

Desarrollo visual en Sofia2 con Raspberry Pi, Node-RED y dashboards

Taller IoT

28-03-2017

Powered by



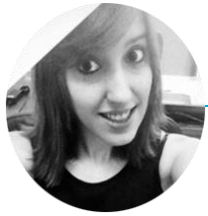
 http://twitter.com/SOFIA2_Platform

 <http://about.sofia2.com>




@ plataformasofia2@indra.es

 <http://sofia2.com>

Hoy presentamos...






Raquel Barrio

 rbarrio@minsait.com
 rbarriov
 @raquel_barrio

IoT Architect
Sofia2 IoT Platform



Jesús Fernandez

 jfgpimpollo@minsait.com
 jesus-fernández-gómez-pimpollo-253b2593
 @jfdezgomez

IoT Architect
Sofia2 IoT Platform

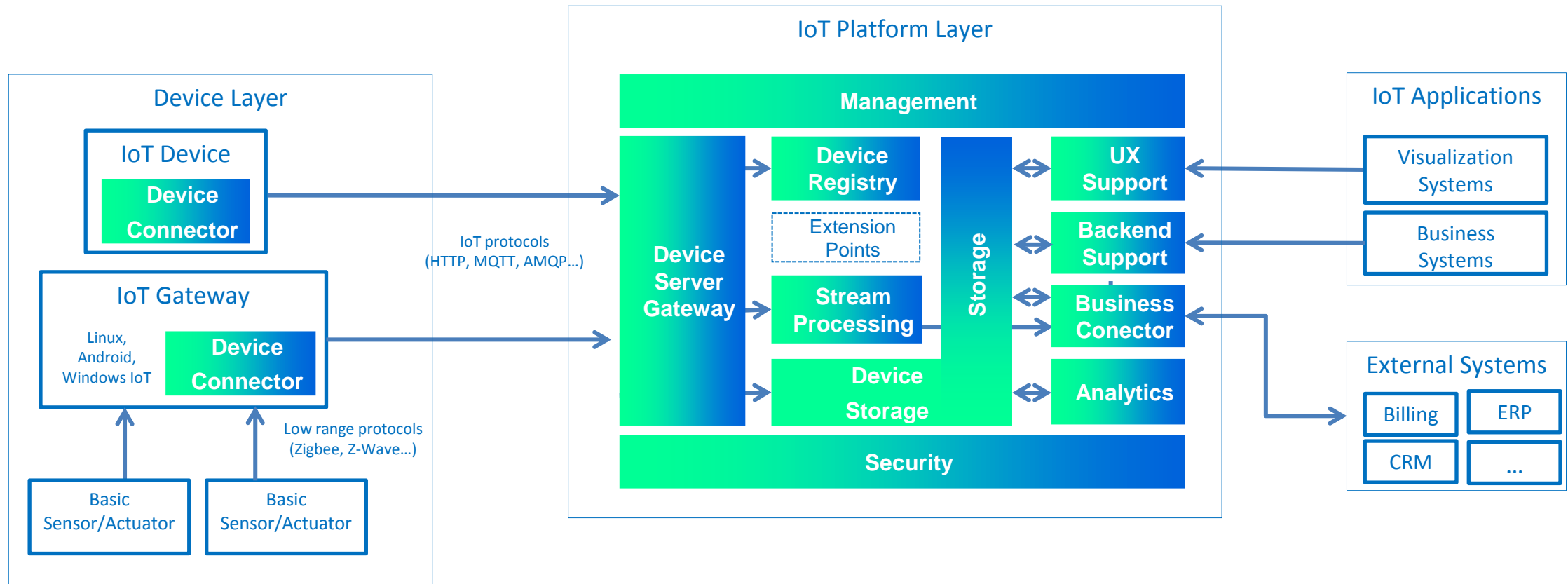
Y queremos contaros...

1. Componentes de Arquitectura IoT (19:00 a 19:15)
2. Demos (19:15 a 19:30)
3. Taller IoT (19:30 a 20:30)
4. Preguntas
5. Pizzas y cañas! (20:35)

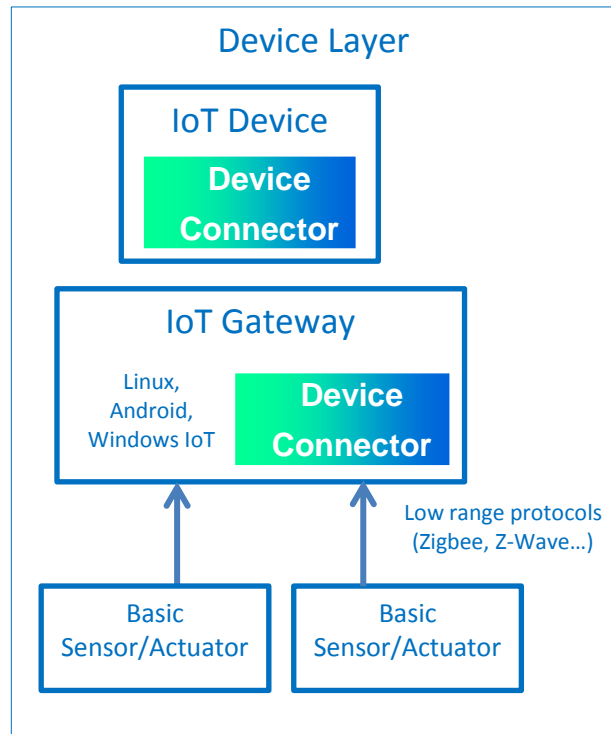
Capítulo 1

Componentes de arquitectura IoT

Arquitectura de referencia IoT



Tipos de Dispositivos IoT



Cualquier dispositivo con conexión a internet puede enviar información y ser comandado. Existen dos aproximaciones principales:

- **Dispositivos integrados con propósito específico:** Se trata de dispositivos que integran los sensores y/o actuadores así como la lógica para enviar/recibir información con la plataforma IoT. Tienen el propósito para el que se fabricaron, no se pueden extender con nuevos sensores, y las actualizaciones de software son para corregir bugs o mejorar rendimiento.
- **Gateways IoT:** Se trata del caso mas habitual. Son dispositivos con procesador (ARM Cortex, Intel Atom...), memoria RAM, almacenamiento en SD, Sistema Operativo, conexión a internet, conexiones GPIO y en algunos casos una o varias interfaces de red PAN o LAN.
Normalmente los sensores no se conectan directamente a internet, sino que se conectan a los pines GPIO de una placa o integran una radio PAN o LAN de bajo coste, para enviar tramas a un Gateway IoT, que es quien las procesa, integra en un mensaje mayor y envía a la plataforma IoT.

