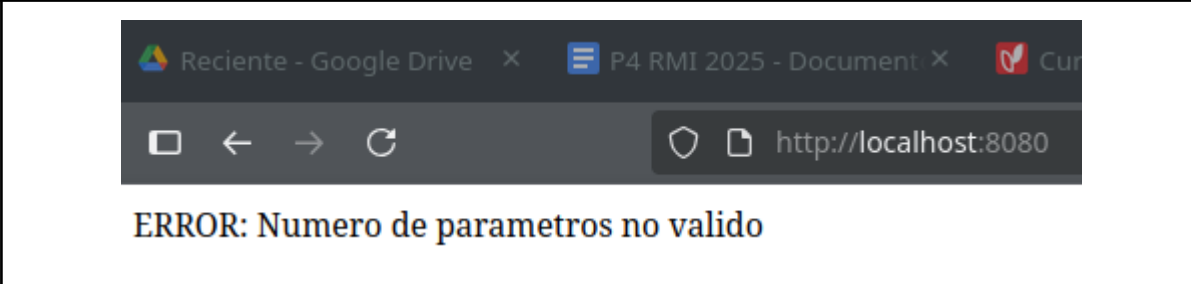
Hola mundo



El servidor se encarga de contestar todas las solicitudes HTTP que le lleguen al puerto 8080 con el mensaje en texto plano 'Hola mundo!' con el estado 200 (OK)
Ademas, en la consola del servidor, se imprimen las cabeceras de la solicitud recibida.

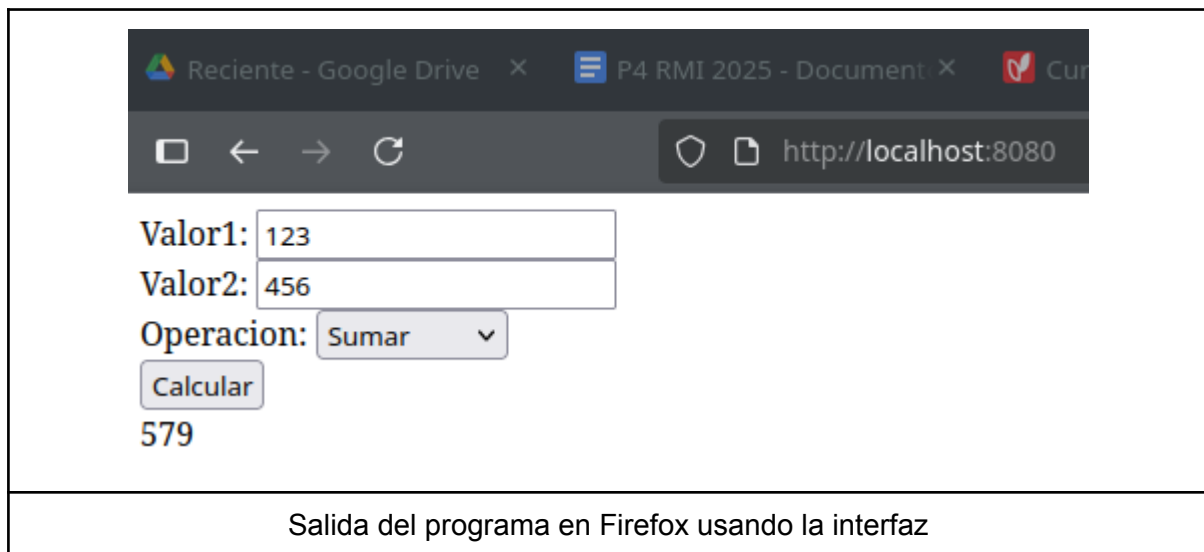
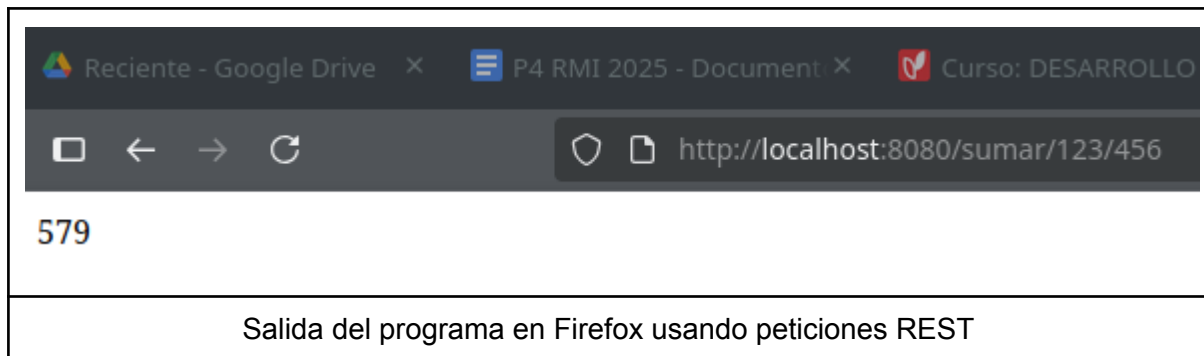
Calculadora simple

