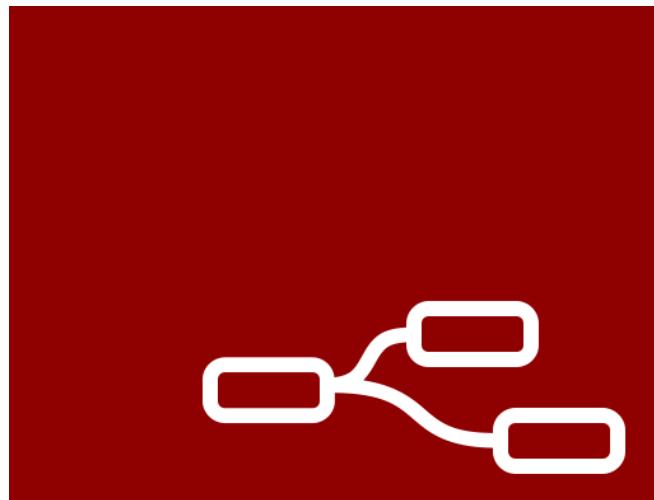


Sistemas Industriales

Practica 3



Node-RED

Elvi Mihai Sabau Sabau^{1[51254875L]}

¹ Universidad de Alicante, Alicante, España.
emss5@alu.ua.es

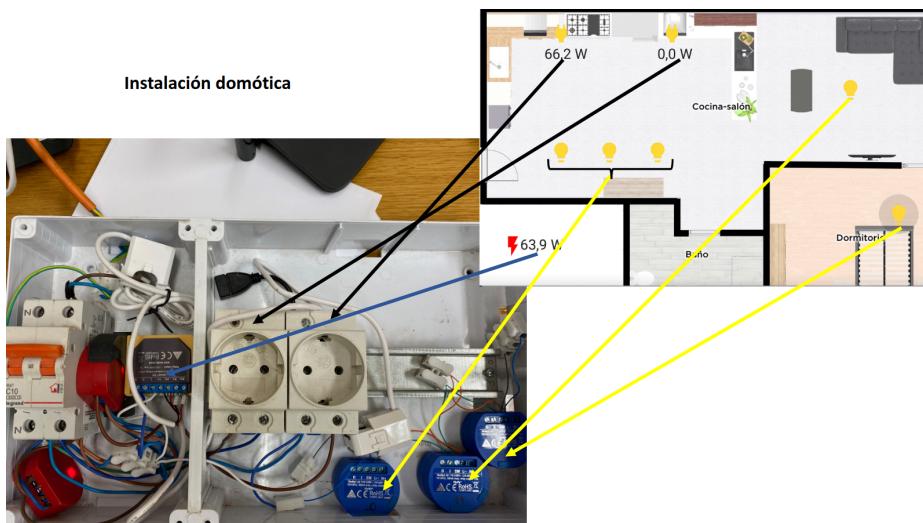
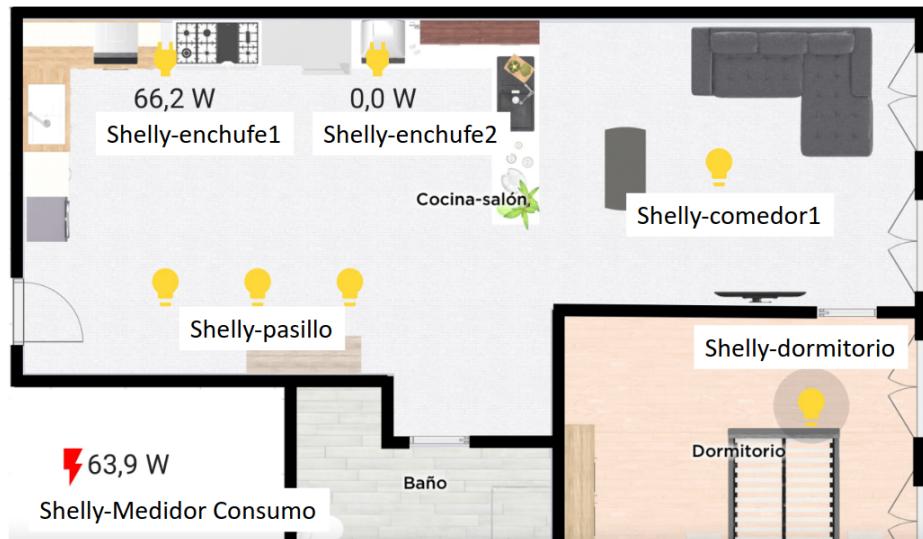
Abstract. En este documento se redacta todas las tareas realizadas durante las sesiones de teoría para el segundo bloque entregable, además de otros apartados extra expuestos por cada sesión.

Descripción de la práctica.	3
Parte 1.	4
Desarollo	4
Criterio 1 y 3 - Interfaz para controlar, ver el estado de los relés y escenarios.	4
Criterio 2 y 4 - Interfaz para monitorizar el consumo de energía de la casa, y habilitar / deshabilitar algoritmos de gestión energética y ahorro ambiental.	6
Esquema global del Flujo.	8
Diseño de la interfaz:	8
Documento a analizar.	9
Parte 2.	10
Ejercicio 1 - Home Assistant	10
Ejercicio 2 - Interacción con una DB, y registro de los eventos de nuestra instalación domótica.	15
Ejercicio 3 - Formulario con QR y Base de datos.	23
Ejemplo SVG.	24
Ejemplo QR.	26
Ejemplo formulario.	30
Ejercicio 3	32
Conclusion	37

Descripción de la práctica.

En esta segunda práctica ya podremos poner en práctica todo lo aprendido en la primera práctica. Esta práctica está fraccionada en 2 partes, la primera será la de realizar un pequeño proyecto para simular la gestión de una instalación domótica de una casa. Y la segunda parte trata sobre la instalación y gestión de HomeAssistant, además del análisis de otros nodos y de la puesta en marcha de otro proyecto más pequeño donde integraremos estos nodos y el home assistant.

El esquema de la casa es la siguiente:



Parte 1.

En esta parte se pide crear una interfaz en node-red para usar de interfaz entre los dispositivos shelly y otros sensores de la instalación domótica de esta casa.

En concreto, se piden los siguientes criterios.

1. Interfaz para controlar y ver el estado de los relés (enchufes y luces).
 - a. Se pide desarrollar una pantalla donde poder ver el estado de los relés, y poder encender / apagarlos.
2. Interfaz para monitorizar el consumo de la vivienda (medidor de consumo).
 - a. Se pide desarrollar una pantalla donde poder ver el consumo actual de la vivienda de forma intuitiva.
3. Interfaz con configuraciones predefinidas (escenarios) para la vivienda.
 - a. Se pide desarrollar una pantalla desde donde podamos seleccionar una preconfiguración para la vivienda, donde gestionaremos a la vez varios relés.
4. Interfaz para gestionar el consumo de la vivienda de forma automática.
 - a. Desarrollar algoritmos de control ambiental utilizando los valores de la Temperatura y Humedad del sensor BLE.
 - b. Desarrollar algoritmos de control del consumo energético de la vivienda, y habilitar / deshabilitar ciertos relés dependiendo del consumo de la vivienda.
 - c. Se pide desarrollar una pantalla donde podamos gestionar el consumo de la vivienda habilitando varios algoritmos que automaticen los relés.
5. Se pide además, el análisis del [siguiente documento](#).

Desarrollo

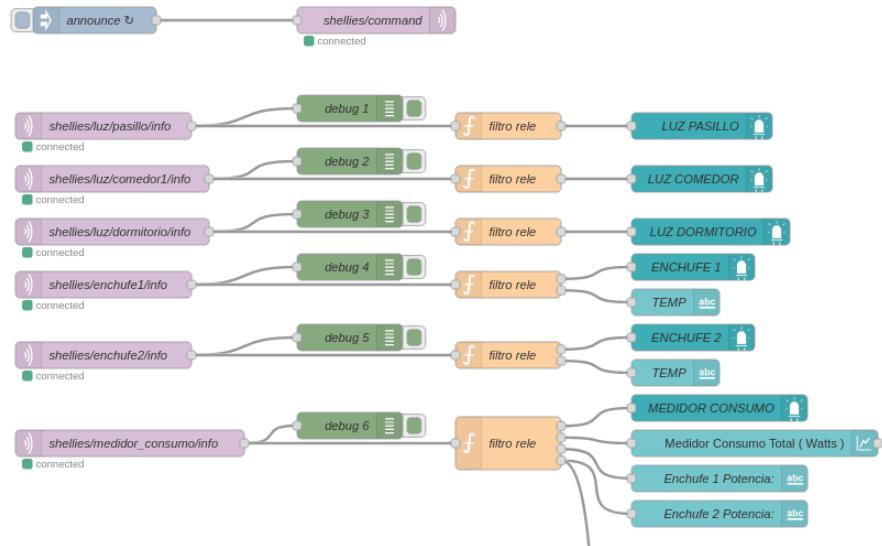
En mi caso, para la implementación del punto 1 al 4, lo que he hecho ha sido agrupar en una pestaña el punto 1 con el punto 3, y en otra el punto 2 con el punto 4, de esta manera damos al usuario la opción de ver los cambios en vivo de las acciones sin tener que desplazarse de una pestaña a otra.

Criterio 1 y 3 - Interfaz para controlar, ver el estado de los relés y escenarios.

Para controlar los relés tenemos 2 maneras, via MQTT o directamente atacar a cada rele via HTTP, yo he usado la primera manera.

Mediante MQTT, se ataca al tópico “shellies/command” con el contenido “announce” para forzar a los dispositivos conectados al broker emitir su estado actual en su totalidad, de esta manera, mediante un bucle de cada 5 segundos, obtengo el estado actual de cada relé de forma sencilla.

En concreto, lo que muestro en el dashboard es el estado actual de cada relé, su consumo y temperatura.



Los filtros de los reles siguen la siguiente logica:

Filtro luces:

```

1 let ison = msg.payload.relays[0].ison;
2 return {payload: ison};
  
```

Filtro enchufes:

```

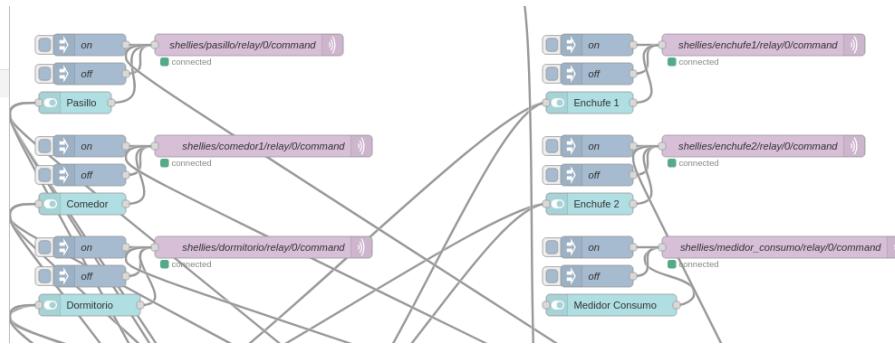
1 let ison = msg.payload.relays[0].ison;
2 let temp = msg.payload.temperature;
3 return [{payload:ison}, {payload:temp}];
  
```

Filtro medidor_consumo:

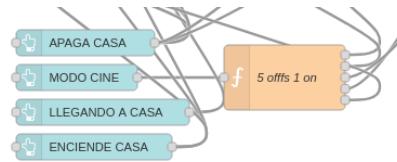
```

1 let ison = msg.payload.relays[0].ison;
2 let enchufe1_power = msg.payload.emeters[0].power;
3 let enchufe2_power = msg.payload.emeters[1].power;
4 let temp = msg.payload.relays[0].temperature;
5 let total = msg.payload.emeters[0].total + msg.payload.emeters[1].total;
6
7 return [{ payload: ison }, { payload: total }, { paload: enchufe1_power }, { payload: enchufe2_power }];
  
```

Después, para controlar cada relé, he creado un array de interruptores que emiten un “on” o “off” dependiendo del estado del interruptor.



Después, para controlar cada escenario, he creado un botón para cada escenario, que devuelve “true” o “false” a cada interruptor, dependiendo del escenario que se quiera habilitar.



Criterio 2 y 4 - Interfaz para monitorizar el consumo de energía de la casa, y habilitar / deshabilitar algoritmos de gestión energética y ahorro ambiental.

