

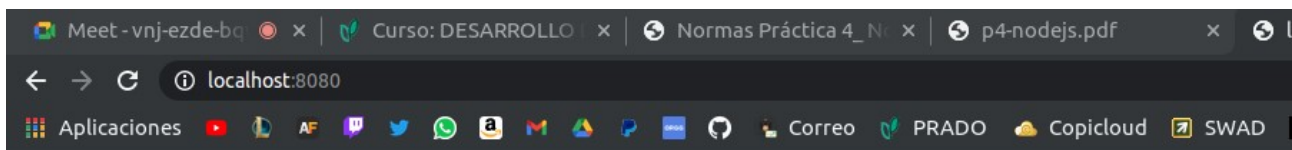
Documentación P4 – NodeJS

Raúl Castro Moreno
3º IS – DSD2

1. Ejemplos

1.1 Primer ejemplo

Enseña un poco como funciona NodeJS, sobre todo, el como importa módulos, en este caso el de http, y crea un server que devuelve el mensaje de “Hola Mundo” como respuesta. También se nos muestra como levantar el servidor, y en que dirección debemos acceder en el navegador, incluyendo el puerto especificado que en este caso es el 8080.

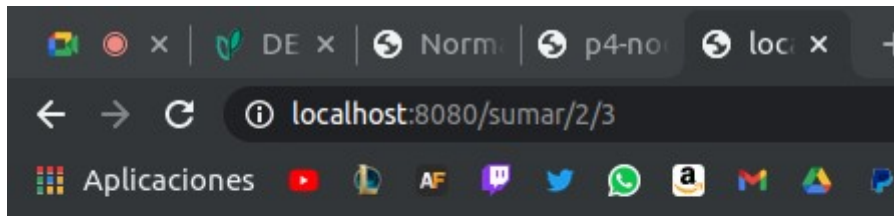


Hola mundo

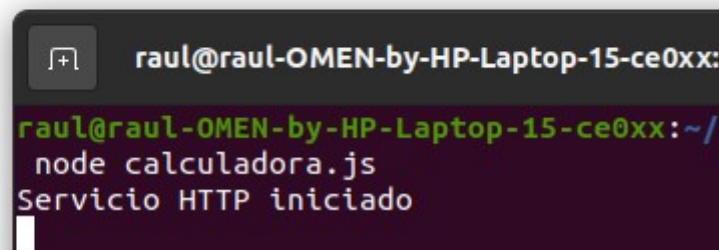
```
raul@raul-OMEN-by-HP-Laptop-15-ce0xx: ~/Escritorio/Home/...
raul@raul-OMEN-by-HP-Laptop-15-ce0xx:~/Escritorio/Home/DSD/p4/codigos-guion2020$
node helloworld.js
Servicio HTTP iniciado
{
  host: 'localhost:8080',
  connection: 'keep-alive',
  'sec-ch-ua': '" Not A;Brand";v="99", "Chromium";v="90", "Google Chrome";v="90"',
  'sec-ch-ua-mobile': '?0',
  'upgrade-insecure-requests': '1',
  'user-agent': 'Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like
Gecko) Chrome/90.0.4430.212 Safari/537.36',
  accept: 'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,imag
e/webp,image/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9',
  purpose: 'prefetch',
  '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,en;q=0.8'
}
{
  host: 'localhost:8080',
```

1.2 Segundo ejemplo

En este ejemplo, ahora se realiza un calculadora básica, con interfaz de tipo REST. En ella se ve como importa el módulo “url”. Este lo utiliza para coger el link, y poder partirlo para sacar la operación y los operandos. Mete en uri el pathname, en nuestro caso “/sumar/2/3” e indexa sobre el diviendolos por el carácter ‘/’. Una vez obtenidos la operación y los operandos, realiza la función de calcular y devuelve el resultado.