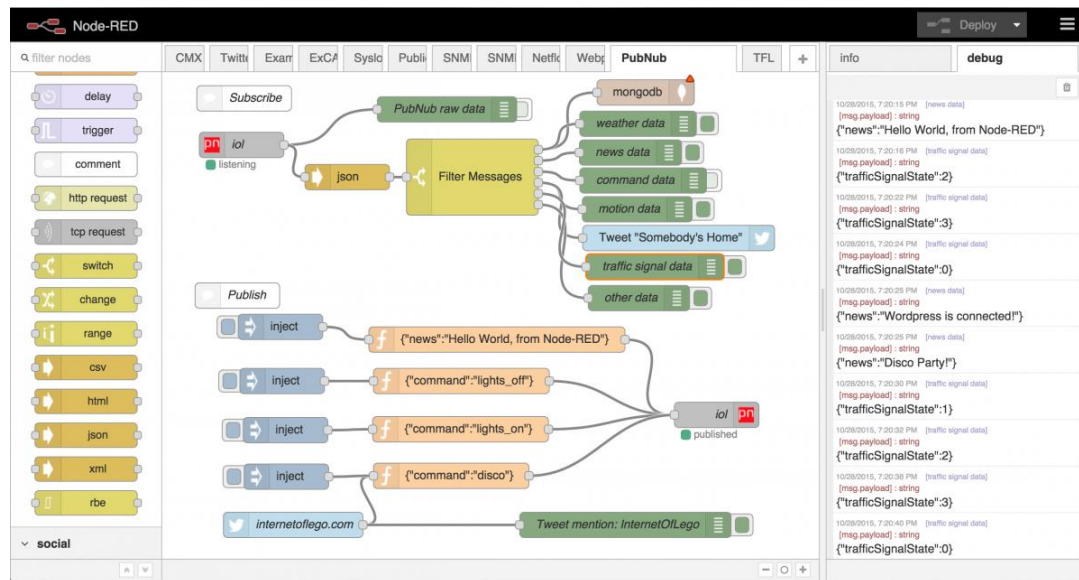
Permiten actualizaciones remotas de software, modificaciones y extensiones la arquitectura de sensores/actuadores, autonomía y almacenamiento de información en modo desconectado.

Node-RED

<https://nodered.org>

Motor de flujos visual, desarrollado por IBM y distribuido como open source. Está desarrollado en NodeJS y es muy ligero pudiendo ejecutarse en dispositivos de capacidades muy reducidas.

Los flujos se construyen mediante la **conexión de nodos** que se ejecutan secuencialmente pudiendo pasar desde un nodo a otro un mensaje con datos que son procesados a lo largo del flujo.



Enfoque IoT: Existen nodos para conectar casi cualquier cosa a un flujo:

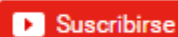
- Pines GPIO, Puerto serie
- Protocolos de alto nivel: MQTT, AMQP, REST, Socket TCP, SMTP...
- Redes sociales y APIs de internet: Twitter, Yahoo! Weather, Telegram...

Abierto, extensible y colaborativo: Cualquiera puede desarrollar sus propios nodos y flujos y hacerlos disponibles al resto de usuarios en los repositorios de npm.

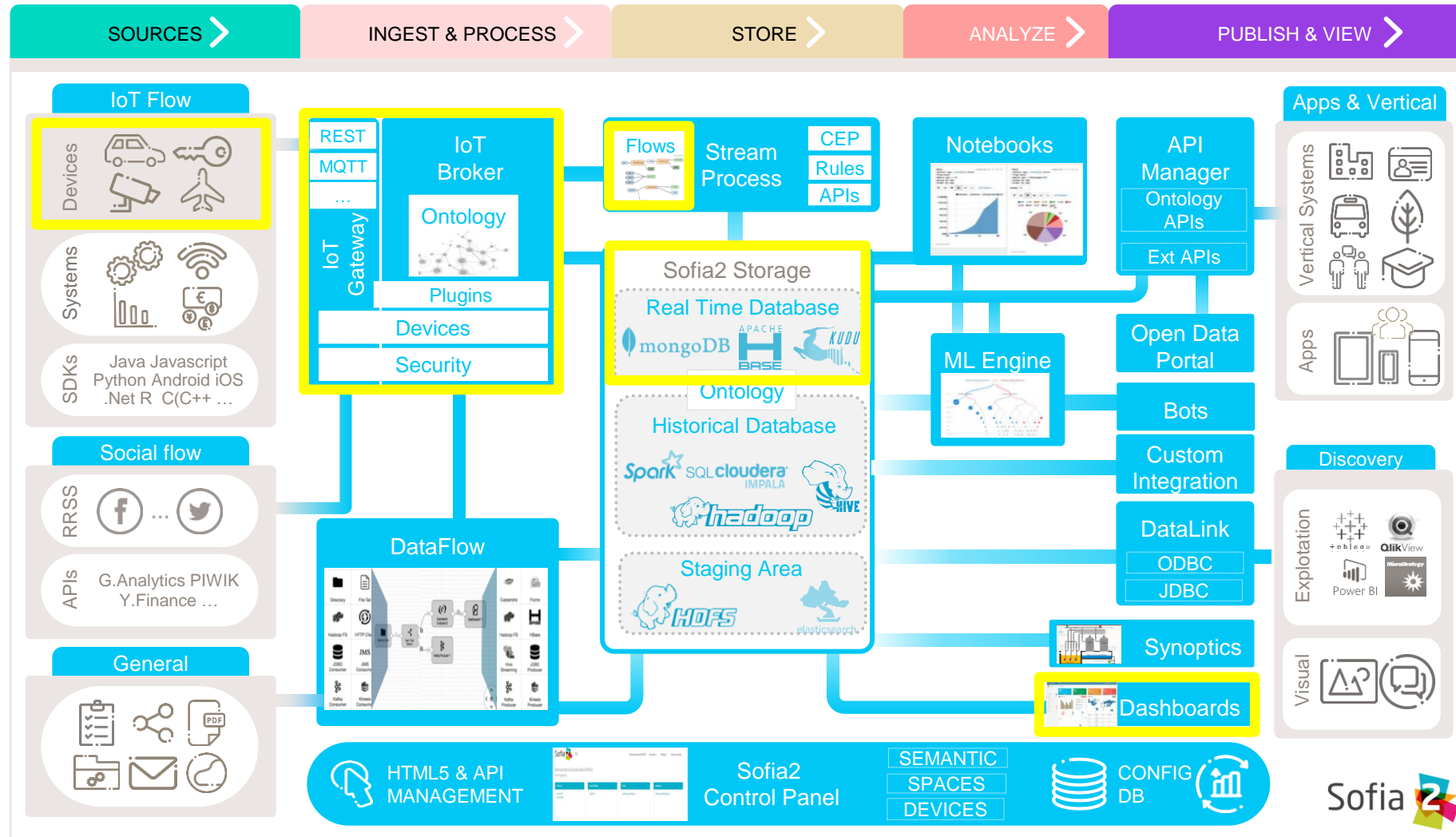
Tiene aplicación tanto en dispositivos para construir aplicaciones visuales, como en plataforma IoT para tratar, procesar o enriquecer la información de forma sencilla (Sofia2, Bluemix, Sensetecnic).