| |
|--|
|  |
| Salida del programa en Firefox al hacer 123 + 456 |

| |
|--|
|  |
| Salida del programa en Firefox sin argumentos |

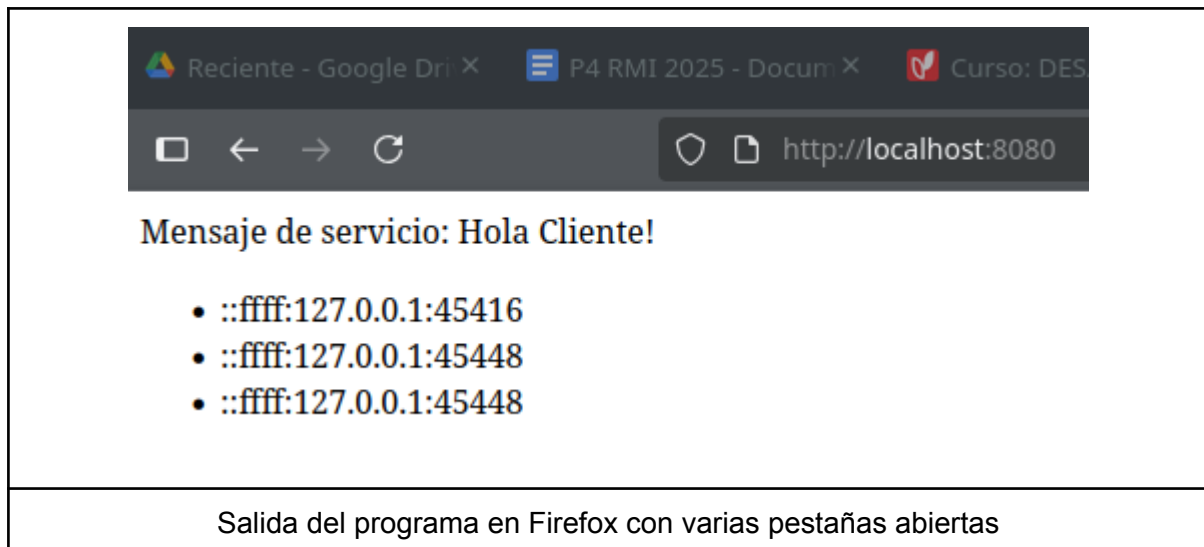
Ahora el servidor se encarga de realizar una operacion aritmetica en base a los parametros escritos en la propia URL de la solicitud. El primer argumento sera el tipo de operacion, y los otros dos los operandos.

Calculadora con interfaz



El servidor funciona igual que el anterior, pero esta vez, si no se le pasa ningun argumento, mostrará una interfaz gráfica en la que podremos realizar las operaciones aritméticas que queramos

Conexiones con Socket.io



El servidor utiliza el paquete Socket.io y una variable global para guardar las conexiones de los usuarios. Cuando se conecta un usuario, el servidor envía un mensaje de bienvenida, guarda la dirección y puerto del usuario y emite un evento para que se actualice la lista de usuarios a todos los que estén utilizando el servicio. Cuando se va a realizar una desconexión, se elimina de la lista de conexiones y se notifica a todos los usuarios.