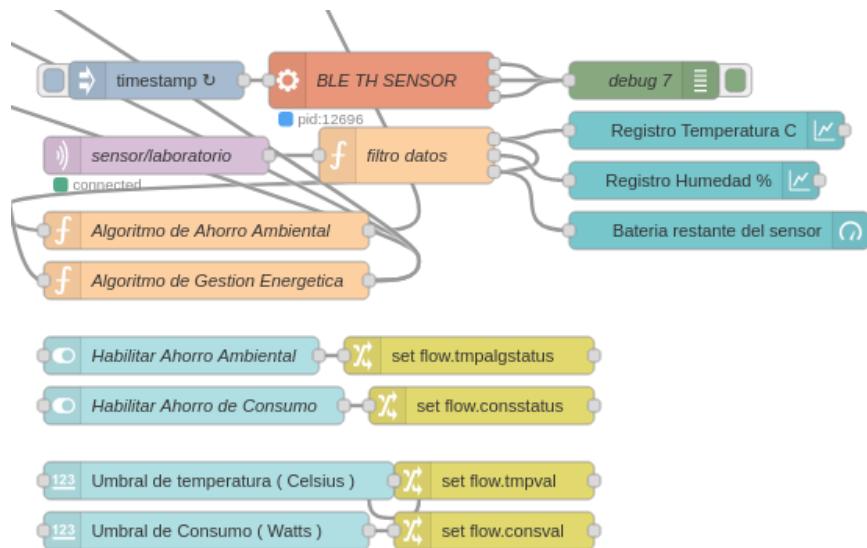
Para gestionar el consumo energético de la vivienda de forma automática, solicitó datos al sensor ambiental y al relé de consumo energético, y con los datos recibidos he realizado 2 algoritmos:

Uno de gestión energética que apaga las luces automáticamente de toda la vivienda dado un umbral de consumo (personalizable por el usuario).

Y otro de ahorro ambiental, donde enciendo o apago el enchufe 2 (donde supuestamente estara enchufada una estufa) dependiendo del umbral de una temperatura mínima dada por el usuario. De esta forma, la calefacción de la casa se gestiona de forma automática, y es configurable por el usuario.

En esta misma pestaña, muestro además 3 graficos y 1 gauge.

Un gráfico que muestra los valores de consumo total de la vivienda, asociado al relé de gestión de consumo, 2 gráficos mostrando la temperatura y la humedad de la vivienda, y 1 gauge que representa la batería restante del sensor de temperatura y humedad.



La lógica de los algoritmos es la siguiente:

```

2 // Si el modo de ahorro esta habilitado
3 // y la temperatura es menor o igual a flow.tmpval
4 // Enciende el rele de la estufa estufa.
5 // ( Enchufe 2 )
6 if (flow.tmpalgstatus && msg.payload <= flow.tmpval )
7 | return {payload: "on"};
8 return {payload: "off"};
```



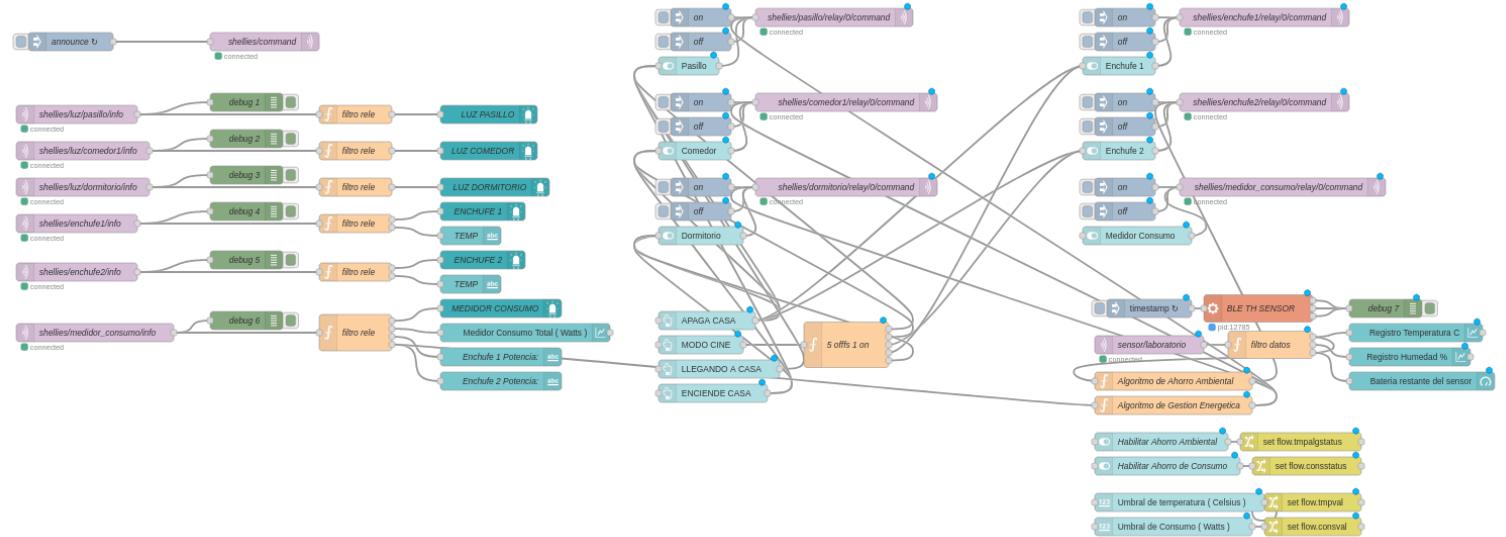
```

2 // Si el modo de gestion energetica esta habilitado
3 // y el consumo general es mayor que flow.consval.
4 // ( Luces )
5 if (flow.consstatus && msg.payload >= flow.consval )
6 | return {payload: "on"};
7 return {payload: "off"};
```

Además, se da la opción al usuario de deshabilitar o habilitar estos algoritmos cuando quiera a través de un interruptor.

Esquema global del Flujo.

El esquema del flow en su totalidad es la siguiente:



Diseño de la interfaz:

Y este es el Diseño resultante de la UI para la primera parte de la Práctica 2:

En la página principal, damos la opción de los criterios 1 y 3, poder ver el estado de los relés, ver consumo y temperatura (enchufes), poder encender y apagarlos, y cambiar el escenario de la casa.

The screenshot displays a web-based smart home control interface. On the left sidebar, there are two main sections: 'PANEL PRINCIPAL' and 'MONITORES Y CONTROL'. The 'MONITORES Y CONTROL' section is currently active, indicated by a teal background.

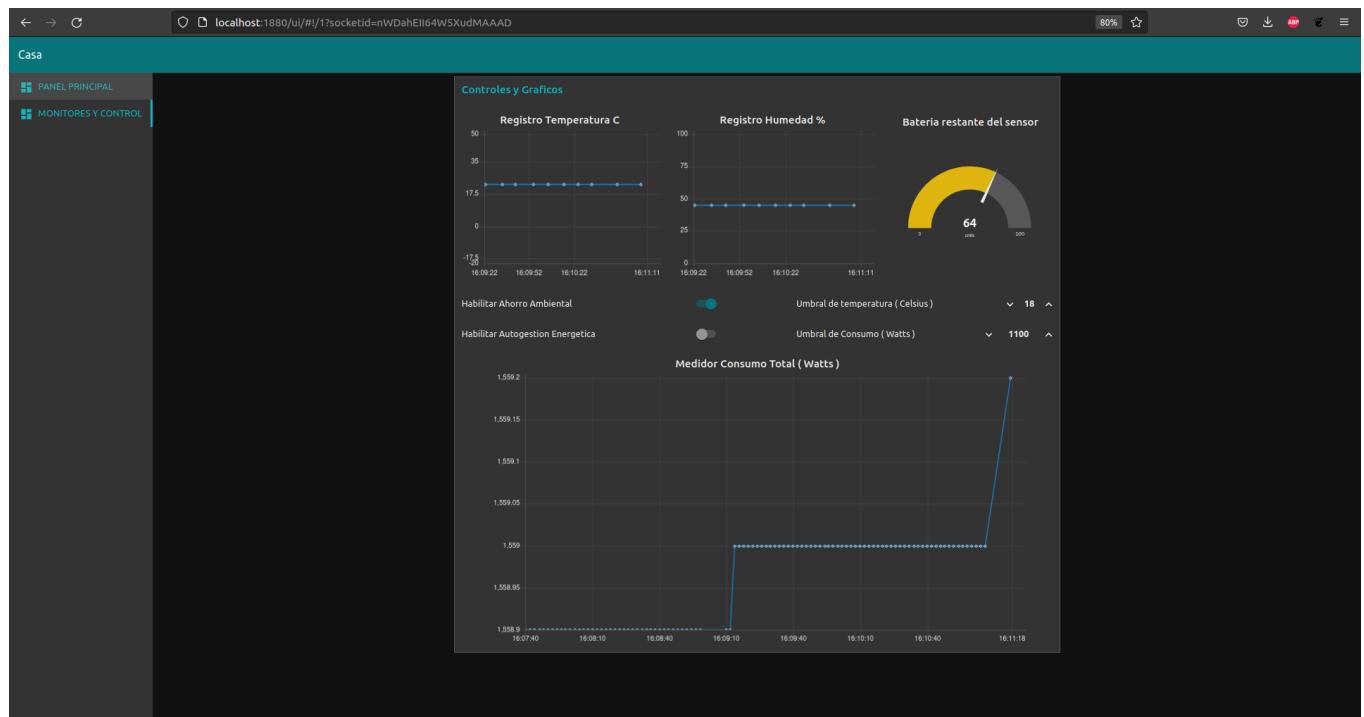
The main content area is organized into five columns:

- BOTONES**: A list of rooms and devices with toggle switches.
 - Dormitorio (On)
 - Comedor (On)
 - Pasillo (On)
 - Enchufe 1 (On)
 - Enchufe 2 (Off)
 - Medidor Consumo (On)
- ESTADO**: Shows green status indicators corresponding to the device's current state.
- CONSUMO**: Displays power consumption data for the Enchufe 1 and Enchufe 2 rows.
 - Enchufe 1: Potencia: 2.36
 - Enchufe 2: Potencia: 2.31
- TEMPERATURA DEL RELE**: An empty column.

At the bottom of the interface, there are four large, horizontally aligned buttons:

- APAGA CASA (Red)
- ENCIENDE CASA (White)
- LLEGANDO A CASA (Blue)
- MODO CINE (Yellow)

En la segunda ventana cubrimos los criterios 2 y 4, mostrar el consumo de la vivienda, también la temperatura y humedad, además de la batería del sensor, y habilitar / deshabilitar 2 algoritmos de ahorro de consumo, uno que apaga las luces si se supera cierto umbral de consumo personalizado por el usuario, y otro algoritmo que gestiona un enchufe (enchufe 2) que tendrá una estufa conectada, este enciende o apaga dicho enchufe dependiendo de la temperatura del ambiente, este valor también se puede personalizar por el usuario.



Documento a analizar.

El documento (TFG de Berbel Bueno) detalla en extensividad ejemplos y métodos para integrar varios dispositivos y tecnologías IoT entre si. En este, se puede observar ejemplos del despliegue de un servidor Node-red en una raspberry, el uso de MongoDB para la persistencia de datos, la configuración y interacción con un sensor SenseHat, detalles sobre la plataforma IWARE y desarrollo de una aplicación usando Android studio.

Mediante estas tecnologías, Berbel Bueno desarollo un sistema de recolección de datos mediante dispositivos IoT, usando geolocalización, automatizado el proceso, desde la cual se puede manejar mediante un panel o una aplicación móvil.

También, para demostrar el nivel de interacción el proyecto permite al usuario seleccionar una posición del mundo, y mostrar los twits de la zona.

Parte 2.