Podéis encontrar mas información en nuestro blog: <https://about.sofia2.com/2016/11/16/que-es-nodered/>

y en nuestro canal youtube **Sofia2 IoT Platform**



Arquitectura IoT Sofia2 ¿que vamos a tocar hoy?



Capítulo 2

Demos

Sensorización y presentación en Dashboard

SOURCES >

INGEST & PROCESS >

STORE >

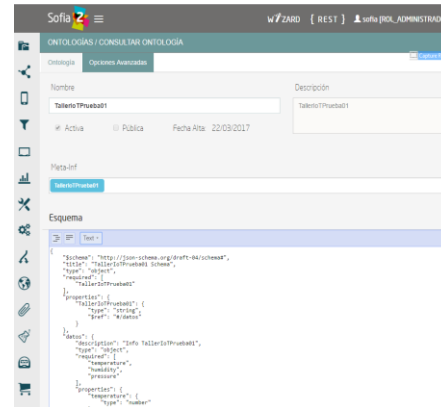
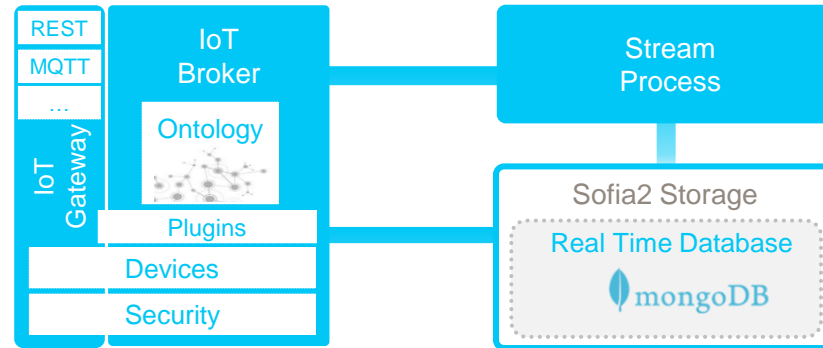
ANALYZE >