Prueba de la base de datos

Reciente - Google Drive
P4 RMI 2025 - Documento
Curso: DESARROLLO DE SI
guionNodejs.pdf

http://localhost:8080

- {"_id":"6833b0009857e864c2f9f43e","host":"::ffff:127.0.0.1","port":60958,"time":"2025-05-26T00:04:16.924Z"}
- {"_id":"6833b0059857e864c2f9f43f","host":"::ffff:127.0.0.1","port":60958,"time":"2025-05-26T00:04:21.260Z"}
- {"_id":"6833b00a9857e864c2f9f440","host":"::ffff:127.0.0.1","port":60958,"time":"2025-05-26T00:04:26.059Z"}
- {"_id":"6833b00e9857e864c2f9f441","host":"::ffff:127.0.0.1","port":60958,"time":"2025-05-26T00:04:30.606Z"}

Salida del programa en Firefox realizando 4 conexiones

```

baseDatosTest> db.test.find({})
[
  {
    _id: ObjectId('6833b0009857e864c2f9f43e'),
    host: '::ffff:127.0.0.1',
    port: 60958,
    time: '2025-05-26T00:04:16.924Z'
  },
  {
    _id: ObjectId('6833b0059857e864c2f9f43f'),
    host: '::ffff:127.0.0.1',
    port: 60958,
    time: '2025-05-26T00:04:21.260Z'
  },
  {
    _id: ObjectId('6833b00a9857e864c2f9f440'),
    host: '::ffff:127.0.0.1',
    port: 60958,
    time: '2025-05-26T00:04:26.059Z'
  },
  {
    _id: ObjectId('6833b00e9857e864c2f9f441'),
    host: '::ffff:127.0.0.1',
    port: 60958,
    time: '2025-05-26T00:04:30.606Z'
  }
]
baseDatosTest> 

```

Estado de la coleccion en Mongosh

Cada vez que un usuario acceda al servidor web, se guardará el host y puerto obtenidos a traves de Socket.io junto con una marca de tiempo. En la interfaz se muestran todas las conexiones realizadas hasta el momento.

Sistema domótico

Se pide implementar un simulador de sistema domótico con al menos dos sensores para la temperatura y la luminosidad y al menos dos actuadores para el aire acondicionado y las persianas.

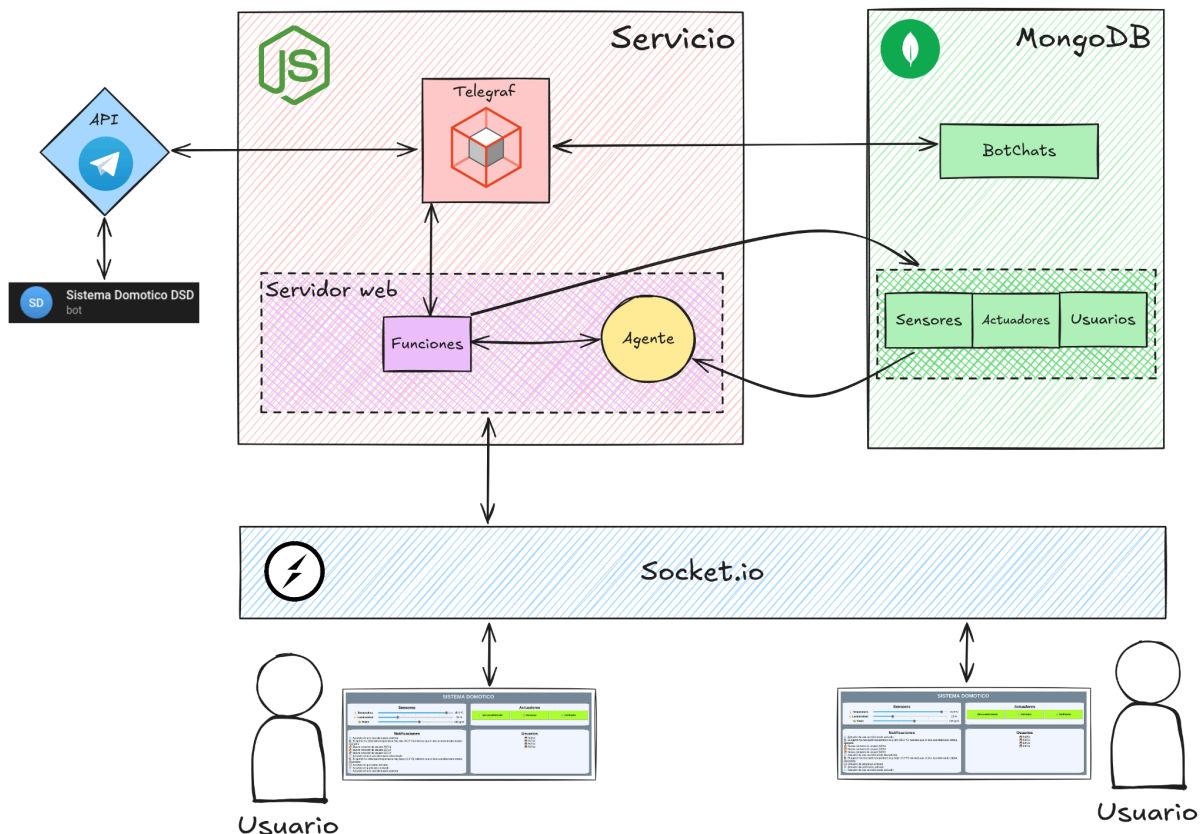
El sistema tendrá implementado un sistema de notificaciones o historial de eventos, una lista de usuarios conectados y un agente integrado en el mismo servicio que evalúe los sensores y tome acciones en base a los valores evaluados. El sistema también proporcionará una interfaz web para los clientes, así podrán interactuar con el sistema.