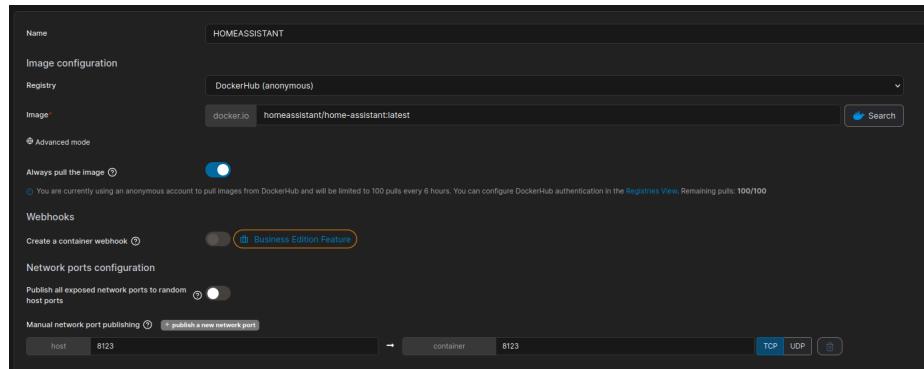
En esta segunda parte se nos pide 3 ejercicios:

- Instalar HomeAsistant, integrarlo con el sistema de la vivienda inteligente, y analizar su potencial.
- Desplegar una base de datos, y conectar nuestra parte 1 con una base de datos, para tener un registro de los datos de los sensores y eventos de la vivienda (para esto usaremos el nodo “node-red-contrib-influxdb”).
- Relacionar un QR con un formulario en local y registrar los datos de este formulario en base de datos. Para este apartado, usaremos los siguientes recursos:
 - <https://gitlab.cern.ch/mro/common/www/node-red/node-red-contrib-ui-svg>
 - https://www.industrialshields.com/es_ES/blog/raspberry-pi-para-la-industria-26/post/tutorial-de-node-red-como-generar-un-codigo-qr-con-raspberry-pi-plc-364
 - <https://nodered-dashboards.gitbook.io/node-red-dashboards/workshop/forms-basics>

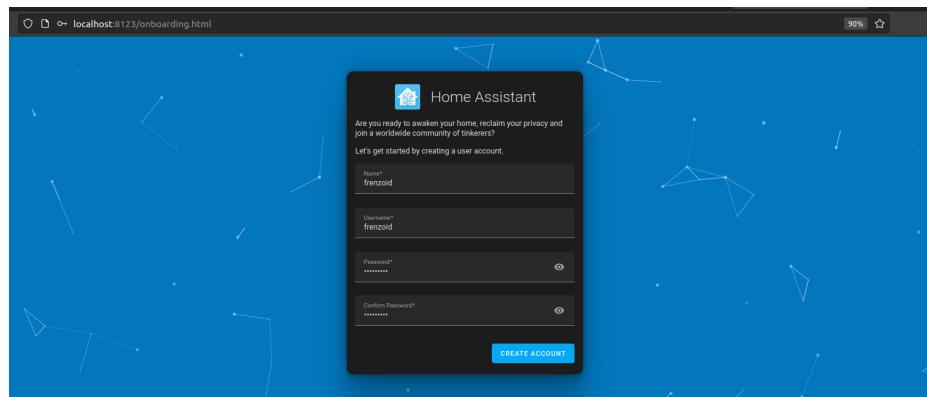
Ejercicio 1 - Home Assistant

Lo primero que vamos a hacer es instalar Home Assistant, usando docker, específicamente usaremos portainer para facilitarnos el despliegue de este contenedor.

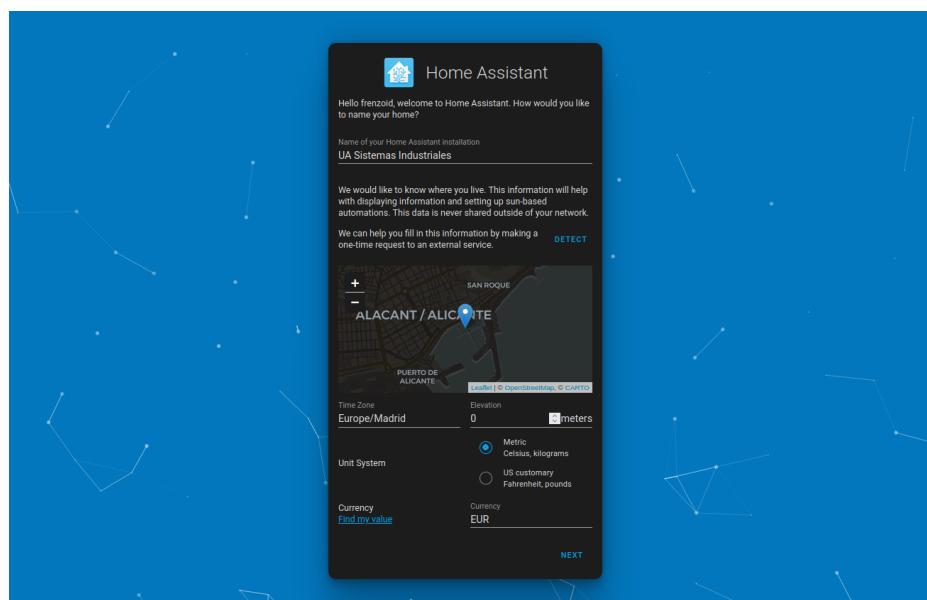
A la hora de configurar el contenedor, solo vamos a especificar el nombre del contenedor, la imagen, y el puerto a mapear (8123).



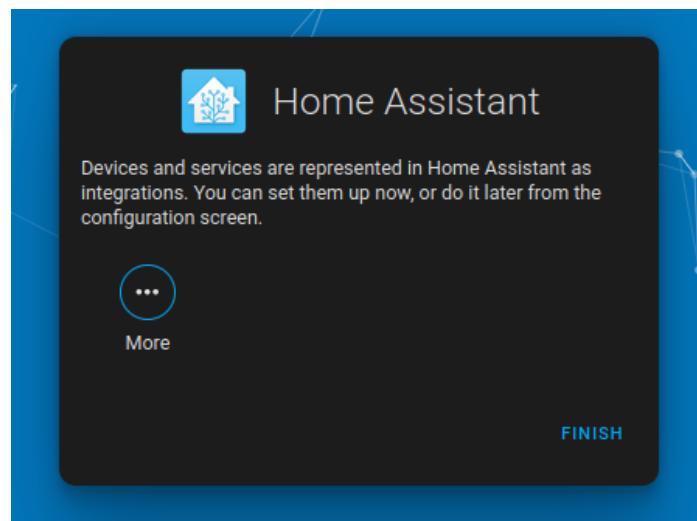
Una vez arrancado, nos pedirá registrarnos una cuenta:



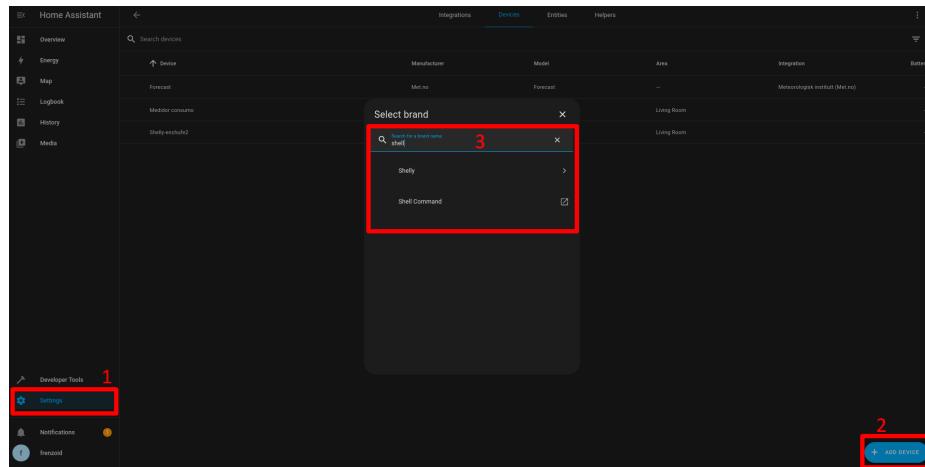
Configuramos el proyecto:



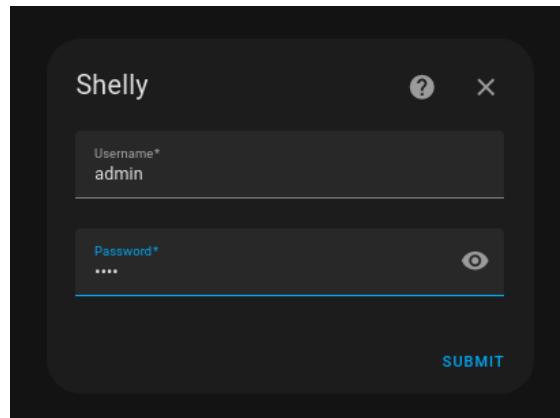
Y ya tendriamos configurado nuestro HomeAssistant. Tambien, podemos desde este panel añadir directamente los dispositivos conectados a nuestra red para enlazarlos a Home Assistant, o ya desde el la página principal.



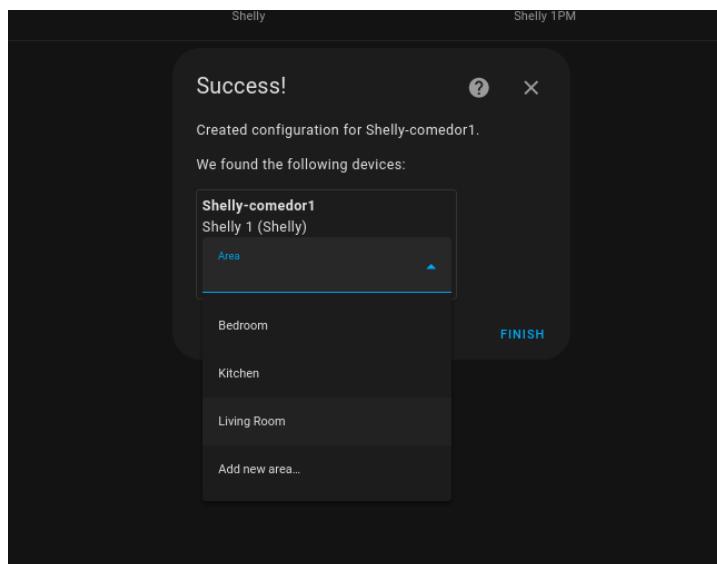
En la pantalla principal, podremos acceder a ajustes > añadir dispositivo, desde aquí se nos abrirá el menú para añadir dispositivos, y buscamos por la marca shelly



En el siguiente panel especificaremos la IP del shelly, y al conectarse, nos pedirá las credenciales:



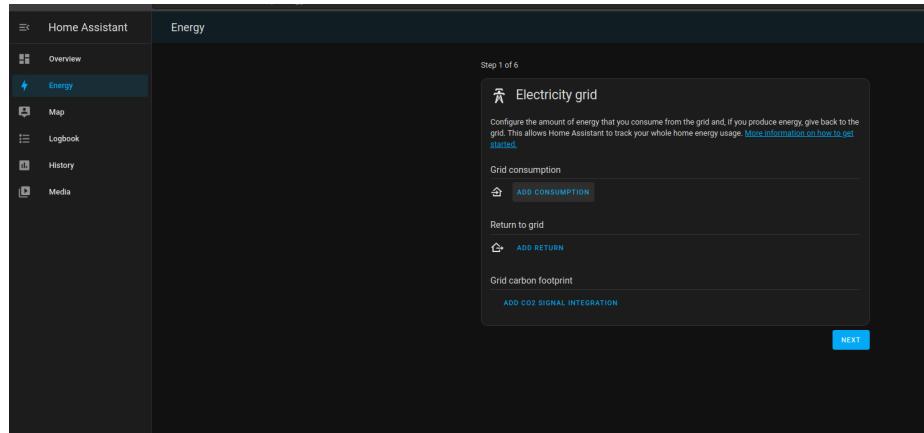
Una vez insertadas las credenciales, asignamos el dispositivo a una “zona”.



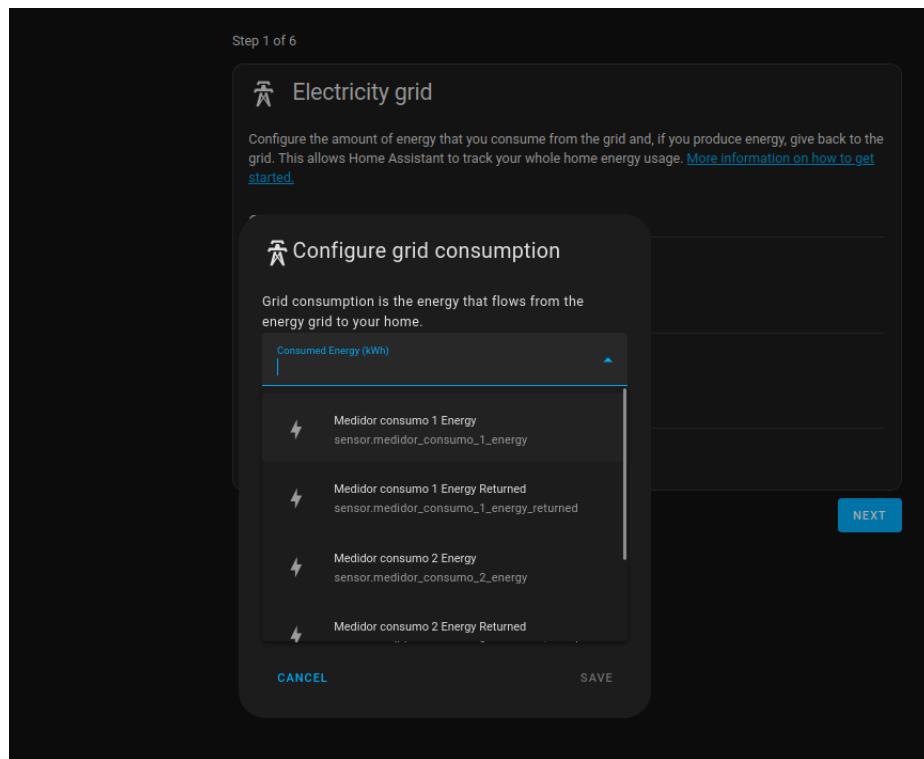
Realizamos este proceso hasta añadir todos los dispositivos.