PUBLISH & VIEW >

SenseHat
Temp, Hum, Press

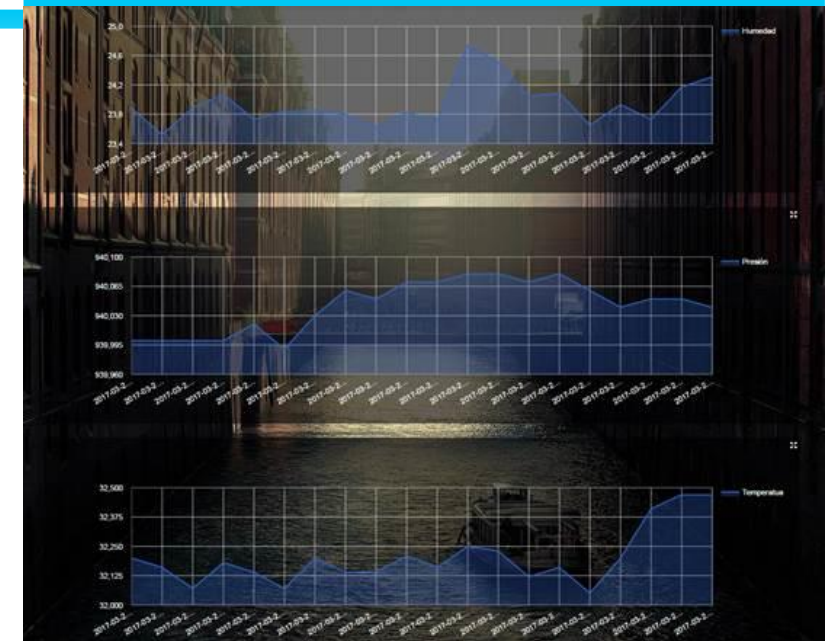


Node-RED

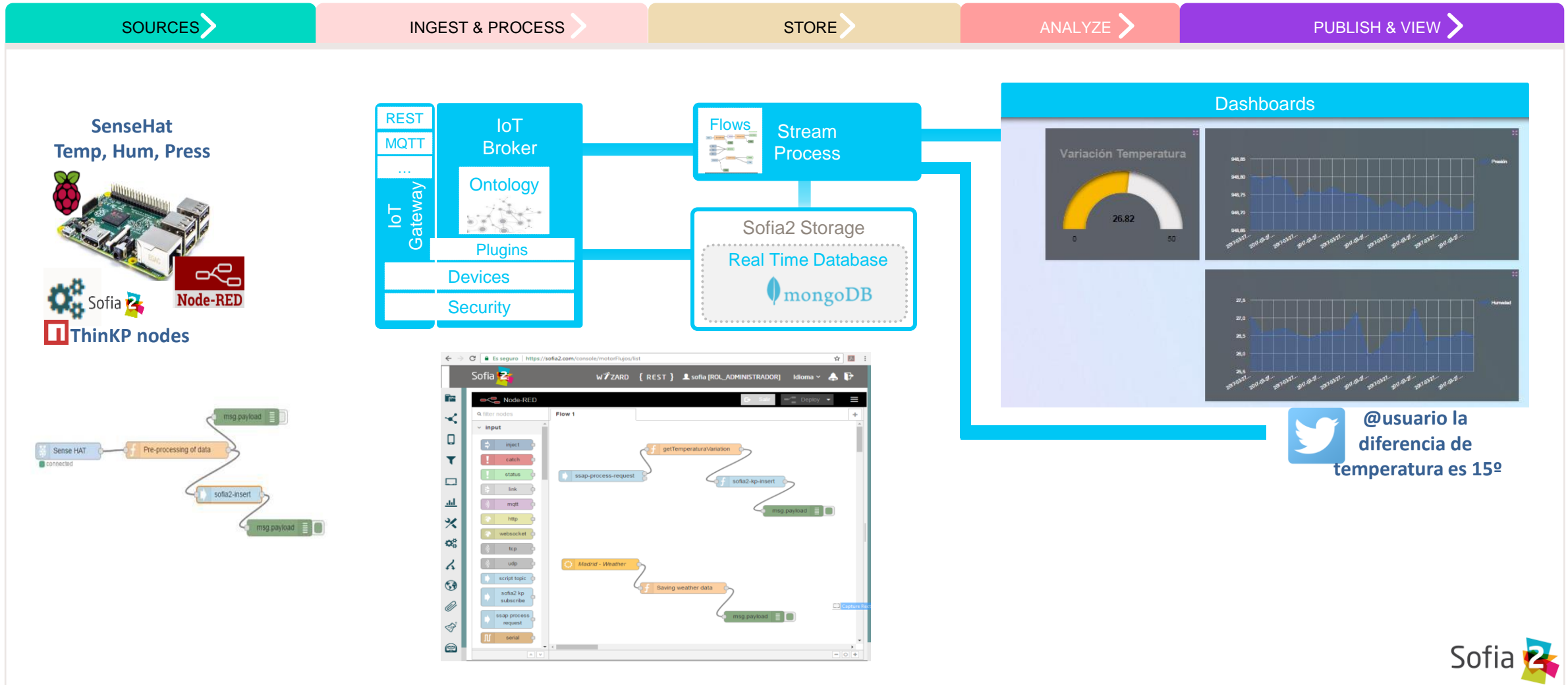


```
{
  "contextData": {
    "session_key": "a885afd5-ca31-43d7-8ba6-023d35736ff3",
    "user": "rbarrio_col",
    "kp": "KPTallerIoTPruebaFlujo1",
    "kp_instancia": "KPTallerIoTPruebaFlujo1",
    "timestamp": "2017-03-23T07:46:37.393Z"
  },
  "TallerIoTPrueba01": {
    "temperature": 32.03,
    "humidity": 23.62,
    "pressure": 939.95
  }
},
{
  "contextData": {
    "session_key": "db35e6e4-31ef-4c79-aeab-aeaf998f7adc",
    "user": "rbarrio_col",
    "kp": "KPTallerIoTPruebaFlujo1",
    "kp_instancia": "KPTallerIoTPruebaFlujo1",
    "timestamp": "2017-03-23T07:46:37.556Z"
  },
  "TallerIoTPrueba01": {
    "temperature": 32.05,
    "humidity": 24.05,
    "pressure": 939.93
  }
}
```