Valor añadido

Adicionalmente se ha implementado:

- Un nuevo sensor de polen y un actuador de purificador de aire
- Una mejora a la interfaz gráfica de usuario, pudiendo actualizar los valores de los sensores con deslizadores, coloreado de los botones para los actuadores según su estado, entre otras mejoras visuales como una mejor organización de los elementos de la misma
- Gestión de variables de entorno privadas con el paquete [dotenv](#)
- Un bot de Telegram que difunde las notificaciones a los usuarios. Además, los usuarios podrán consultar el estado del sistema y abrir/cerrar el aire acondicionado y cerrar/abrir las persianas. Se ha implementado usando el framework [Telegraf](#). El nickname del bot es @dsdp4lu1smgbot

Funcionamiento del sistema



En términos generales, el sistema se compone de tres componentes principales

- **Servicio web.** Encargado de procesar las solicitudes HTTP que reciba y los datos recibidos de los WebSockets generados por [Socket.io](#). Interactúa con la base de datos para realizar operaciones de lectura y escritura. También tiene acceso al framework Telegraf para procesar las interacciones del bot. El agente está integrado en el mismo servicio, actuando solamente cuando se procesa un cambio de uno de los sensores del sistema.
- **MongoDB.** Base de datos del sistema. Guarda toda la información que necesite permanecer de manera persistente, como la información de los sensores, los actuadores, las notificaciones, las conexiones abiertas y los chats del bot en los que está registrado
- **SocketIO.** Modelo de WebSockets para poder realizar la comunicación y envío de información entre clientes web y servidores mediante eventos.

Estructura del proyecto

El proyecto está compuesto por los siguientes archivos

| Fichero | Descripción |
|---|--|
| .env | Fichero usado para guardar variables de entorno que, normalmente guardan información sensible como la dirección de la base de datos de MongoDB, la dirección del servidor web o el token del bot de Telegram |
| cliente.html | Código HTML de la interfaz del cliente |
| cliente.css | Hoja de estilos en cascada de la interfaz del cliente |
| cliente.js | Funcionalidad de la interfaz del cliente |
| servidor.js | Código del servidor. También contiene la funcionalidad del agente y del bot. Debe ser ejecutado por Node.js |
| package.json package-lock.json | Ficheros de dependencias del proyecto |

Instalación de dependencias y ejecución del servicio

Para instalar las dependencias del proyecto, simplemente ejecutamos en el mismo directorio:

```
> npm install
```

Para ejecutar el servicio, ejecutamos el comando

```
> node servidor.js
Sistema domotico en marcha
Sat, 24 May 2025 18:36:30 GMT
```

Base de datos

Se han creado las siguientes colecciones de MongoDB para la implementación del sistema. El servidor será el encargado de inicializar la base de datos en caso de que no haya sido creada o falten colecciones por crear.

| Nombre | Descripción |
|--------------------------|--|
| AireAcondicionado | Estados que ha ido tomando el actuador del aire acondicionado, junto una marca de tiempo |
| BotChats | Chats en los que se ha ido registrando el bot de Telegram, junto a una marca de tiempo. El bot usará los chats de esta colección para difundir las notificaciones. |
| EstadosIniciales | Estados por defecto que deben tomar los sensores y actuadores, en caso de que no exista ninguno en sus respectivas colecciones. |
| Luminosidad | Estados que ha ido tomando el sensor de luminosidad, junto una marca de tiempo |
| Notificaciones | Registro de todas las notificaciones del sistema, junto con una marca de tiempo |
| Persianas | Estados que ha ido tomando el actuador de las persianas, junto una marca de tiempo |
| Polen | Estados que ha ido tomando el sensor de polen, junto una marca de tiempo |
| Purificador | Estados que ha ido tomando el actuador del purificador de aire, junto una marca de tiempo |
| Temperatura | Estados que ha ido tomando el sensor de temperatura, junto una marca de tiempo |
| Usuarios | Registro de todos los usuarios conectados mediante la interfaz web, junto a una marca de tiempo |