El medidor de consumo tiene un proceso especial para ser añadido, ya que provee de datos de control y consumo electrico, para este dispositivo realizaremos el siguiente proceso:

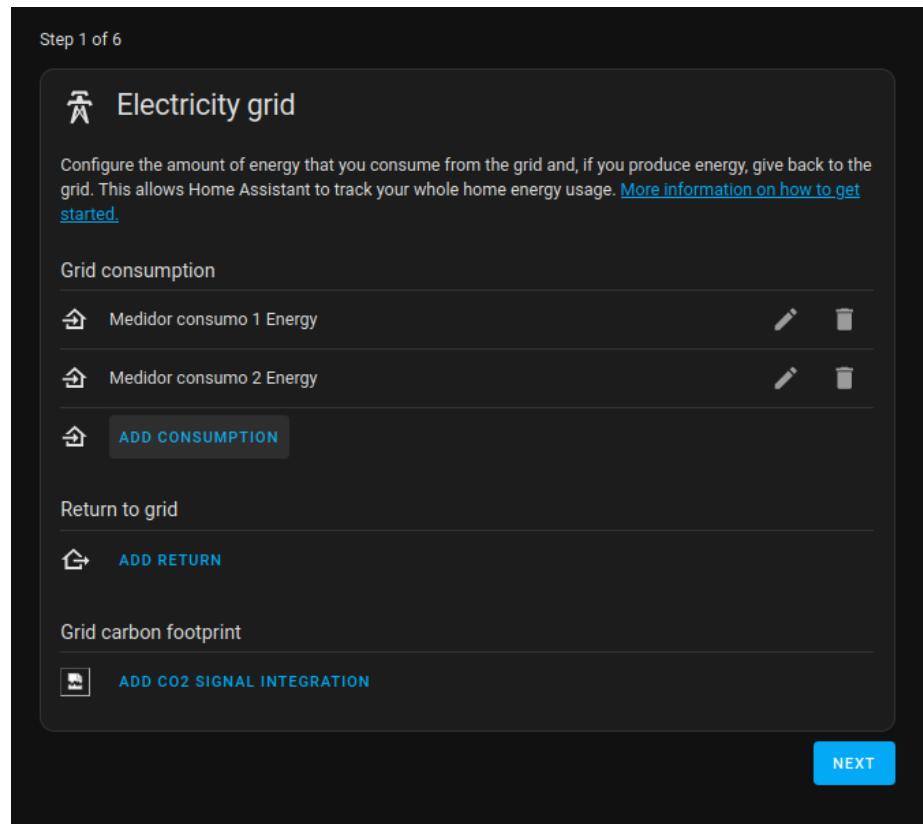
Vamos a Energy > Setup, y seguimos los pasos para añadir los consumos del medido de consumo.



En el panel de configuración de consumo electrico, añadimos las dos medidas del medidor de consumo.



En nuestro caso, deberá quedar de la siguiente forma:



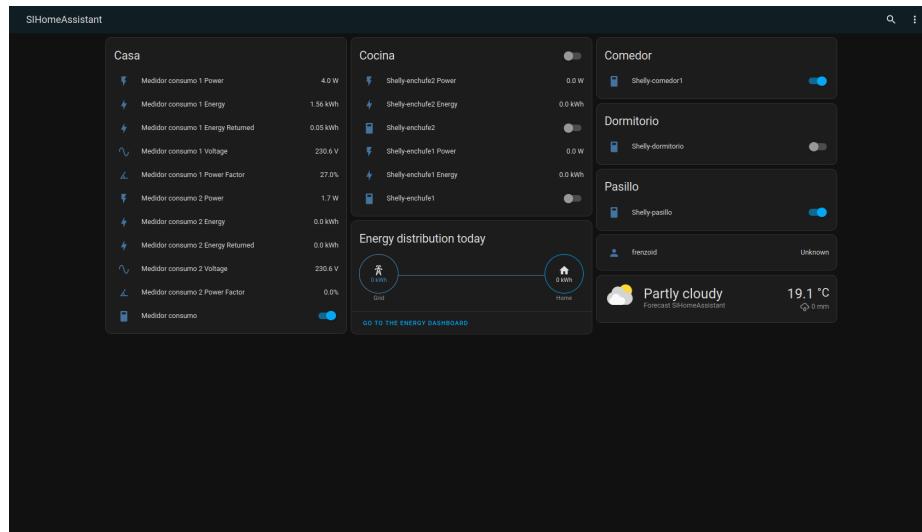
Los siguientes paneles hacen referencia a otras fuentes de energía, ignoraremos el resto de configuraciones apretando al botón “Next” hasta que finalice la configuración.

Una vez añadido todos los dispositivos, en logbook podremos ver los eventos de los dispositivos.

The screenshot shows the Home Assistant Logbook interface. The sidebar includes Overview, Energy, Map, Logbook (selected), History, Media, Developer Tools, Settings, Notifications, and frenzoid. The main area displays a log of events for December 5, 2022, from 2:00 PM to 5:00 PM. The events listed are:

- Shelly-comedor1 turned on 3:35:04 PM - 1 second ago
- Shelly-enchufe2 turned off 3:34:12 PM - 1 minute ago
- Shelly-enchufe2 became unavailable 3:34:12 PM - 1 minute ago
- Shelly-enchufe2 Overheating cleared (no detected) 3:34:12 PM - 1 minute ago
- Shelly-enchufe2 Overpowering cleared (no detected) 3:34:12 PM - 1 minute ago
- Shelly-enchufe2 became unavailable 3:33:59 PM - 1 minute ago
- Shelly-enchufe2 Reboot became unavailable 3:33:39 PM - 1 minute ago
- Shelly-enchufe2 Overheating became unavailable 3:33:39 PM - 1 minute ago
- Shelly-enchufe2 Overpowering became unavailable 3:33:39 PM - 1 minute ago
- Shelly-comedor1 turned off 3:32:52 PM - 2 minutes ago
- Shelly-dormitorio turned off 3:32:41 PM - 2 minutes ago
- Shelly-enchufe2 turned off 3:32:23 PM - 3 minutes ago
- Shelly-enchufe1 turned off 3:32:14 PM - 3 minutes ago
- Shelly-pasillo turned on triggered by service switch.turn_on 3:32:13 PM - 3 minutes ago - frenzoid

Tambien, en nuestro dashboard podremos ver y interactuar con los dispositivos añadidos. Tambien, nuestro dashboard puede quedar más ordenado dependiendo de las zonas creadas y de los dispositivos asignados a estas zonas.



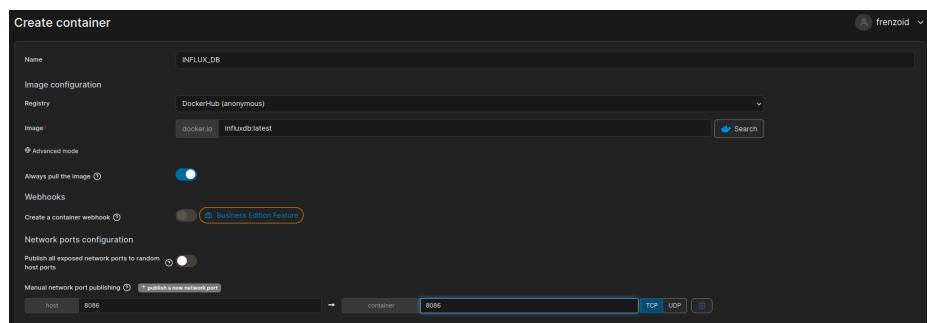
Por la experiencia de esta práctica, podemos asegurar que HomeAssistant es una herramienta imprescindible para automatizar una casa, desde el control de muchísimos dispositivos IoT (gracias a su alto potencial de integración con la mayoría de fabricantes), hasta gestionar y controlar el consumo de esta. Es por estos motivos, una de las herramientas más potentes que he conocido para la gestión domótica, que por encima es una variante self-hosted.

Ejercicio 2 - Interacción con una DB, y registro de los eventos de nuestra instalación domótica.

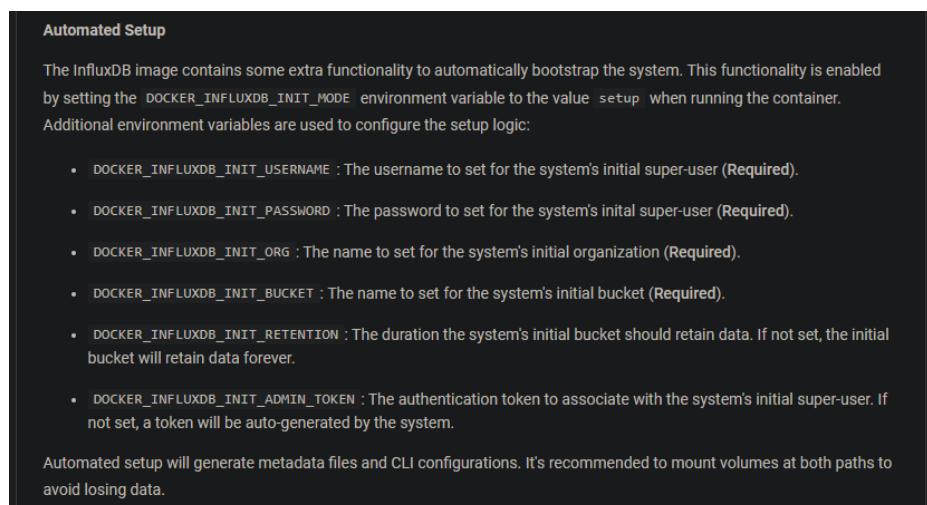
Para este objetivo, usaremos el nodo influxdb.

Que nos permite interactuar con bases de datos influxdb. InfluxDB es una base de datos que guarda datos por series de tiempos, por lo tanto difiere al esquema al que estamos acostumbrados de las bases de datos relacionales.

Primero antes de todo, vamos a desplegar la base de datos via docker, usando portainer.



La documentación de la imagen de docker de influxdb nos pide que especifiquemos mediante variables de entorno las credenciales y otros datos de configuración para el contenedor.

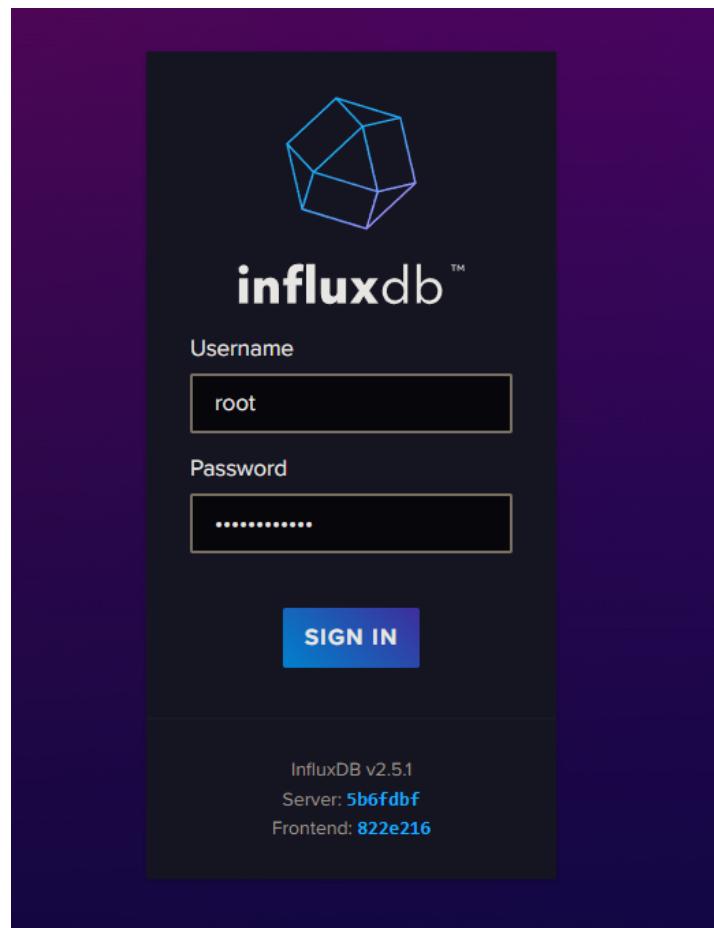


Para ello, desde el apartado “Env” añadimos las siguientes variables. (Tenerlas en cuenta para más adelante !)