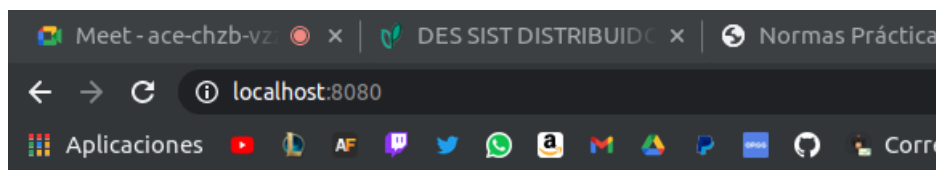


5

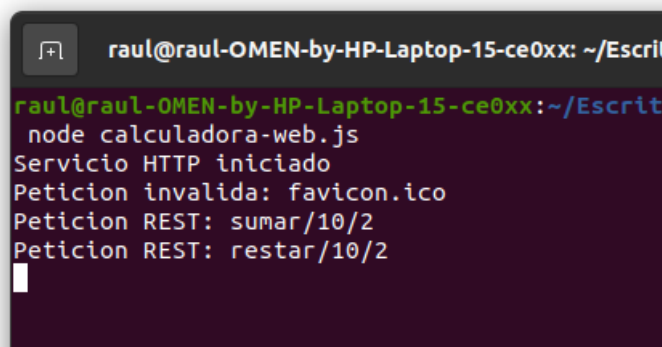


1.3 Tercer ejemplo

Utiliza el anterior como base, pero esta vez, se añade un archivo “html”, para darle una interfaz gráfica a la calculadora y sin tener que escribir nosotros la URL con la operación. Ahora el .js se encarga de mostrar esta página además de todo lo que hacia anteriormente, y para ello requiere los módulos de FileSystem y de Path, y los mimeTypes para las cabeceras. El html, se encarga a parte de ofrecer la interfaz gráfica, con los datos que pongamos en la interfaz, crea una petición REST al servidor, que se interpreta como en el anterior ejemplo, y obtiene el resultado el cuál es mostrado en el html.



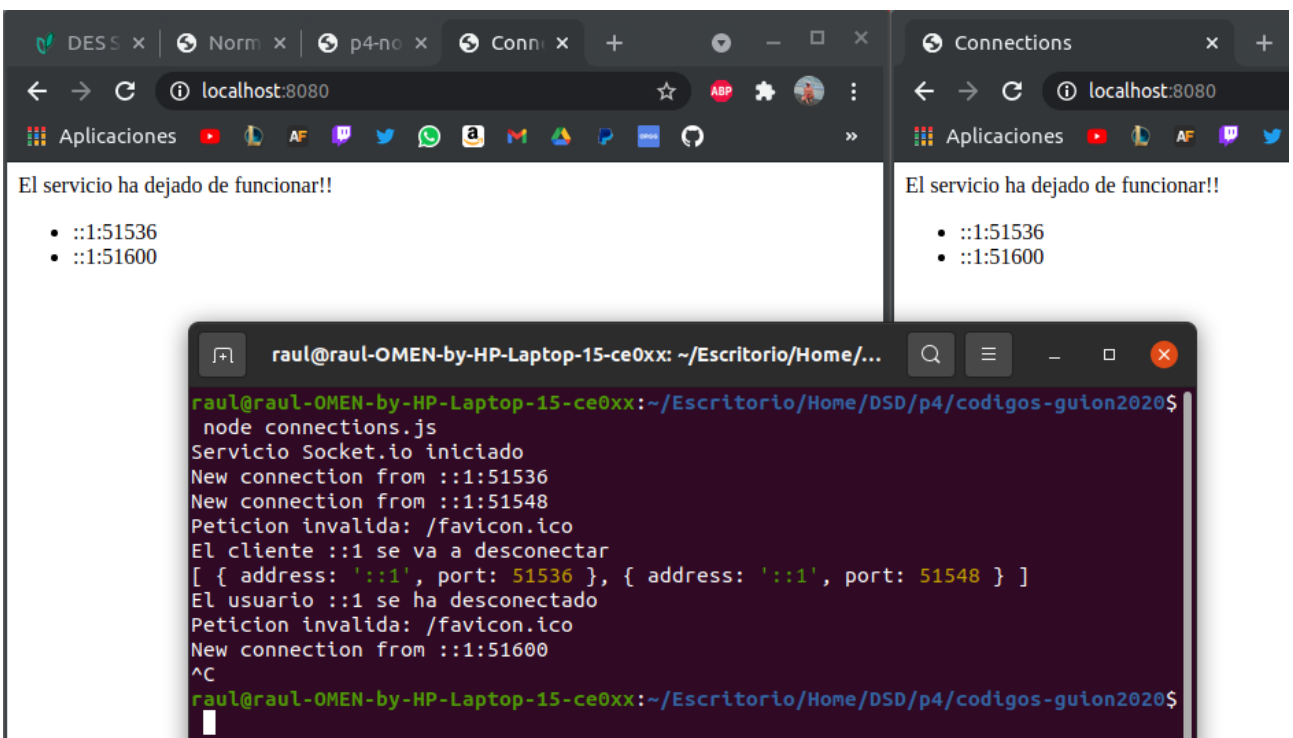
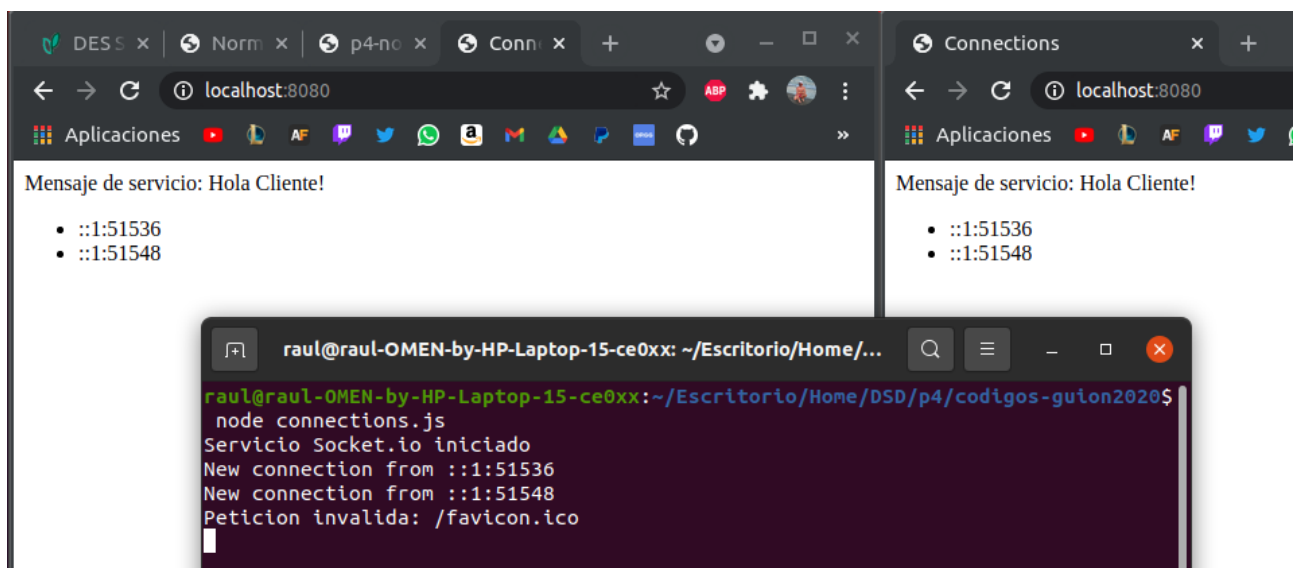
8