Servicio web

Se encarga de lanzar el servidor HTTP, establecer una conexión con la base de datos, inicializar los WebSockets del lado de servidor de Socket.io y de lanzar el bot de Telegram. Tendrá constancia de todas las colecciones de la base de datos, si detecta que faltan colecciones de datos correspondientes a los sensores y actuadores, lo inicializa. Todos los comportamientos están definidos en el código del servicio, estas son algunas de las funciones o métodos más importantes

| Función | Descripción |
|--|--|
| mostrarError | Muestra en la consola un error y como se ha provocado |
| emitirEventoATodos | Hace un broadcast de un evento a todos los clientes conectados |
| estadoMasReciente | Obtiene el estado de un sensor o actuador más reciente de la base de datos. La colección debe tener documentos con el campo estado |
| establecerEstadosPorDefecto | Comprueba si existen estados por defecto en la base de datos e inicializa los estados de los sensores y actuadores que no tengan ninguno. Se llama solo una vez cuando se inicia el servidor. |
| actualizarNotificacionesEnTodos | Obtiene las 10 notificaciones más recientes y hace un broadcast de estas notificaciones a todos los clientes conectados |
| botDifundeMensaje | El bot envía un mensaje a todos los chats registrados en la base de datos |
| notificar | Se crea una notificación en la base de datos, esto incluye la difusión a los clientes y el envío de la misma por el bot |
| actualizarSensor | Actualiza un sensor o actuador <u>en los clientes</u> , se debe especificar la colección correspondiente, el identificador de evento y un manejador que se encargará de hacerle saber a los usuarios sobre dicha actualización. A partir de este método también se ha implementado actualizarTemperatura, actualizarLuminosidad, actualizarPolen, actualizarAire, actualizarPersianas, actualizarPurificador y actualizarSensores. Se ejecuta cuando recibe un evento de alternar un actuador por parte de los clientes. |
| actualizarSensores | Actualiza todos los sensores y actuadores en los clientes |

| | |
|------------------------------------|--|
| registrarConexion | Registra una conexion en la base de datos y lo notifica |
| actualizarUsuariosEnTodos | Actualiza la lista de usuarios en todos los clientes |
| alternarActuador | Cambia el estado de un actuador, se debe especificar la coleccion correspondiente, el metodo de actualizacion (uno de los especificados anteriormente p.e actualizarAire) y un prefijo para la notificacion que se enviará. A partir de este método también se ha implementado alternarAire, alternarPersianas y alternarPurificador |
| emitirEventoACliente | Emite un evento a un cliente concreto |
| actualizarSensoresEnCliente | actualizarSensores, pero solo aplica a un cliente concreto |
| cambiarTemperatura | Cambia la temperatura del sistema por un determinado valor, aquí el agente evalúa este valor y decide si debe alternar el actuador del aire acondicionado o no. Se ejecuta cuando recibe un evento de cambio de temperatura del cliente. |
| cambiarLuminosidad | Cambia el nivel de luminosidad del sistema por un determinado valor, aquí el agente evalúa este valor y decide si debe alternar el actuador de las persianas o no. Se ejecuta cuando recibe un evento de cambio de luminosidad del cliente. |
| cambiarPolen | Cambia el nivel de polen del sistema por un determinado valor, aquí el agente evalúa este valor y decide si debe alternar el actuador del purificador de aire o no. Se ejecuta cuando recibe un evento de cambio de nivel de polen del cliente. |
| desconectar | Elimina la conexion de un usuario de la base de datos. Se ejecuta cuando un cliente se desconecta del sistema. |

Comandos del bot de Telegram

A continuación se muestra la lista de comandos del bot de Telegram, aunque el mismo bot provee una descripción de cada uno de los comandos con `/help`