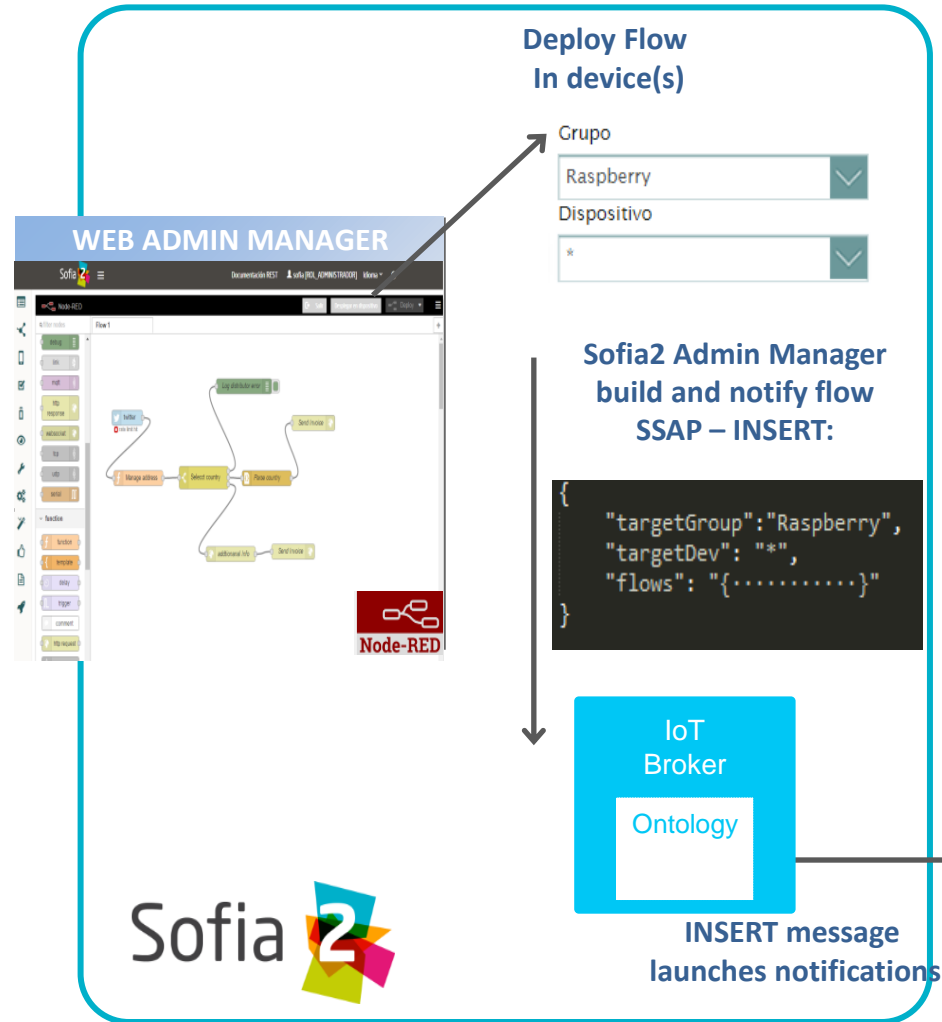
Dashboards



Con lógica en plataforma

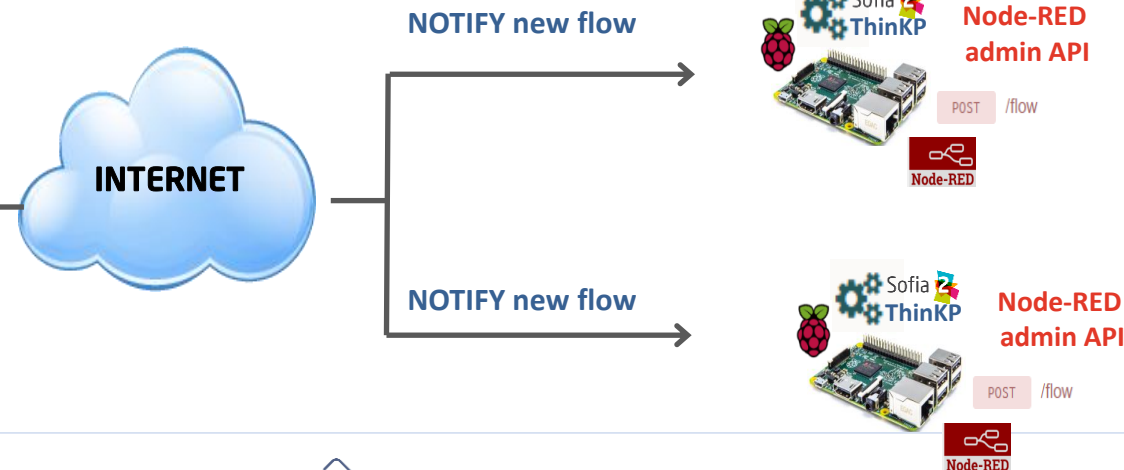


Despliegue centralizado de software a dispositivos



Ejemplo de desarrollo centralizado de flujos en plataforma y despliegue remoto en uno o un grupo de dispositivos:

- Lógica de dispositivos desarrollada como un flujo visual Node-RED en plataforma IoT.
- Pequeño agente de actualización remota ejecutándose en el dispositivo. Se trata de un ThinKP (app Sofia2) suscrito a la plataforma IoT utilizando protocolo de mensajería SSAP.
- Una vez desarrollado y probado el flujo en la plataforma, se comunica al broker IoT para que ejecute las notificaciones a los dispositivos suscritos.
- El agente ThinKP ejecutado en el dispositivo despliega el/los flujos notificados utilizando el API de administración de Node-RED



Capítulo 3

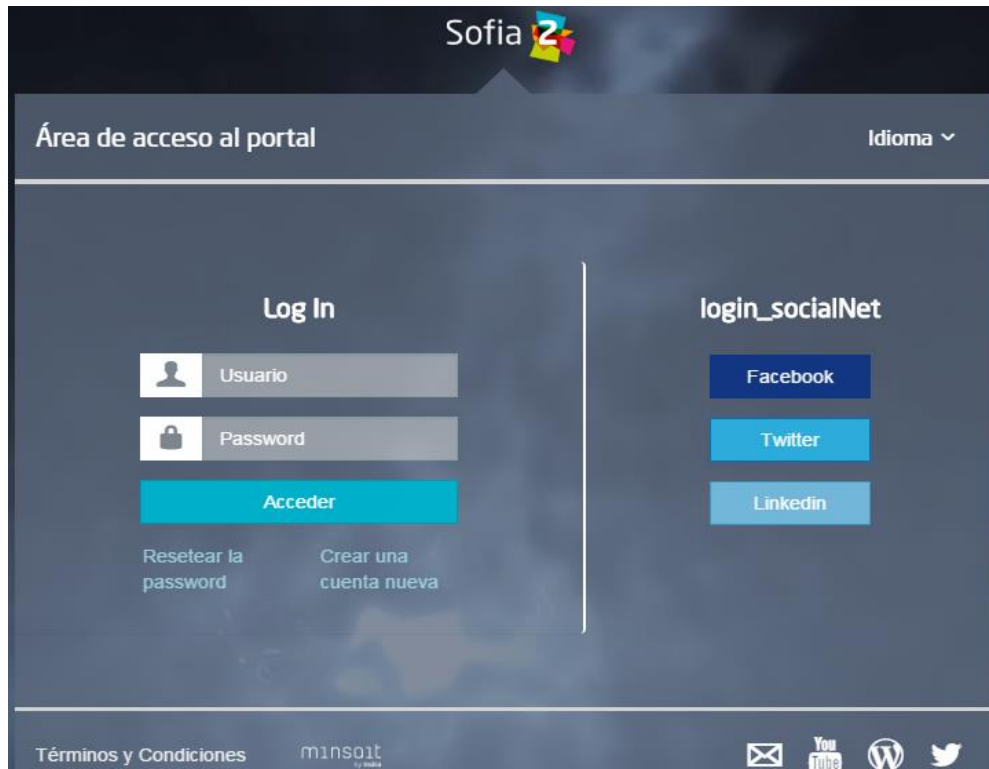
Taller IoT

Pasos Previos (I)

Por cada grupo de trabajo, crear usuario en Sofia2 si no se dispone de uno.