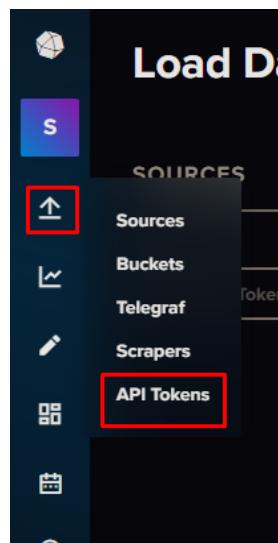
name	DOCKER_INFLUXDB_INIT_MODE	value	setup	<input type="button" value="Remove value"/>
name	DOCKER_INFLUXDB_INIT_USERNAME	value	root	<input type="button" value="Remove value"/>
name	DOCKER_INFLUXDB_INIT_PASSWORD	value	rootrootroot	<input type="button" value="Remove value"/>
name	DOCKER_INFLUXDB_INIT_ORG	value	SIND	<input type="button" value="Remove value"/>
name	DOCKER_INFLUXDB_INIT_BUCKET	value	SIND_B	<input type="button" value="Remove value"/>

Y desplegamos el contenedor.

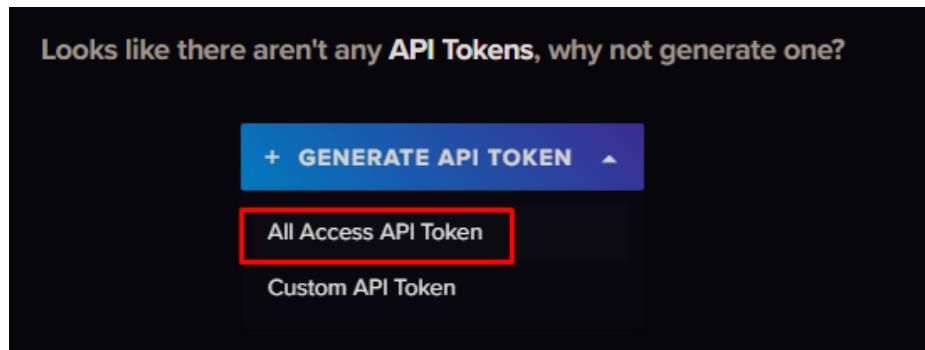
Accedemos a localhost:8086 y iniciamos sesión con las credenciales que hemos definido en las variables de entorno del contenedor.



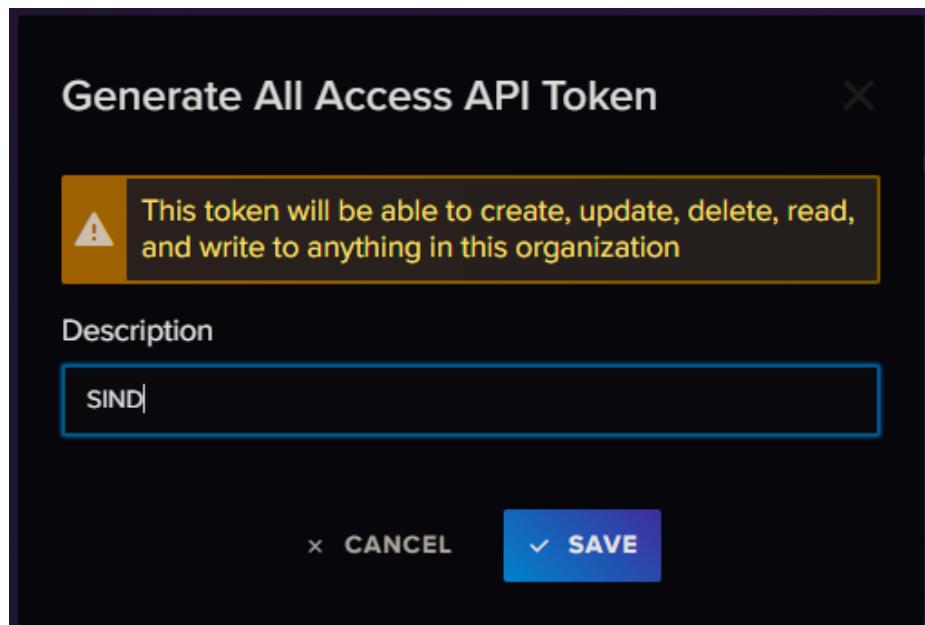
Omitimos el tutorial, y vamos a al apartado de API Tokens:



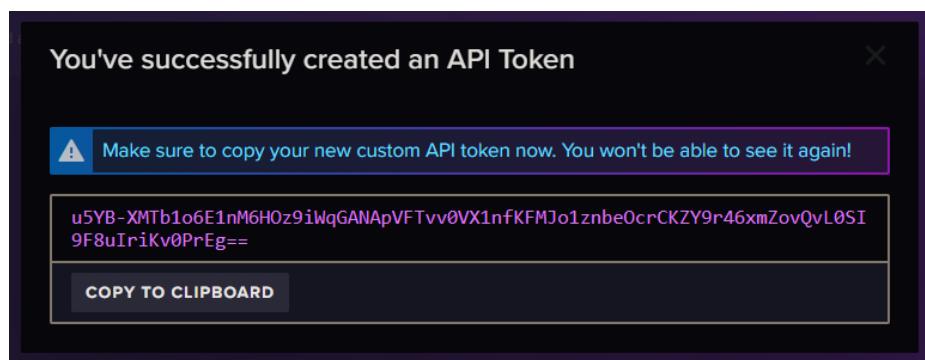
Y creamos un token con todos los permisos.



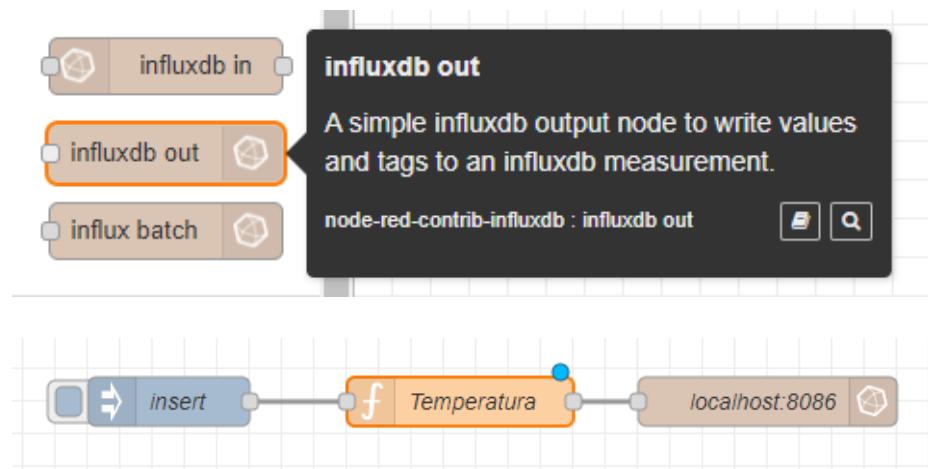
Le damos un nombre.



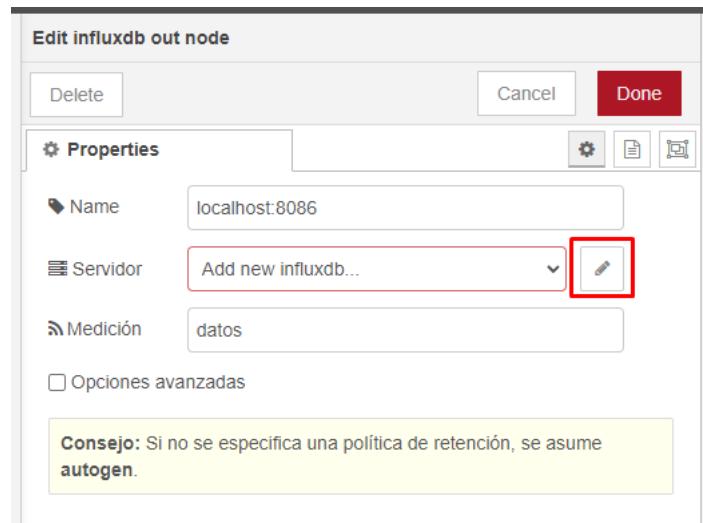
Y copiamos el token. Este token es lo que usaremos en nodered como credencial para acceder a la base de datos.



Ahora, en nodered, vamos a crear un nodo “influxdb out” para insertar datos, haremos una pequeña prueba para comprobar y familiarizarnos con este nodo.



Accedemos al nodo de “influxdb out” > Servidor: y añadimos un nuevo servidor:



Y rellenamos los datos para agregar un nuevo servidor. En el campo de Token, pegamos el token que hemos generado antes.



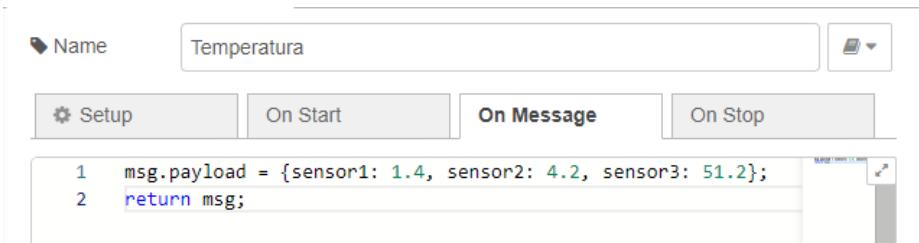
Una vez añadido el servidor, volvemos, y ahora nos aparecerán campos nuevos, especificamos la organización y la “medida” que en el caso de influxdb es un campo discriminante para asociar datos insertados. (Bucket corresponde con el nombre dado al bucket que se creará por defecto mediante la variable de entorno especificada anteriormente).



Ahora en nuestro flujo:



En la función tenemos los siguiente:



Ahora, ejecutamos el inject “insert”, y vamos a localhost:8086 > Data Explorer, ponemos la vista de “tabla simple”, seleccionamos el bucket que hemos puesto al principio en las variables de entorno de docker que es la misma que hemos puesto en el nodo de nodered, seleccionamos la _measure que hemos especificado en el nodo, y a continuación los campos que concuerdan con las claves que hemos definido en la función.