Se recuerda que el bot de Telegram es @dsdp4lu1smgbot

| Comando | Descripción |
|---------------------------|---|
| <code>/start</code> | El bot registra este chat. Siempre es el primer comando que se ejecuta en cada chat nuevo. |
| <code>/help</code> | Muestra una lista de los comandos disponibles junto a una breve descripción de lo que hace cada uno |
| <code>/estado</code> | Muestra el estado de los sensores y los actuadores |
| <code>/aire</code> | Alterna el actuador del aire acondicionado |
| <code>/persianas</code> | Alterna el actuador de las persianas |
| <code>/purificador</code> | Alterna el actuador del purificador |

Cliente

La interfaz del cliente consta de 4 secciones

- Sensores: Hay un deslizador y un valor numerico por sensor para mostrar el estado actual de dicho sensor, junto a un título para saber a cuál corresponde
- Actuadores: Hay un botón por actuador y este cambia de color según su estado
- Notificaciones: Lista de notificaciones, limitado a mostrar hasta las 10 más recientes por el servicio
- Usuarios: Lista de usuarios conectados al servicio, solo se muestra el puerto

| Función | Descripción |
|---------------------------------|---|
| cambiarTemperatura | Emite un evento actualizar-temperatura con un determinado valor |
| cambiarLuminosidad | Emite un evento actualizar-luminosidad con un determinado valor |
| cambiarPolen | Emite un evento actualizar-polen con un determinado valor |
| alternarAire | Emite un evento alternar-aire |
| alternarPersianas | Emite un evento alternar-persianas |
| alternarPurificador | Emite un evento alternar-purificador |
| actualizarTemperatura | Actualiza el sensor de temperatura en la interfaz con un determinado valor |
| actualizarLuminosidad | Actualiza el sensor de luminosidad en la interfaz con un determinado valor |
| actualizarPolen | Actualiza el sensor de polen en la interfaz con un determinado valor |
| actualizarAire | Actualiza el actuador del aire en la interfaz con un determinado valor |
| actualizarPersianas | Actualiza el actuador de las persianas en la interfaz con un determinado valor |
| actualizarPurificador | Actualiza el actuador del purificador de aire en la interfaz con un determinado valor |
| actualizarListaUsuarios | Actualiza la lista de usuarios en la interfaz a partir de una lista determinada |
| actualizarNotificaciones | Actualiza la lista de notificaciones en la interfaz a partir de una lista determinada |
| mostrarErrorCritico | Muestra un mensaje si el cliente pierde la conexion con el servidor |

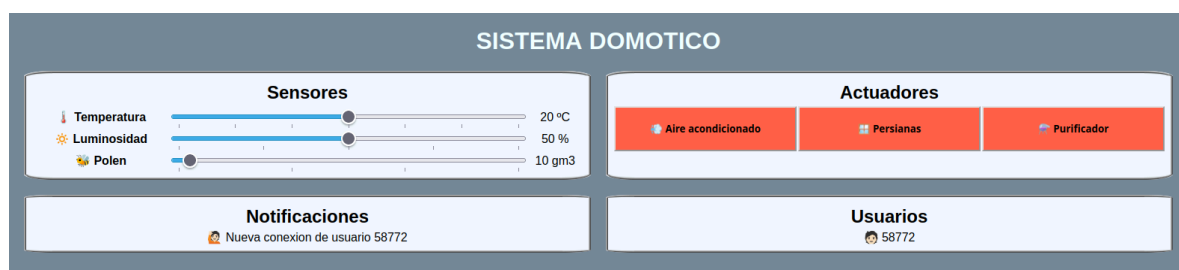
Eventos de Socket.io

| Función | Descripción |
|-------------------------------|--|
| obtener-notificaciones | Generado por el servicio. Actualiza las notificaciones en la interfaz de los clientes. |
| obtener-usuarios | Generado por el servicio. Actualiza los usuarios en la interfaz de los clientes. |
| obtener-temperatura | Generado por el servicio. Actualiza la temperatura en la interfaz de los clientes. |
| obtener-luminosidad | Generado por el servicio. Actualiza la luminosidad en la interfaz de los clientes. |
| obtener-polen | Generado por el servicio. Actualiza el polen en la interfaz de los clientes. |
| obtener-aire | Generado por el servicio. Actualiza el estado del aire en la interfaz de los clientes. |
| obtener-persianas | Generado por el servicio. Actualiza el estado de las persianas en la interfaz de los clientes. |
| obtener-purificador | Generado por el servicio. Actualiza el estado del purificador en la interfaz de los clientes. |
| actualizar-temperatura | Generado por el cliente. Cambia la temperatura del sistema |
| actualizar-luminosidad | Generado por el cliente. Cambia la luminosidad del sistema |
| actualizar-polen | Generado por el cliente. Cambia el nivel de polen del sistema. |
| alternar-aire | Generado por el cliente. Alterna el actuador del aire. |
| alternar-persianas | Generado por el cliente. Alterna el actuador de las persianas. |
| alternar-purificador | Generado por el cliente. Alterna el actuador del purificador. |
| disconnect | Generado por el cliente cuando se desconecta de la interfaz web. |