1.4 Cuarto ejemplo

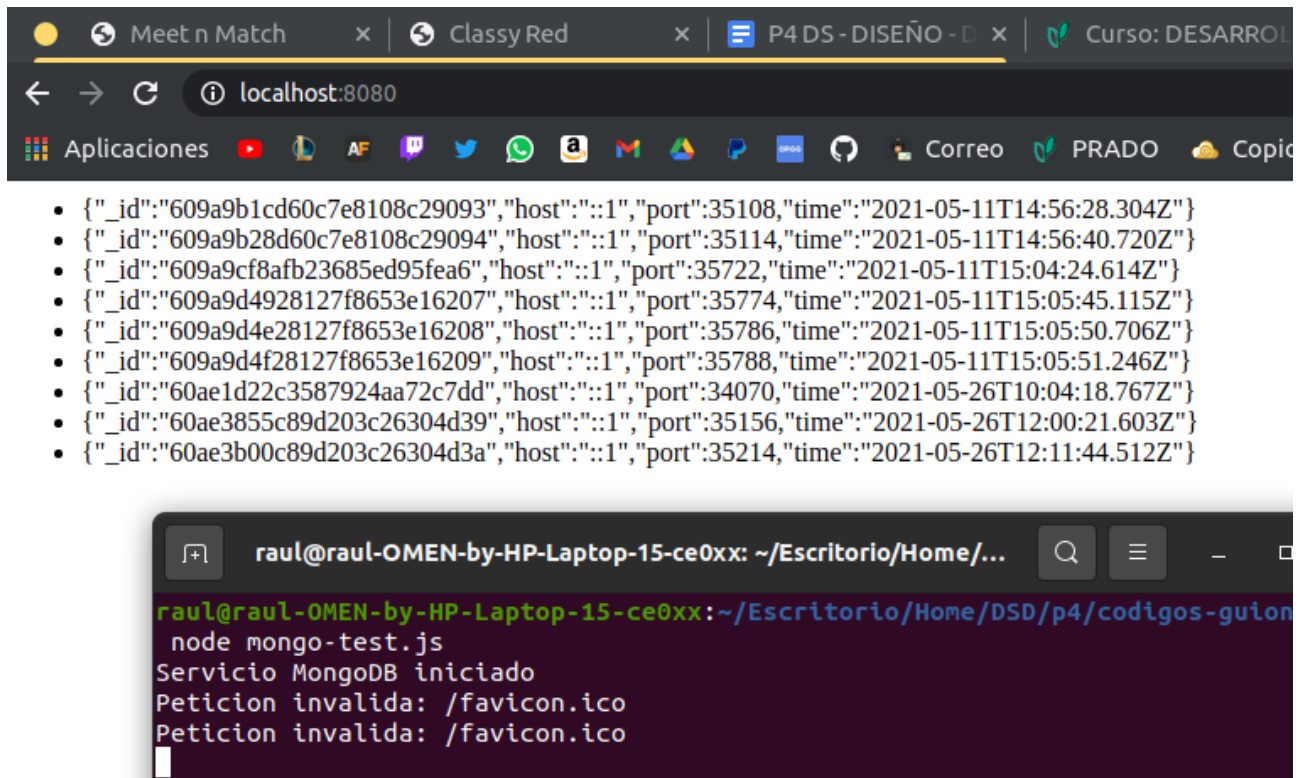
En este ejemplo, se muestra la conexión mediante socket.io en los cuales un cliente y el servicio se conectan y se subscriben a diferentes eventos, tanto de conexión, desconexión y otro llamado “output_evt”. Se observa como tanto cliente y servicio deben realizar .on y .emit de los eventos para poder tratarlos.