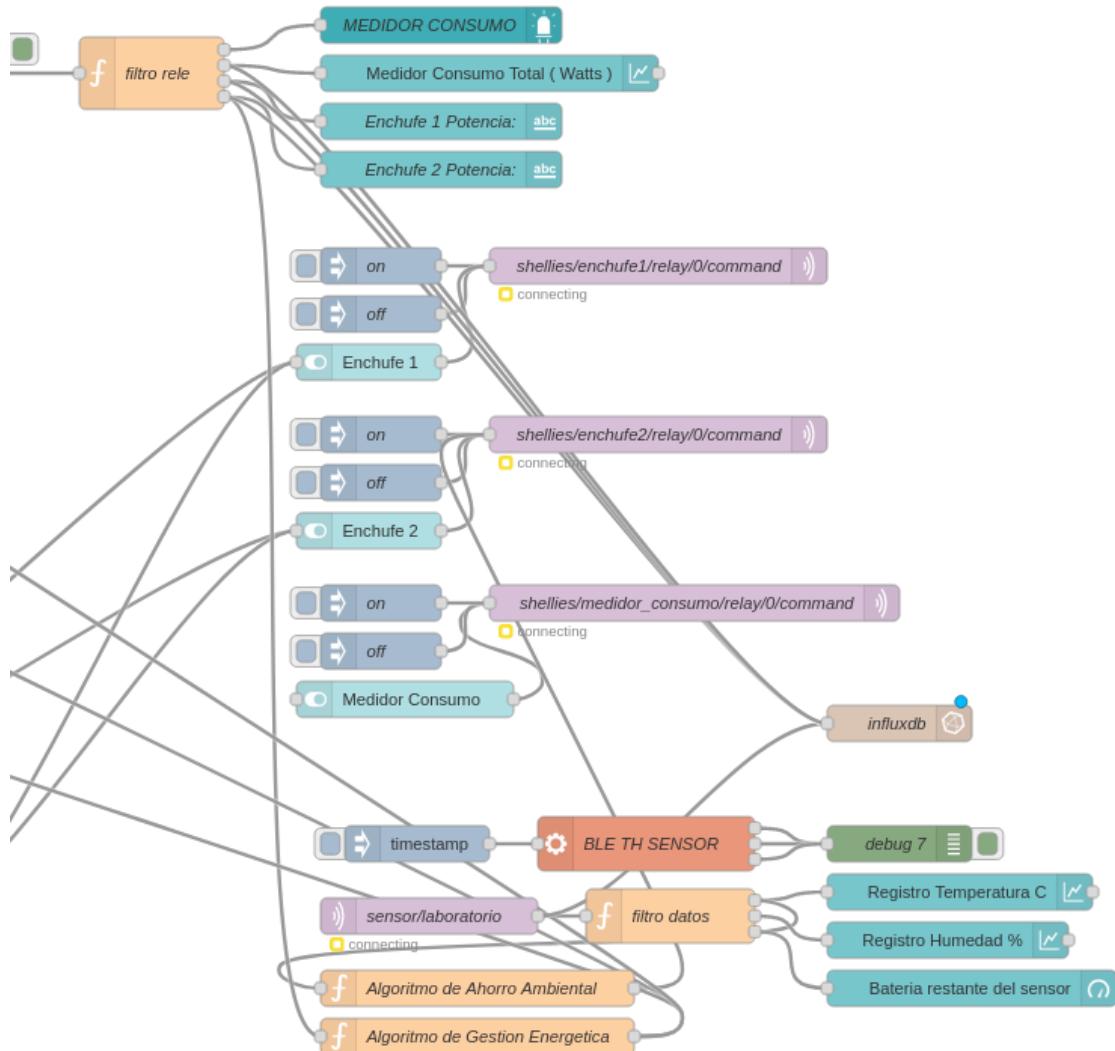
Estos son los parámetros de nuestra petición, ahora para solicitar los datos pulsamos en “Submit” y podremos ver en la tabla los datos insertados via nodered.

The screenshot shows the Grafana Data Explorer interface. The top navigation bar has a 'S' icon (1) and a dropdown menu set to 'Simple Table' (2). The main area displays a table with three rows of data:

table	_measurement	_field	_value	_time
mean	group	group	no group	dateTime:RFC3339
0	datosejemplo	sensor1	1.4	2022-12-11T17:23:50.000Z
1	datosejemplo	sensor2	4.2	2022-12-11T17:23:50.000Z
2	datosejemplo	sensor3	51.2	2022-12-11T17:23:50.000Z

The bottom section shows the 'Query 1 (0.12s)' editor. The 'FROM' clause (3) contains the bucket 'SIND_B'. The first filter (4) is applied to the '_measurement' field, with 'datosejemplo' selected. The 'SCRIPT EDITOR' tab (5) is active, and the 'SUBMIT' button is highlighted with a red box.

Ahora, para la práctica, aplicaremos este ejemplo para el dispositivo de medición de consumo, la temperatura y la humedad. En nuestro flujo, hemos añadido el nodo de influxdb, y insertamos 6 campos: temperatura, humedad, %bateria (sensor de temperatura), consumo enchufe 1, consumo enchufe 2, consumo total.



Y ahora, si vamos al panel de Data Explorer, podremos ver estos datos.

The screenshot shows the Data Explorer interface with a table of data. The columns are: _table, _measurement, _field, _value, and _time. The data includes rows for average and battery values over time. The interface has a sidebar with filters and a query editor at the bottom.

_table	_measurement	_field	_value	_time
result	group	string	no group	dateTime:RFC3339
0	datossensores	average	3	2022-12-12T14:14:56.726Z
0	datossensores	average	3	2022-12-12T14:14:58.726Z
1	datossensores	battery	62	2022-12-12T14:14:56.726Z

Query 1 (0.04s)

FROM: SIND_B

Filter: _field

Filter: _field

WINDOW PERIOD: AUTO

SCRIPT EDITOR

SUBMIT

Ejercicio 3 - Formulario con QR y Base de datos.

A continuación, para este último ejercicio, vamos a instalar los siguientes nodos, y vamos a hacer un pequeño ejemplo de cada uno:

The screenshot shows the GitHub repository for Node-RED-contrib. It lists three packages: contrib-ui-svg, node-red-contrib-qrcode-generator, and node-red-dashboard. The first two are listed as installed, while the third is shown as having been installed.

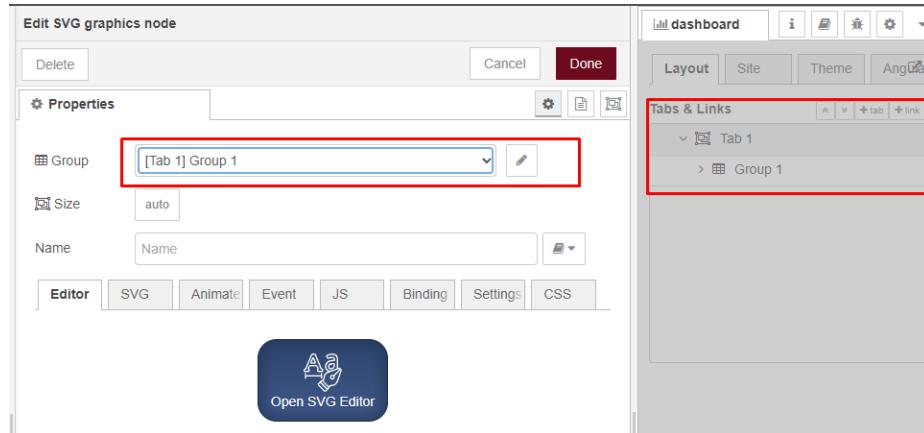
- contrib-ui-svg**: A Node-RED widget node to show interactive SVG (vector graphics) in the dashboard. Version 2.3.1, installed 1 year, 1 month ago.
- node-red-contrib-qrcode-generator**: QRCode Generator. Version 0.4.0, installed 1 year, 10 months ago.
- node-red-dashboard**: A set of dashboard nodes for Node-RED. Version 3.2.3, installed 2 weeks ago.

Ejemplo SVG.

Vamos a importar un nodo SVG.

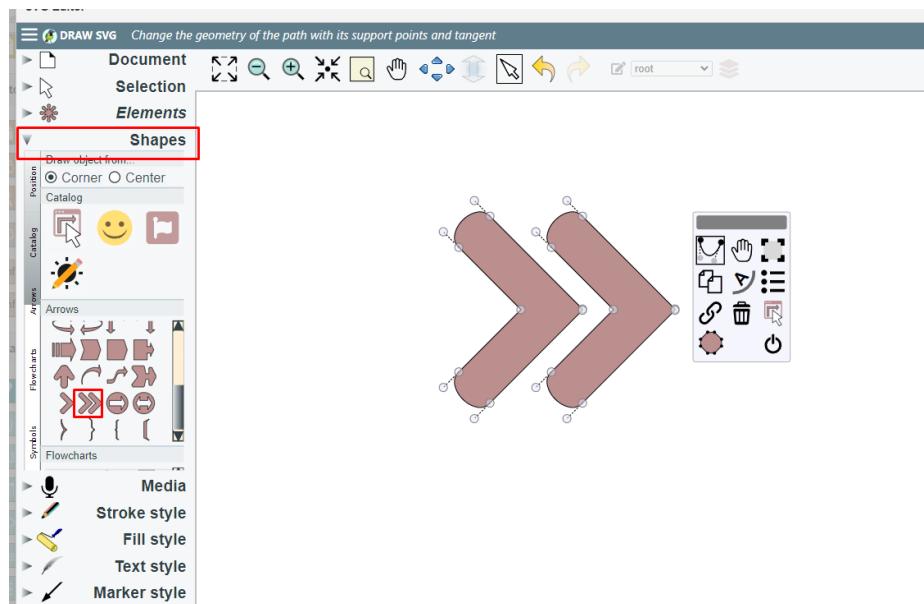


Vamos a asignarle un grupo, ya que este nodo sigue formado parte del conjunto de UI, y abrimos el editor.

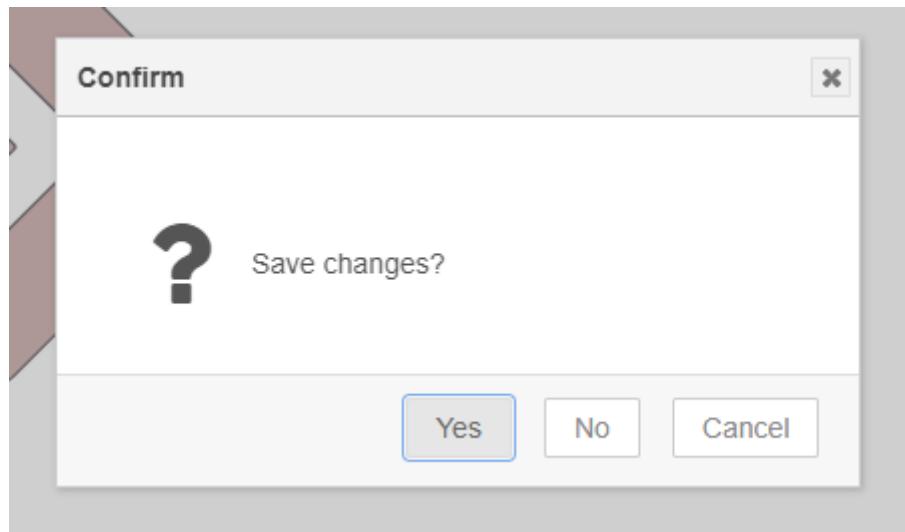


El editor es similar a la aplicación paint, nos permite subir imágenes, editarlas, y guardarlas para mostrarlas posteriormente en nuestra página web.

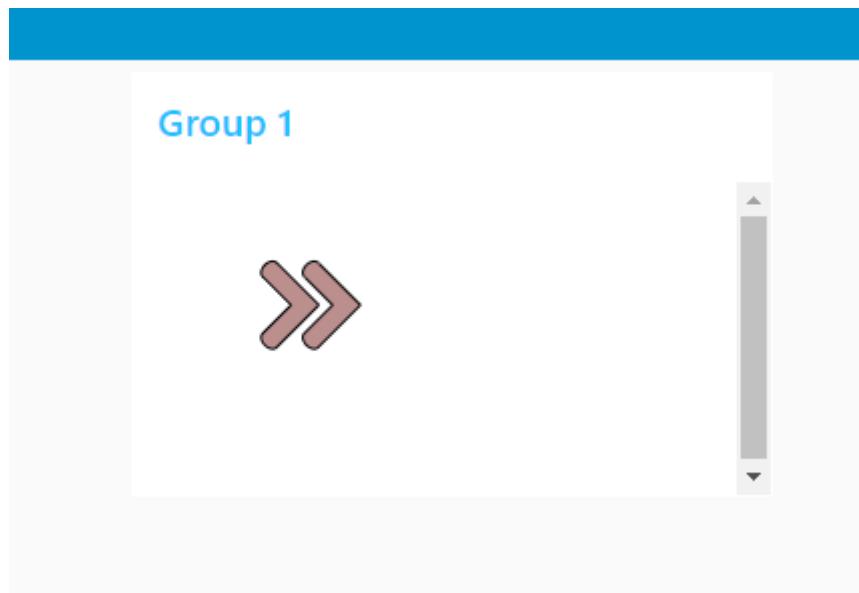
Yo voy a añadir una figura.



Y al salir, guardamos.



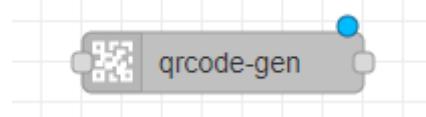
Y ahora, en la interfaz, podemos ver la imagen.