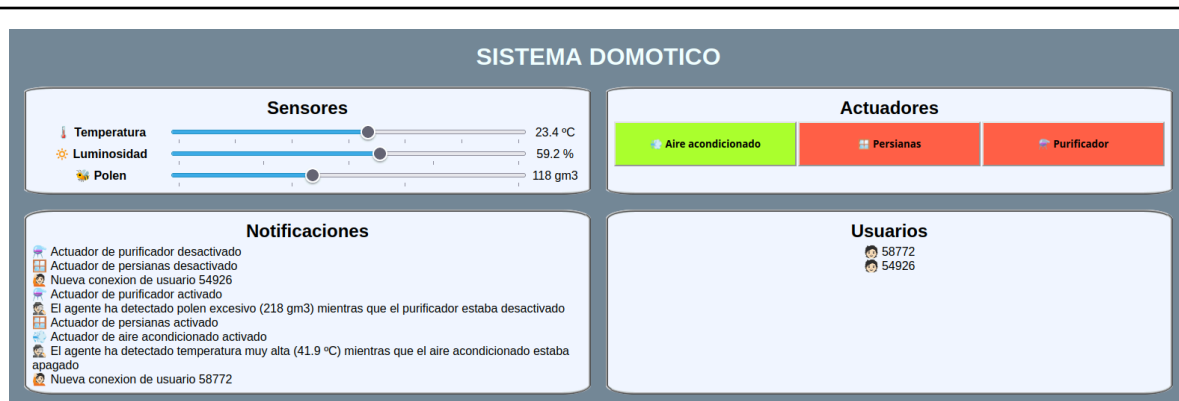
Capturas

```
lu1smgb@kubuntu:~/Escritorio/DSD/2024-2025/P4$ node servidor.js
Sistema domotico en marcha
Sun, 25 May 2025 22:41:50 GMT
Estados iniciales establecidos en la base de datos
Se ha establecido un estado por defecto para Temperatura
Se ha establecido un estado por defecto para Luminosidad
Se ha establecido un estado por defecto para Polen
Se ha establecido un estado por defecto para AireAcondicionado
Se ha establecido un estado por defecto para Persianas
Se ha establecido un estado por defecto para Purificador
```