Se puede comprobar el funcionamiento abriendo varias ventanas de navegación. Cada vez que se abra una nueva ventana o se cierre se observa que la lista de usuarios va modificándose dinámicamente, agregándose o eliminándose usuarios de la lista, respectivamente. Además, se puede probar a terminar la ejecución del servicio. Se ve como se muestra un mensaje indicándolo.



1.5 Quinto ejemplo

En este ejemplo, ahora se usa todo lo anterior, y además se añade el uso de MongoDB para guardar los datos en una base de datos. En este caso se guardan los datos de los usuarios que se conectan y se muestran, cada vez que alguien se conecta se guarda una entrada nueva en la base de datos, la cual incluso apagando el servicio, sigue guardando los datos.



2. Sistema Domótico IoT

2.1 Parte Básica

Se nos pedía realizar un sistema domótico básico. Este tiene que contar con **2 sensores de luminosidad y temperatura** los cuáles sus medidas se simulan mediante un formulario el cuál publica el evento con dicha medida. También cuenta con **actuadores**, los cuales serían un **Aire Acondicionado y una persiana motorizada**. Estos actuadores son utilizados por el usuario, que los puede modificar manualmente, o por un agente, el cual utiliza los actuadores cuando las medidas captadas por los sensores superan un umbral máximo o mínimo, y a parte de usar los actuadores, notifica de esa acción a todos los usuarios del sistema.

Para empezar, hemos cogido de base el quinto ejemplo realizado, ya que vamos a usar **NodeJs, Socket.io y MongoDB**.

Sobre este ejemplo, en la parte del servidor, hemos en primer lugar las **variables** que **guardaran la Temperatura y Luminosidad** actual, y también hemos creado variables con los sus **Umbrales** respectivamente. Y por consiguiente, se han creado unas variables de **tipo booleano** que controlan el **estado de los actuadores**.

Con estas variables, ya podemos realizar una función llamada **“agenteDomótico”**, la cuál las va a utilizar para con uso de lógica, establecer cuando se producen las alarmas y los cambios en los actuadores por parte del agente. Además, por como lo he pensado yo a la hora de realizar la práctica, esta función **devolverá un array de mensajes y esos mensajes serán lo que se le muestre a los usuarios, que cuentan las acciones que ha realizado el agente**. Un ejemplo de esa lógica sería:

```
“SI ( temperaturaActual > UmbralMaximoTemperatura && !aireAcondicionado ){  
  
    aireAcondicionado = true  
    mensajes.add( “ El agente ha encendido el aire Acondicionado ” )  
}”
```

Una vez realizada la función con todas las posibles combinaciones, nos vamos a la parte de la comunicación mediante **Sockets**. Es aquí donde haremos un emit a los sockets clientes que se conecten ya sean usuarios o sensores, de los valores de los sensores y actuadores iniciales, para que puedan mostrarlos. El **u.emit(‘valoresSensores’)** se encarga de esto. Además a parte de eso, se le enviará también los mensajes de alarma obtenidos de la función **“agenteDomótico”**. Dentro de los cliente tenemos por consiguiente un **socket.on(‘valoresSensores’)** los cuales llaman a una función render que se encarga de imprimir en pantalla los valores recibidos por el socket cliente del servidor.

También, en el servidor, en la parte de el usuario conectado del servidor, he puesto escuchas a los clientes usuarios que se realizan sobre los cambios manuales de los actuadores. Esto quiere decir que tenemos un **u.on(‘AC’)** que enciende/apaga el actuador del aire acondicionado y lo notifica a todos los sockets. Lo mismo con la persiana.