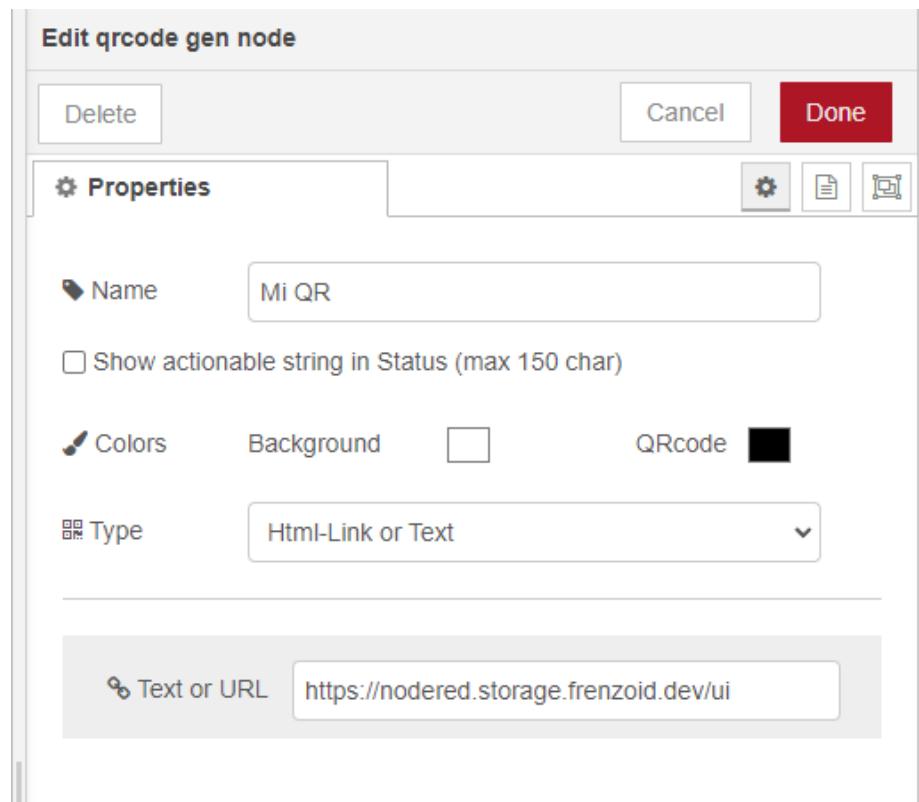
Con este nodo podemos animar, crear dibujos, mapas, importar imágenes, y mucho más para integrar estos dibujos en nuestra interfaz de node red.

Ejemplo QR.

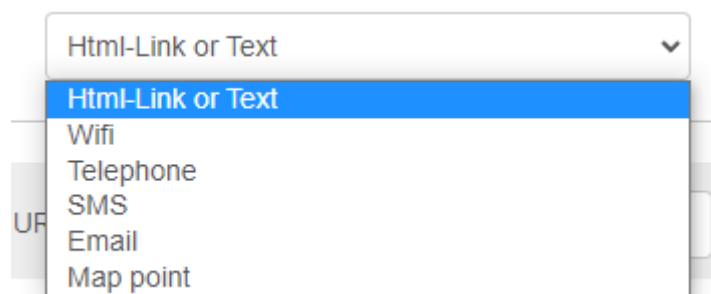
Vamos a importar un nuevo nodo QR.



Veamos su configuración.



Como podemos ver, la configuración es bastante sencilla.



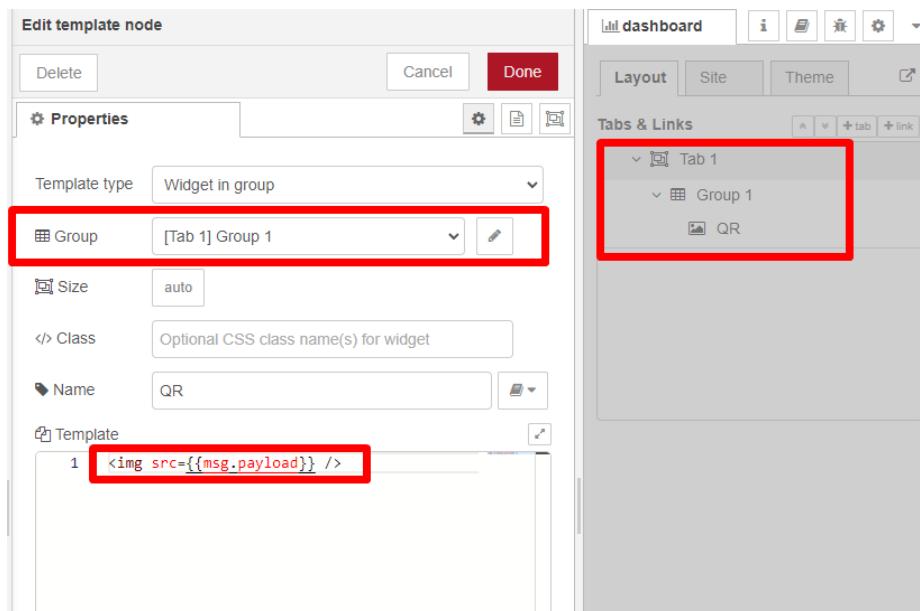
Y tenemos varios formatos desde el cual crear este QR.

Ahora, para mostrarlo en nuestra interfaz, deberemos enganchar nuestro nodo QR un inject para generar el QR, y un tag , para ello nos importamos un nodo "template".

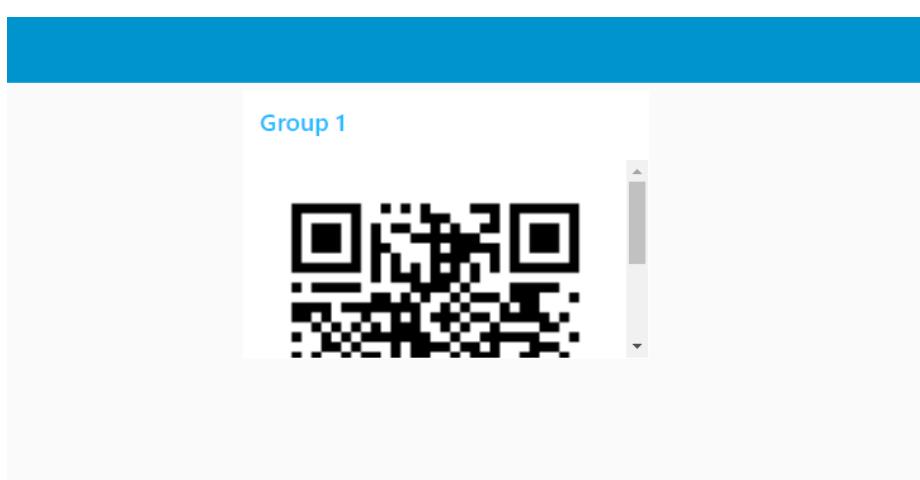


Este nodo template, nos permitirá renderizar HTML en cualquier tab / grupo de nuestra ui a la cual asociemos.

Para enganchar el QR a un tag , abriremos la configuración del nodo template, y añadimos los siguientes:

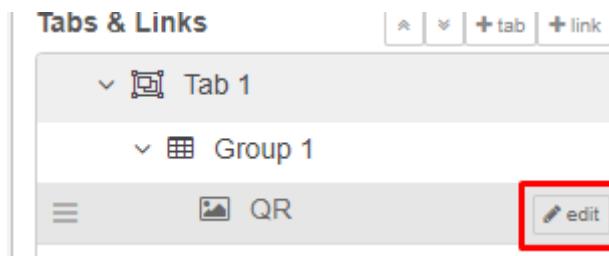


Y ahora, en nuestra UI podremos ver el QR.



Oops, para que el QR es más grande que el tamaño asignado automáticamente, vamos a arreglar eso.

Vamos al panel lateral > Dashboard > QR > edit:



Y ahora, en Size, 6x6:

Template type: Widget in group

Group: [Tab 1] Group 1

Size: 6 x 6

Class: </>

Name:

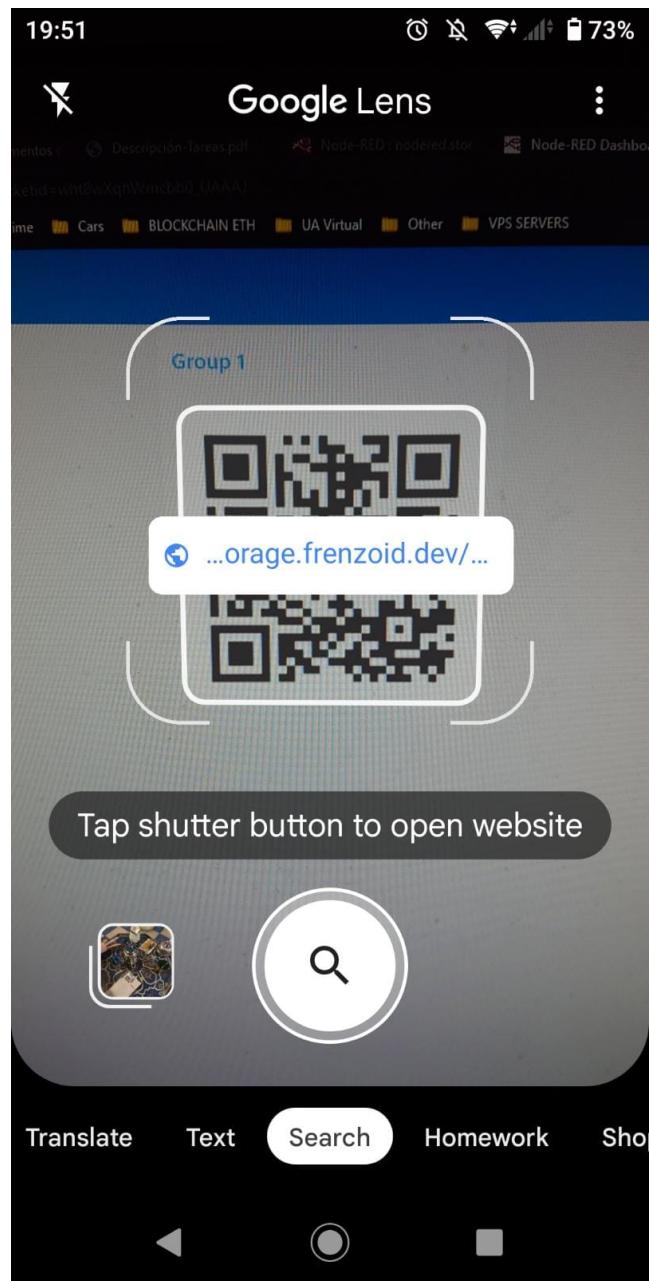
Template:

1	
---	-------

Y ahora sí, podemos ver el QR correctamente.



Y por último vamos a probarlo desde el móvil, usando google lens.



Como conclusiones se ha de mencionar que ha sido bastante sencillo de usar y de integrar, esta funcionalidad tiene mucho potencial porque se puede usar incluso para implementar mecanismos 2FA.

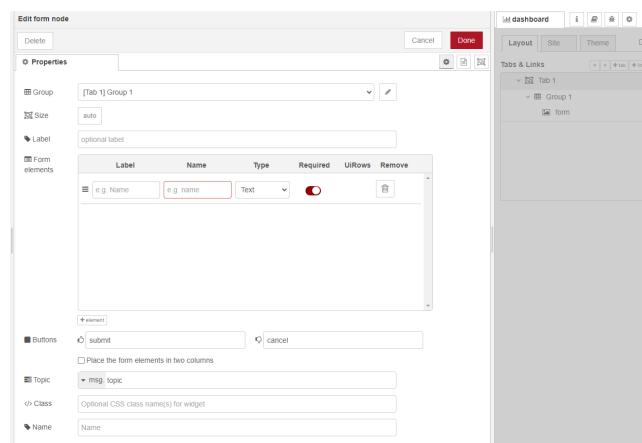
Ejemplo formulario.

Importamos un formulario de los nodos de nodered dashboard.

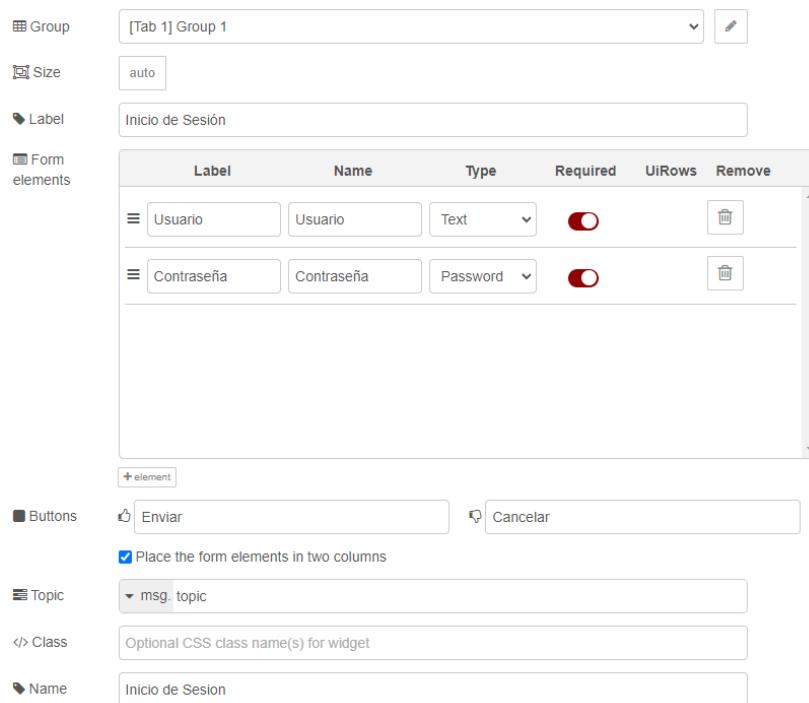


Vamos a revisar su configuración.