Entrar en <https://sofia2.com/console/login> y seleccionar **Crear una cuenta nueva**

Rellenar los datos de alta y registrarse. En cuestión de minutos tendremos rol colaborador



The screenshot shows the Sofia2 login interface. At the top, there's a header with the Sofia2 logo and a language dropdown labeled 'Idioma'. Below this is a section titled 'Área de acceso al portal'. The main content area is divided into two columns. The left column is titled 'Log In' and contains a 'Usuario' input field, a 'Password' input field, and an 'Acceder' button. Below these fields are links for 'Resetear la password' and 'Crear una cuenta nueva'. The right column is titled 'login_socialNet' and contains three buttons for social media login: 'Facebook', 'Twitter', and 'LinkedIn'.

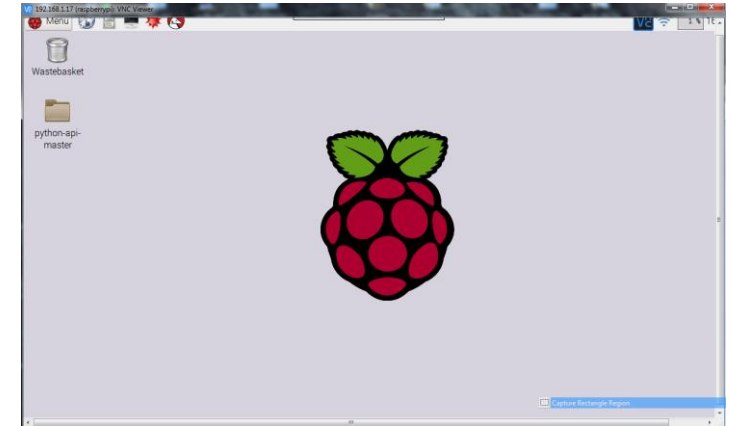


The screenshot shows the Sofia2 user registration interface. At the top, there's a header with the Sofia2 logo and a language dropdown labeled 'Idioma'. Below this is a section titled 'Área de acceso al portal'. The main content area is titled 'Solicitud Alta de Usuario'. It contains four input fields: 'Usuario', 'Password', 'Nombre', and 'Email'. Below these fields is a checkbox with the text 'Al hacer clic en "Registrarse", aceptas las Términos y Condiciones y confirmas que leíste nuestra política de datos, incluido el uso de cookies.' Below the checkbox is a 'Registrarse' button and a link 'Volver al registro'.

Pasos Previos (II)

Preparar nuestra Raspberry para conectar por VNC:

- Si no está instalado (Raspbian por ejemplo ya lo trae de fabrica), instalar un servidor VNC:
 - **\$> sudo apt-get update**
 - **\$> sudo apt-get install realvnc-vnc-server**
- Habilitar Real-VNC para que arranque como servicio, y configurar usuario/password VNC:
 - **\$> sudo raspi-config**
 - **Interfacing Options > VNC**
- Conectar nuestra Raspberry a la red del Campus y anotar su IP:
 - **\$> ifconfig**



Conectar a nuestra Raspberry desde nuestro PC por VNC:

- Descargar el cliente de Real VNC para nuestro SO (No requiere instalación): <https://www.realvnc.com/download/viewer/>
- Ejecutarlo, introducir la IP de nuestra Raspberry y cuando se nos solicite introducir las credenciales asignadas cuando habilitamos el servicio VNC en la Raspberry.
- Una vez que veamos el escritorio de la Raspberry, configurar una resolución con la que nos sintamos cómodos:
 - **\$> sudo raspi-config**
 - **Advanced Options > Resolution** (Por ejemplo elegir 1280x720)
 - **\$> sudo reboot**

Pasos Previos (III)

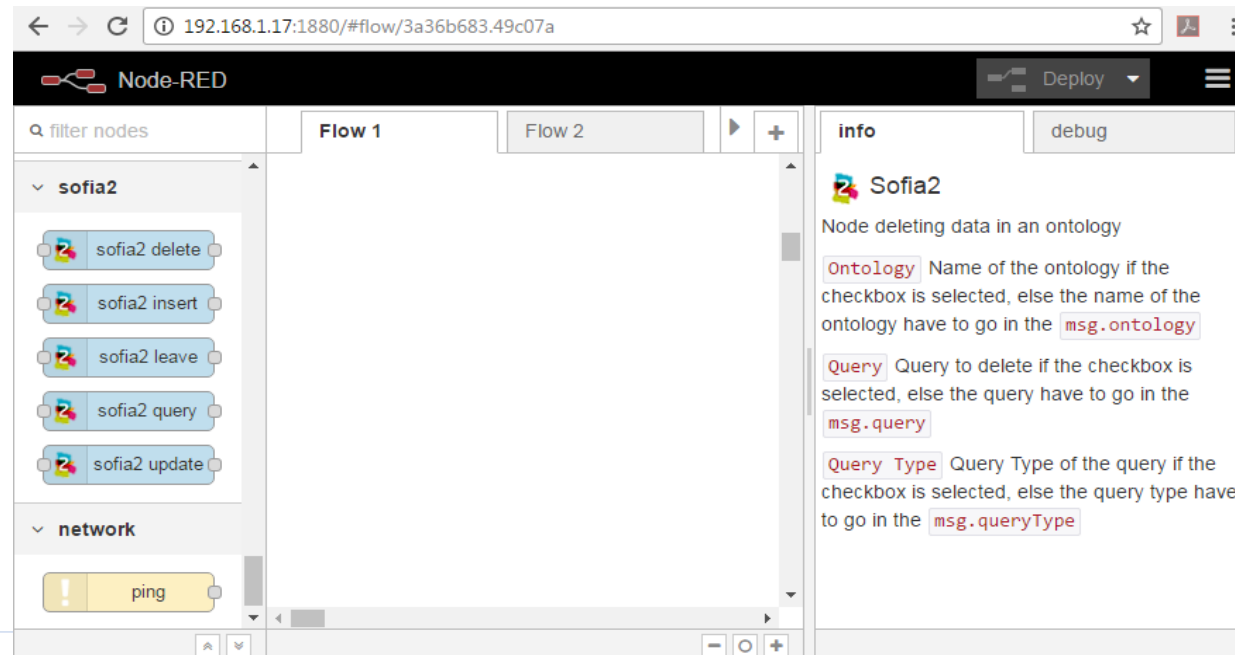
Instalar Node-RED en nuestra Raspberry si no está instalado ya:

- `$> sudo npm install -g --unsafe-perm node-red`

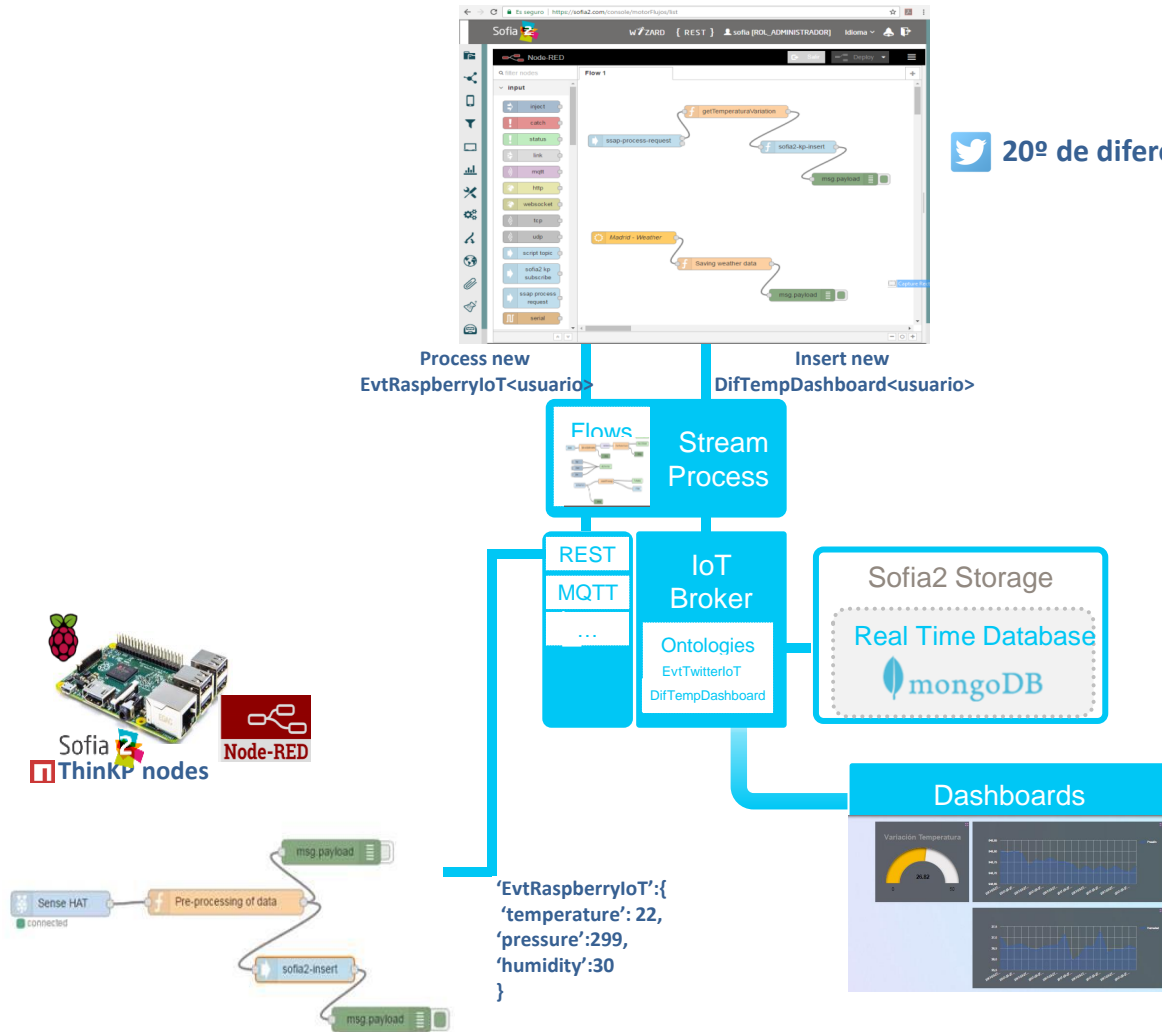
Instalar Nodos Sofia2 ThinkP:

- `$> cd $HOME/.node-red`
- `$> npm install q`
- `$> npm install mqtt`
- `$> npm install wait-until`
- `$> npm install node-red-contrib-thinkp-sofia2`
- `$> sudo service node-red restart`

Comprobar que los nodos están disponibles en la paleta:



Taller. ¿Qué vamos a hacer?



Vamos a desarrollar dos Flujos, uno en la Raspberry Pi y otro en la plataforma Sofia2:

- **El flujo de la Raspberry Pi** será el que desencadene el proceso:
 - Un nodo controla los datos de la placa SenseHat (Si no disponemos de SenseHat la simularemos).
 - Periódicamente se envía un mensaje a plataforma Sofia2 con los datos de temperatura, humedad y presión a la ontología **EvtRaspberryIoT<usuario>**
- **El flujo en Sofia2** aplica una regla muy sencilla para enviar un tweet y mostrar información en un Dashboard:
 - Un nodo escuchará eventos recibidos desde la Raspberry hacia la ontología **EvtRaspberryIoT<usuario>**.
 - Otro nodo recupera temperatura de un servicio de Internet.
 - Dentro del flujo, otro nodo hace una comparación de temperaturas y si la diferencia es superior a un umbral se envía el tweet.
 - La diferencia de temperatura también se almacenará en la ontología **DifTempDashboard<usuario>** para alimentar un Dashboard.