Esta última notificación se debe a la función **“actualizarValoresUsuarios”** la cual tiene como parámetro un mensaje, y realiza lo mismo que la función comentada antes en **u.emit(‘valoresSensores’)**, solo que ahora el emit, no lo realiza el usuario, sino el socket del servidor a todos los sockets con un **io.sockets.emit(‘valoresSensores’)**.

Por último, en el servidor, tenemos una escucha de los valores de los sensores llamado **u.on('obtenerValoresSensores')** donde los datos que recibe son , el sensor que se está simulando, el valor de la medida, y una marca de tiempo de cuando se realiza. Esto se guarda en la base de datos, teniendo así un registro de los cambios ocurridos en las medidas de los sensores. Este registro se envía con una función **'obtenerRegistro()'** que se encarga de sacar los datos de la base de datos y enviarlos.

Ahora, dentro del html de lo que sería la página para los sensores, vamos a tener el formulario para simular la medida de los sensores, y aparte unos cuantos `<div>` con identificadores que utilizarán unas funciones de js de un script en el mismo fichero, para rellenarlos con, los datos de las medidas y sus valores, los datos de los actuadores y su estado (función **"render"**), un apartado con un listado de los usuarios activos en el servidor (función **"actualizarUsuarios"**) y por último, el registro con todas las acciones realizadas con los sensores (función **"actualizarRegistro"**), toda esta información obtenida por comunicación con el servidor mediante sockets. Y una función de **"enviar"**, que es la que se encarga de emitir el valor simulado establecido en el formulario, para que el servidor actualice su información.

Y por último, otro html para la página de usuario, con 2 apartados, uno con la misma información de las medidas, sus valores y los estados de los actuadores. El otro, con los botones para encender/apagar los actuadores de forma manual por el usuario.

Comentar que tanto la página de sensores como la de usuario, muestran los mensajes tanto de alarmas del agente, como de cambios manuales realizados por el usuario.

2.2 Parte Adicional

Para la parte adicional se ha realizado la adicción de más sensores y aparatos, se ha intentado realizar un interfaz de usuario más completa, se contempla la detección de eventos más complejos, se obtiene información de internet de alguna web, y se utilizan otros paquetes de Node.

2.2.1 Nuevos Sensores y Actuadores

Se han creado 3 sensores nuevos, dos de ellos son iguales que el de Temperatura y Luminosidad. Estos son el sensor de **Humedad** y **Suciedad**. El otro sensor es el de **Velocidad del Viento**, pero este funciona diferente ya que usa datos reales que comentaré más adelante. También se han creado 3 nuevos actuadores, un **Humificador**, una **Roomba**(robot aspirador) y una **Ventana** motorizada. Para todos estos se crean las variables necesarias, ya sea de umbrales, de estado de actuadores etc. Y se crean todos los métodos de comunicación de igual forma que para Temperatura y Luminosidad para poder actualizar sus valores y demás.

También se ha añadido una funcionalidad extra, la cuál es que se permite al usuario, activar o desactivar el agente domótico. En caso de estar desactivado, no enviará ninguna alarma ni tampoco modificará el estado de ningún actuador.

2.2.2 Eventos complejos

Como ahora tenemos nuevos sensores, podemos realizar unas combinaciones a la hora de activar o desactivar actuadores.

Se han realizado combinaciones por tanto entre el aire Acondicionado, el Humificador y la ventana, basadas en la realidad. Por ejemplo, si se va a encender el aire Acondicionado, debido a alguna alarma de temperatura, el agente comprueba también que la ventana este cerrada, y si no lo está la cierra. Otra también tiene que ver con que al encender un aire Acondicionado, ya que seca mucho el ambiente, pues se enciende el humificador, si es que la humedad actual no supera el umbral máximo de humedad. Y con esa lógica se formulan las demás combinaciones.

2.2.3 Obtención de información de Internet

Como se ha comentado en el 2.2.1, tenemos el sensor nuevo de **Velocidad del Viento**, pero lo que tiene de especial, es que nos da la velocidad del viento real sobre la ciudad de Granada, es decir no se simula su valor.

Esto se ha conseguido mediante la obtención de información de la API de <https://openweathermap.org/>. Para conseguirlo, he tenido que registrarme en la página, para que me pudieran dar una key de la API, y poder así obtener la información que quería sobre la velocidad del viento. Una vez obtenida la key, en el servidor, he tenido que utilizar el módulo “**request**”, para poder formular las peticiones.

Para ello he tenido que ejecutar antes el comando:

```
npm install request --save
```

El cual añadía todas las dependencias necesarias sobre el módulo. Luego en el servidor crear una variable con el **require(‘request’)**.