Por lo que podemos ver, este nodo nos facilita muchísimo la generación de formularios complejos con varios campos, ya que automáticamente nos genera una UI con los campos que especifiquemos, ahorrando mucho tiempo al hacerlo usando nodos individuales.



Vamos a crear un formulario de log-in.



Y ahora, vamos a ver como ha quedado en la UI.

The screenshot shows a user interface titled "Group 1". It contains two input fields: "Usuario *" with the value "frenzoid" and "Contraseña *" with a masked value consisting of twelve dots. Below the inputs are two blue buttons: "ENVIAR" on the left and "CANCELAR" on the right.

Inicio de Sesión	
Usuario *	Contraseña *
frenzoid	*****

ENVIAR **CANCELAR**

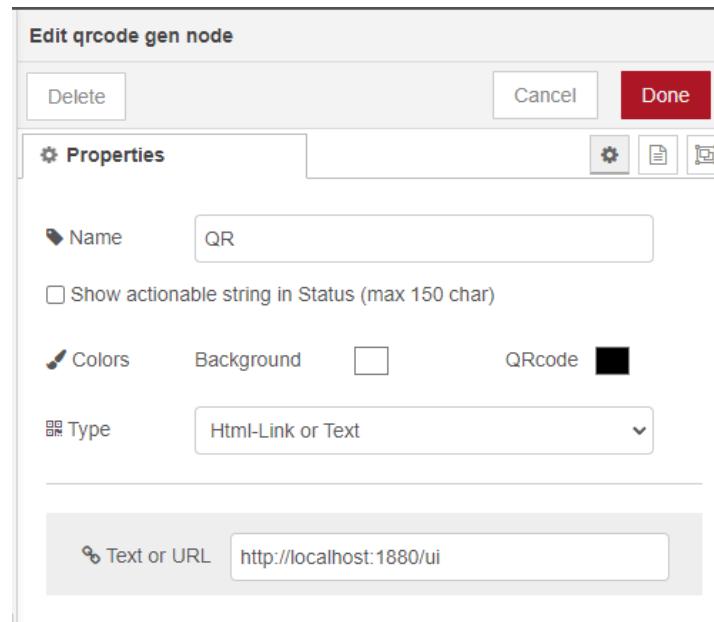
El resultado es fantástico, una UI limpia, sin mencionar la sencillez para crearla.

Ejercicio 3

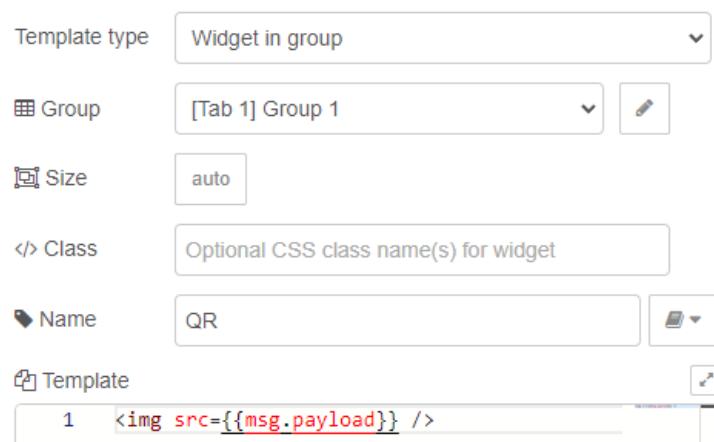
Para este ejercicio vamos a crear:

- Un nodo QR que apunte a la URL local de nuestra interfaz.
- Un nodo template para renderizar el QR.
- Un nodo Formulario que obtenga los datos de un usuario, en concreto su nombre, su edad, y su peso.
- Un nodo influxdb out la cual usaremos para insertar los datos recibidos por el nodo del formulario (estos datos ya vienen formateados en un objeto).

Primero, el nodo QR:



Y el nodo Template:



Los juntamos, y añadimos un inject, desplegamos y ejecutamos el inject.



Ahora, vamos a crear otro grupo en nuestro dashboard, y un formulario con los campos mencionados anteriormente. Asociamos el formulario al nuevo grupo.

Group [Tab 1] Group 2 auto Datos del cliente

Label	Name	Type	Required	UiRows	Remove
Nombre	nombre	Text	<input checked="" type="checkbox"/>		Delete
Edad	edad	Number	<input checked="" type="checkbox"/>		Delete
Peso (kg)	peso	Number	<input checked="" type="checkbox"/>		Delete

este nombré será la clave con la cual se guardará el valor:
{nombre: "blabla", edad: xx, peso: xx }

+ element Guardar Cancelar

Place the form elements in two columns

Topic msg. topic

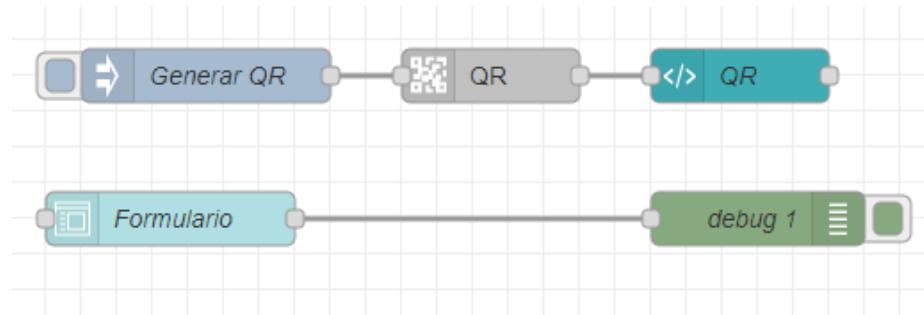
Class Optional CSS class name(s) for widget

Name Formulario

Ahora, nuestro dashboard se verá así:



Y nuestro flujo así:



Vamos a interactuar con el formulario:

```
11/12/2022, 20:28:26  node: debug 1
msg.payload : Object
▶ { nombre: "frenzoid", edad: 25, peso:
70 }
```

Perfecto, ahora vamos a añadir el nodo influxdb. La medición en este caso será “datosformulario”.

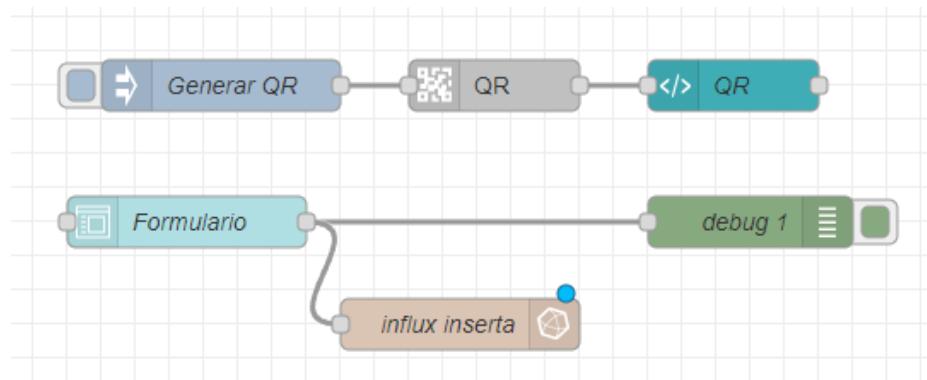
Edit influxdb out node

Delete Cancel Done

Properties

Name	influx inserta
Servidor	[v2.0] localhost
Organización	SIND
Bucket	SIND_B
Medición	datosformulario
Precisión	Milliseconds (ms)

El flujo quedará así:



Vamos a insertar un datos, y a revisar si existe en el data explorer.



Datos del cliente

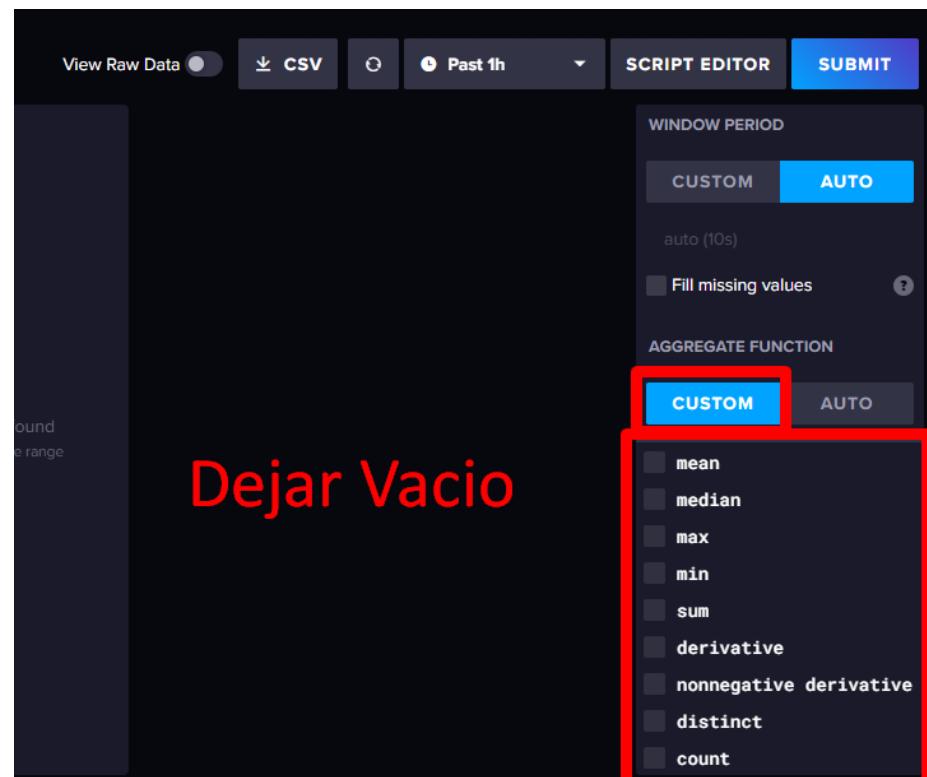
Nombre *
Samuel

Edad *
24

Peso (kg) *
84.2

GUARDAR **CANCELAR**

Y ahora, en el data explorer, antes de darle a select para ver los datos, tendremos que deshabilitar las funciones agregadas por defecto. Estas funciones sirven para aplicar automáticamente funciones de media, mediana, etc.. a los datos, pero en nuestro caso hay un campo que es de tipo “string”, y el select fallaría. Para solventar esto, tendremos que deshabilitar los aggegators, Data explorer > Derecha Abajo > Aggregate Function > Custom > Desmarcar todo.



View Raw Data CSV Past 1h **SCRIPT EDITOR** **SUBMIT**

WINDOW PERIOD

CUSTOM **AUTO**

auto (10s)

Fill missing values

AGGREGATE FUNCTION

CUSTOM **AUTO**

- mean
- median
- max
- min
- sum
- derivative
- nonnegative derivative
- distinct
- count

Y ahora, al ejecutar el select, podremos ver todos los datos de los campos seleccionados.

The screenshot shows the Data Explorer interface with the following details:

- Table View:** Shows three rows of data from the "datosformulario" measurement. Row 0: edad=24, nombre=Samuel, peso=84.2. Row 1: edad=24, nombre=Samuel. Row 2: edad=24, nombre=Samuel.
- Query Editor:**
 - FROM:** SIND_B
 - Filter:** Three dropdowns for measurement, field, and measurement again, each with three selected items: edad, nombre, and peso.
 - Result Panel:** Displays the query results with the message: "No tag keys found in the current time range".
 - Buttons:** View Raw Data, CSV, Past 1h, Script Editor, and Submit.
 - Settings:** WINDOW PERIOD (CUSTOM/AUTO), auto (10s), Fill missing values, and AGGREGATE FUNCTION.

Conclusion

Durante esta segunda parte hemos aprendido muchísimo, usando una casa inteligente como escenario hemos desarrollado un proyecto para automatizar los dispositivos IoT del entorno. También nos hemos familiarizado con plataformas como HomeAssistant, y hemos aprendido a integrar varios nodos diferentes como QR, renderizado de imágenes y el salvado de datos en una base de datos, despliegue e interacción con esta.