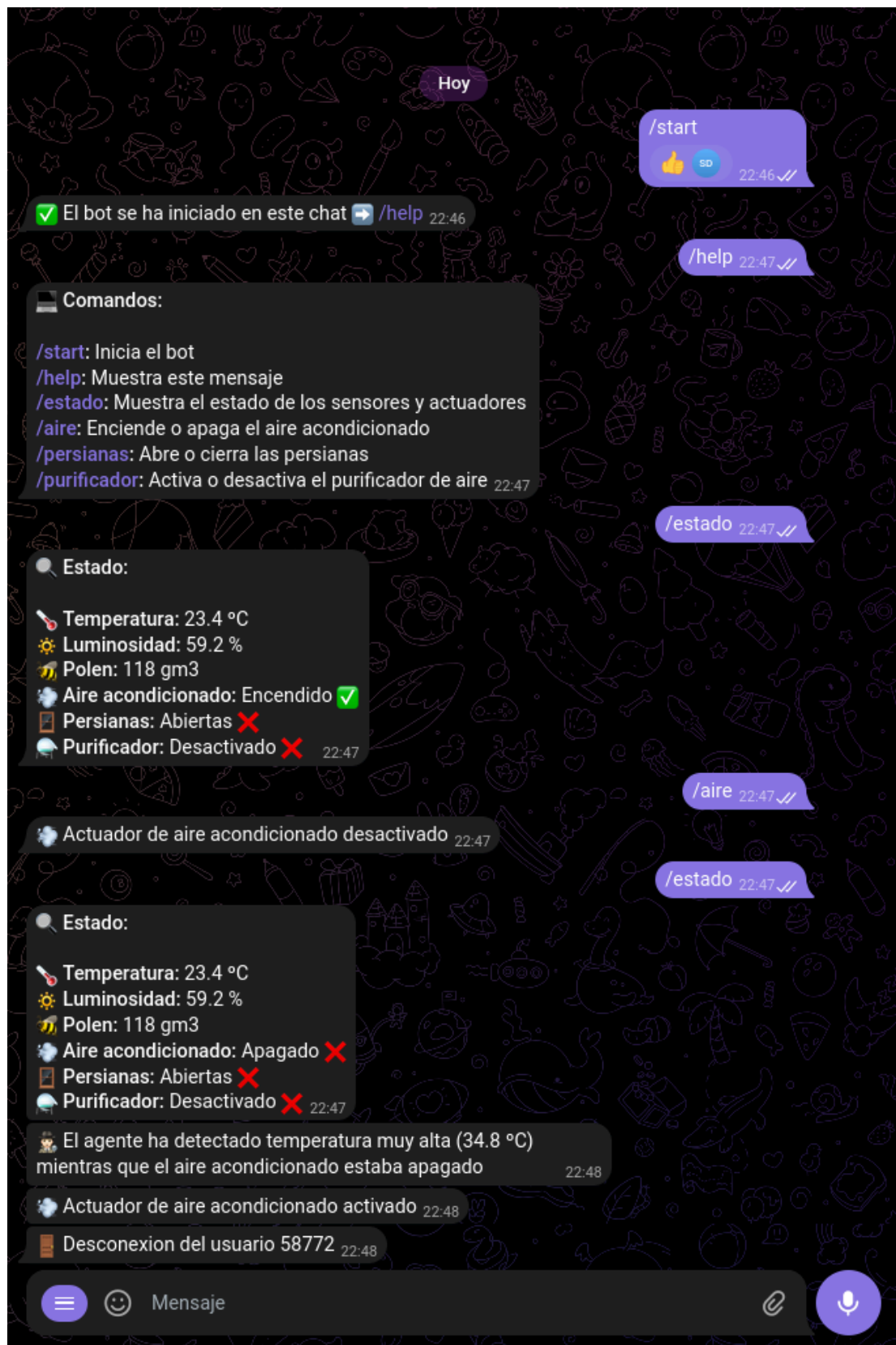
Salida del servidor cuando se ejecuta por primera vez, o con la base de datos vacía



Interfaz web del servicio con los estados por defecto, abierta con un navegador Firefox



Interfaz web del servicio tras cierta actividad, abierta con un navegador Firefox



Muestra de uso del bot de Telegram

Utilizar el sistema como usuario

- Si se quiere cambiar el valor de un sensor, utilice el deslizador y suéltelo cuando se encuentre en el valor deseado, la actualización en el sistema se realizará justo cuando lo suelte.
 - Si la temperatura sobrepasa los 30 grados y el aire acondicionado está apagado, el aire acondicionado se encenderá
 - Si la temperatura baja de los 15 grados y el aire acondicionado está encendido, el aire acondicionado se apagará
 - Si la luminosidad sube del 50% y las persianas están abiertas, se cerrarán
 - Si el nivel de polen sube de los 150 gm³ (gramos por metro cúbico), se activará el purificador de aire
- Si se quiere alternar un actuador, simplemente pulse el botón correspondiente. Los estados de los actuadores se pueden saber mediante su color (verde si está encendido o actuando o rojo si está apagado o no está actuando). En las persianas, si están cerradas, entonces se entiende que están haciendo algo, están bloqueando la luz de las ventanas.
- Si se quiere utilizar el bot de Telegram, vaya a Telegram y busque el bot por su nick @dsdp4lu1smgbot, la primera vez aparecerá un botón en grande que pone START, al hacer clic sobre él se ejecutará el comando /start y el bot registrará su chat, de modo que se le notificará cada vez que el sistema realice una notificación. Para saber qué hace cada comando, escriba /help.

Posibles mejoras

- Hacer al agente y al bot independientes del servicio web, como microservicios.
- Emplear programación orientada a objetos para lograr una fácil creación de sensores y actuadores, con métodos y atributos comunes, así como buenas prácticas para reducir la escritura de código repetitivo, aunque ya se aplica en ciertos métodos.
- Desplegar el servicio, por ejemplo, en la nube, para poder acceder a él desde cualquier lugar.
- Modo silencioso para el bot, de manera que el usuario no recibe todas las notificaciones del sistema.