Finalmente, creo la función obtenerVientoApi, donde hago un request a la siguiente url:

[http://api.openweathermap.org/data/2.5/weather?
q=ciudad&lang=es&units=metric&appid=apiKey](http://api.openweathermap.org/data/2.5/weather?q=ciudad&lang=es&units=metric&appid=apiKey)

Donde en ciudad pongo Granada y en apiKey, la key que me ha proporcionado la página. Ya con eso, me devuelve la información completa del tiempo, y solo uso la velocidad del viento, la cual la guardo en la variable de la velocidad de viento actual que tiene el servidor.

Comentar por último, que esta función se realiza cada 20 segundos.

2.2.4 Otros paquetes de Node

El paquete que he utilizado, ha sido el de “**nodemailer**”, en mi caso, para enviar correos electrónicos que notifiquen al usuario del estado de las medidas de la casa.

Al igual que antes, instalamos las dependencias del módulo de nodemailer.

```
npm install nodemailer --save
```

Y lo guardamos en una variable con el **require(‘nodemailer’)**.

Para usarlo es sencillo, creamos un transportador con **nodemailer.CreateTransport({})**, donde especificaremos el servicio que usaremos (en mi caso gmail) y luego la autenticación de la

cuenta, la cual va a ser la que envíe los correos. (en mi caso he escogido una que tenía que no usaba). Después de esto, debemos de crear las opciones del correo, donde especificaremos que correo manda el email, el destinatario, el asunto y el contenido del email. Una vez realizado, tan solo hay que realizar un **transportador.sendMail(opcionesCorreo)**.

Comentar por último que aquí he tenido problemas de autenticación, y todo tenía que ver con que uso el servicio gmail, el cuál es de Google, y este tiene bloqueado las aplicaciones que le parecen insseguras. Para que funcionase, tan solo tuve que entrar con la cuenta que se autentifica y manda los correos en la siguiente página y activar la opción.

<https://myaccount.google.com/u/0/lesssecureapps?pli=1&rapt=AEjHL4NsHaRZXdAtbxSYfwycXCQjNRwrLTTLiXbz1KRooY9yESoFBeRVk2bcb6lLRzGotbwv8Md67aTFSQNHsLJ2vL6jE79S-A>

2.3 Ejemplos de ejecución

1. Página de simulación de los sensores

Página para Simular los Sensores

Valor a simular:

Sensor a simular: Luminosidad

Simular

Temperatura (°C) = 25 ||| Aire Acondicionado: false
Luminosidad (%) = 50 ||| Persiana : false
Humedad (%) = 50 ||| Estado Humidificador: false
Suciedad (%) = 0 ||| Estado Roomba: false
Velocidad viento (m/s) = 4.85 ||| Estado ventana: false
Estado del agente domótico : true

[Ir a la Página usuario](#)

Registro

- No hay datos guardados en el registro

Usuarios activos

- {"address":"","1","port":57884}
- {"address":"","1","port":57874}

2. Página del Usuario

The screenshot shows a web browser with two tabs: 'Sensores IoT' and 'Usuario IoT'. The address bar shows 'localhost:8080/usuario.html'. The page has a yellow background and is titled 'Página para el usuario'. It is divided into two main sections: 'Información del Sistema Domótico' on the left and 'Panel de control manual del usuario' on the right. The left section displays various sensor readings and system status: 'Temperatura (°C) = 25', 'Aire Acondicionado: false', 'Luminosidad (%) = 50', 'Persiana : false', 'Humedad (%) = 50', 'Estado Humidificador: false', 'Suciedad (%) = 0', 'Estado Roomba: false', 'Velocidad viento (m/s) = 4.85', 'Estado ventana: false', and 'Estado del agente domótico : true'. The right section contains a vertical stack of seven orange buttons: 'Encender/Apagar Aire Acondicionado', 'Encender/Apagar Persiana', 'Encender/Apagar Ventana', 'Encender/Apagar Roomba', 'Encender/Apagar Humidificador', 'Encender/Apagar Agente Domótico', and 'Enviar correo notificador'.

Información del Sistema Domótico

Temperatura (°C) = 25 ||| Aire Acondicionado: false
Luminosidad (%) = 50 ||| Persiana : false
Humedad (%) = 50 ||| Estado Humidificador: false
Suciedad (%) = 0 ||| Estado Roomba: false
Velocidad viento (m/s) = 4.85 ||| Estado ventana: false
Estado del agente domótico : true

Panel de control manual del usuario

Encender/Apagar Aire Acondicionado
Encender/Apagar Persiana
Encender/Apagar Ventana
Encender/Apagar Roomba
Encender/Apagar Humidificador
Encender/Apagar Agente Domótico
Enviar correo notificador

3. Simulación del sensor de temperatura a 40°C, supera el umbral de los 30°C y el agente muestra una alarma de que ha activado el Aire Acondicionado.

The screenshot shows the 'Página para Simular los Sensores' web interface. It has a yellow background and a title bar with 'Sensores IoT' and 'Usuario IoT' tabs. The address bar shows 'localhost:8080'. The page contains a simulation form with a text input for 'Valor a simular' (set to 40), a dropdown for 'Sensor a simular' (set to 'Temperatura'), and a 'Simular' button. Below the form, it displays the updated system status: 'Temperatura (°C) = 40', 'Aire Acondicionado: true', 'Luminosidad (%) = 50', 'Persiana : false', 'Humedad (%) = 50', 'Estado Humidificador: false', 'Suciedad (%) = 0', 'Estado Roomba: false', 'Velocidad viento (m/s) = 4.85', 'Estado ventana: false', and 'Estado del agente domótico : true'. A message states: 'El agente ha puesto el aire acondicionado porque la temperatura es muy alta'. There is a link 'Ir a la Página usuario'. On the right, under 'Usuarios activos', there is a list of two active users: {'address': ':::1', 'port': '57932'} and {'address': ':::1', 'port': '57928'}. At the bottom, under 'Registro', there is a log entry: {'sensor': 'temperatura', 'valor': '40', 'time': '2021-06-01T18:04:39.333Z'}.

Página para Simular los Sensores

Valor a simular:
40

Sensor a simular:
Temperatura

Simular

Temperatura (°C) = 40 ||| Aire Acondicionado: true
Luminosidad (%) = 50 ||| Persiana : false
Humedad (%) = 50 ||| Estado Humidificador: false
Suciedad (%) = 0 ||| Estado Roomba: false
Velocidad viento (m/s) = 4.85 ||| Estado ventana: false
Estado del agente domótico : true
El agente ha puesto el aire acondicionado porque la temperatura es muy alta
[Ir a la Página usuario](#)

Usuarios activos

- {"address": ":::1", "port": "57932"}
- {"address": ":::1", "port": "57928"}

Registro

- {"sensor": "temperatura", "valor": "40", "time": "2021-06-01T18:04:39.333Z"}

4. Muestra , de que las alarmas del agente se notifican y muestran a todos los usuarios, en este caso, al simular una humedad por debajo del umbral mínimo.



The screenshot shows a web browser window with the address bar displaying 'localhost:8080/usuario.html'. The page has a yellow background and is titled 'Página para el usuario'. It is divided into two main sections: 'Información del Sistema Domótico' on the left and 'Panel de control manual del usuario' on the right.

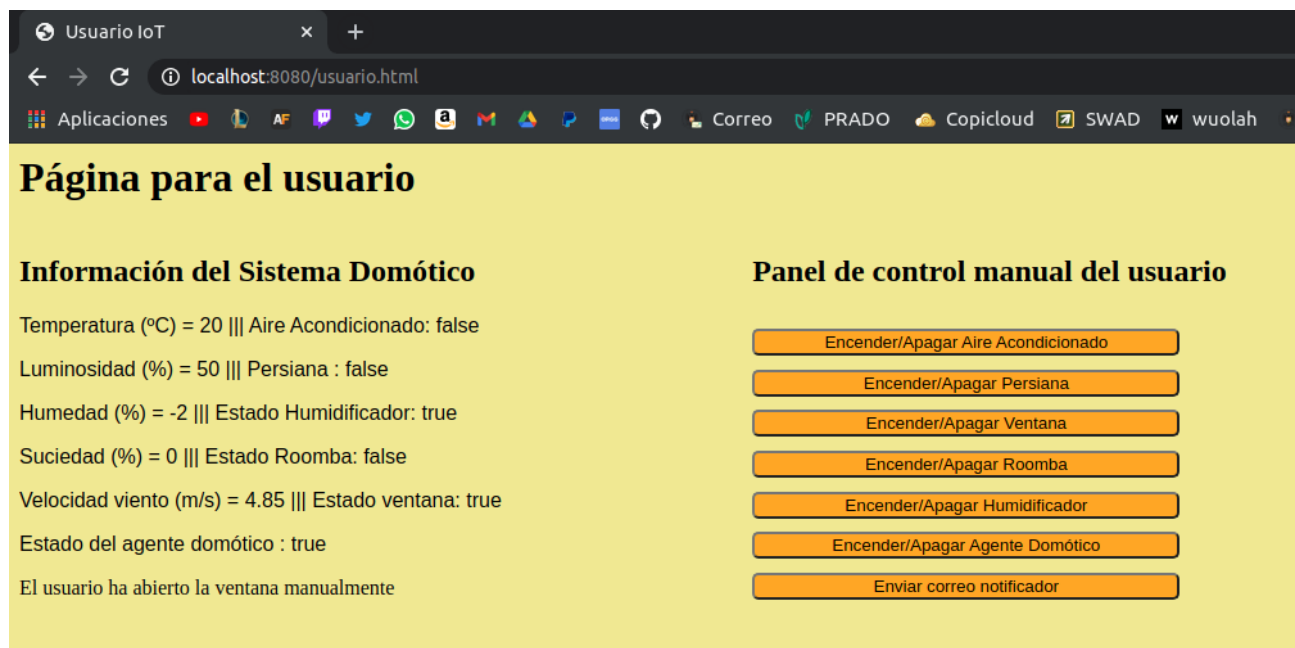
Información del Sistema Domótico

- Temperatura (°C) = 20 ||| Aire Acondicionado: false
- Luminosidad (%) = 50 ||| Persiana : false
- Humedad (%) = -2 ||| Estado Humidificador: true
- Suciedad (%) = 0 ||| Estado Roomba: false
- Velocidad viento (m/s) = 4.85 ||| Estado ventana: false
- Estado del agente domótico : true
- El agente ha encendido el humidificador porque el ambiente está muy seco

Panel de control manual del usuario

- Encender/Apagar Aire Acondicionado
- Encender/Apagar Persiana
- Encender/Apagar Ventana
- Encender/Apagar Roomba
- Encender/Apagar Humidificador
- Encender/Apagar Agente Domótico
- Enviar correo notificador

5. Mensaje que se muestra cuando el usuario pulsar algún botón para modificar un actuador de manera manual, en este caso la ventana.



The screenshot shows the same web browser window as before, but with updated system information and a message at the bottom of the 'Información del Sistema Domótico' section.

Información del Sistema Domótico

- Temperatura (°C) = 20 ||| Aire Acondicionado: false
- Luminosidad (%) = 50 ||| Persiana : false
- Humedad (%) = -2 ||| Estado Humidificador: true
- Suciedad (%) = 0 ||| Estado Roomba: false
- Velocidad viento (m/s) = 4.85 ||| Estado ventana: true
- Estado del agente domótico : true
- El usuario ha abierto la ventana manualmente

Panel de control manual del usuario

- Encender/Apagar Aire Acondicionado
- Encender/Apagar Persiana
- Encender/Apagar Ventana
- Encender/Apagar Roomba
- Encender/Apagar Humidificador
- Encender/Apagar Agente Domótico
- Enviar correo notificador

6. Muestra de un evento complejo. Se marca un temperatura que supera el umbral, por lo tanto el agente activa el Aire Acondicionado, sin embargo la ventana esta abierta, y es por ello que la cierra .

The screenshot shows a web browser window with the address bar displaying 'localhost:8080/usuario.html'. The page has a yellow background and is titled 'Página para el usuario'. It is divided into two main sections: 'Información del Sistema Domótico' on the left and 'Panel de control manual del usuario' on the right.

Información del Sistema Domótico

Temperatura (°C) = 50 ||| Aire Acondicionado: true
Luminosidad (%) = 50 ||| Persiana : false
Humedad (%) = -2 ||| Estado Humidificador: true
Suciedad (%) = 0 ||| Estado Roomba: false
Velocidad viento (m/s) = 4.85 ||| Estado ventana: false
Estado del agente domótico : true

El agente ha puesto el aire acondicionado porque la temperatura es muy alta ,

El agente ha cerrado la ventana porque el Aire Acondicionado está en uso

Panel de control manual del usuario

The panel contains seven orange buttons with the following labels: 'Encender/Apagar Aire Acondicionado', 'Encender/Apagar Persiana', 'Encender/Apagar Ventana', 'Encender/Apagar Roomba', 'Encender/Apagar Humidificador', 'Encender/Apagar Agente Domótico', and 'Enviar correo notificador'.

7. Muestra del correo notificador que podemos enviar.

The screenshot shows a Gmail interface with a dark theme. The left sidebar contains navigation options: 'Redactar', 'Recibidos', 'Destacados', 'Pospuestos', 'Enviados', 'Borradores', 'Más', and a 'Meet' section with 'Nueva reunión' and 'Unirse a una reunión'. The main area displays an email titled 'Informe del Sistema IoT' received from 'ravenwraith00@gmail.com'.

Informe del Sistema IoT (Recibidos x)

ravenwraith00@gmail.com
para mí ▾

Temperatura (°C) = 50 ||| Aire Acondicionado: true
Luminosidad (%) = 50 ||| Persiana : false
Humedad (%) = -2 ||| Estado Humidificador: true
Suciedad (%) = 0 ||| Estado Roomba: false
Velocidad viento (m/s) = 4.85 ||| Estado ventana: false
Estado del agente domótico : true

At the bottom of the email are two buttons: 'Responder' and 'Reenviar'.