Además de los flujos, crearemos un sencillo **dashboard** alimentado por una ontología mantenida por el flujo de la plataforma.

Taller. Alta de ontologías en Sofia2 (I)

Daremos de alta dos ontologías en Sofia2:

- **EvtRaspberryIoT<usuario>** con los campos:
 - temperature: Tipo number
 - humidity: Tipo number
 - pressure: Tipo number
- **DifTempDashboard<usuario>** con el campo:
 - variacion: Tipo number

Para cada ontología, en la consola de administración de Sofia2:

Crear Ontologia



>

Creación Paso a Paso



Taller. Alta de ontologías en Sofia2 (II)

Editar la ontología:

ONTOLOGÍAS / CREACIÓN GUIADA ONTOLOGÍA

Ontología Opciones Avanzadas

Nombre (*)

EvtTwitterIoTjfgpimpollo

Mínimo 5 caracteres

✓ Activa

Versión Plantilla Actual

0

☐ Pública

Descripción (*)

Ontologia Taller IoT

Mínimo 5 caracteres

Meta-Inf (*)

nodered

Mínimo 2 caracteres

Configuración Esquema

^ SELECCIÓN PLANTILLA

Categoría

General	Smart City	Environment	Smart Industry	Smart Building	Smart Home
Alert	Audit	Empty	EmptyBase	Evtnt	Feed
Plantilla para la definición	Plantilla para registros de		Plantilla con una	Nueva versión	Plantilla de medidas

Nombre: **EvtRaspberryIoT<usuario>**

Descripción: **Ontología Taller IoT**

Meta-Inf: **nodered**

Categoría: **General**

Plantilla: **EmptyBase**



Taller. Alta de ontologías en Sofia2 (III)

^ AÑADIR NUEVAS PROPIEDADES

Nueva Propiedad Ontología

Propiedad

T.datos

requerido

Añadir

Propiedad	T.datos	requerido	
temperature	number	requerido	🗑️
humidity	number	requerido	🗑️
pressure	number	requerido	🗑️

Additional Properties

Utilizar el formulario para añadir las propiedades previamente comentadas

^ GENERADOR ESQUEMA E INSTANCIA JSON

Generar Esquema

Cancelar Crear

Una vez creadas las propiedades de la ontología, pulsar **Generar Esquema** para generar el JSON-Schema que define nuestra ontología

Finalmente, generado el JSON-Schema de la ontología, pulsar **Crear**

Taller. Alta de ThinkKP en Sofia2 (I)

Una vez tenemos dadas de alta las ontologías del modelo de datos, daremos de alta la **configuración lógica de las aplicaciones** (ThinkKp) que las utilizarán. **Daremos de alta un ThinkKp**, que representará al conjunto de aplicaciones que interactuarán con Sofia2 en el taller.

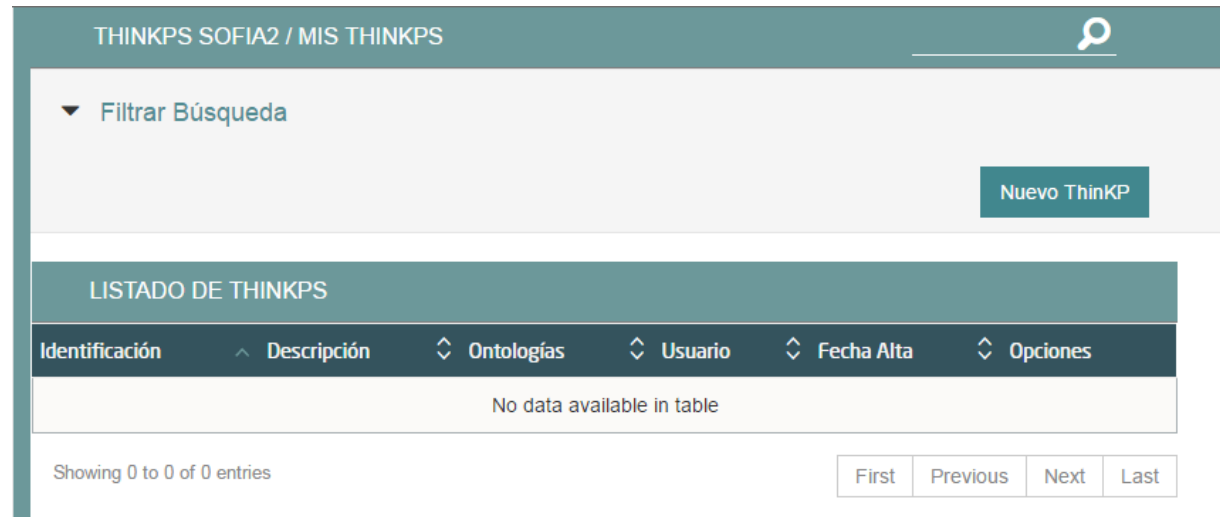
En la consola de administración de Sofia2:

Mis ThinkKPs



>

Nuevo ThinkKP



Taller. Alta de ThinkKP en Sofia2 (II)

Asignar al ThinkKP el nombre **NodeRedRasp<usuario>** y una descripción, y pulsar Crear

THINKPS SOFIA2 / NUEVO THINKP

ThinKP Mis Tokens Mis Instancias

Identificación

NodeRedRaspjfgpimpollo

Descripción

ThinkKP taller IoT

Ontología Ontologías de Grupo

EvtTwitterIoTjfgpimpollo-Ontologia Taller

Una vez creado, seleccionar la pestaña **Mis Tokens** y **anotar el token de autenticación**, ya que lo necesitaremos mas adelante, para autenticar a las aplicaciones

THINKPS SOFIA2 / CONSULTAR THINKP

ThinKP Mis Tokens Mis Instancias

ThinKP	Propietario	Token	Ult. Conexión	Activo
NodeRedRaspjfgpimpollo	jfgpimpollo	09dc8456137c4a568e9ca15a6571fb9a		✓

Showing 1 to 1 of 1 entries

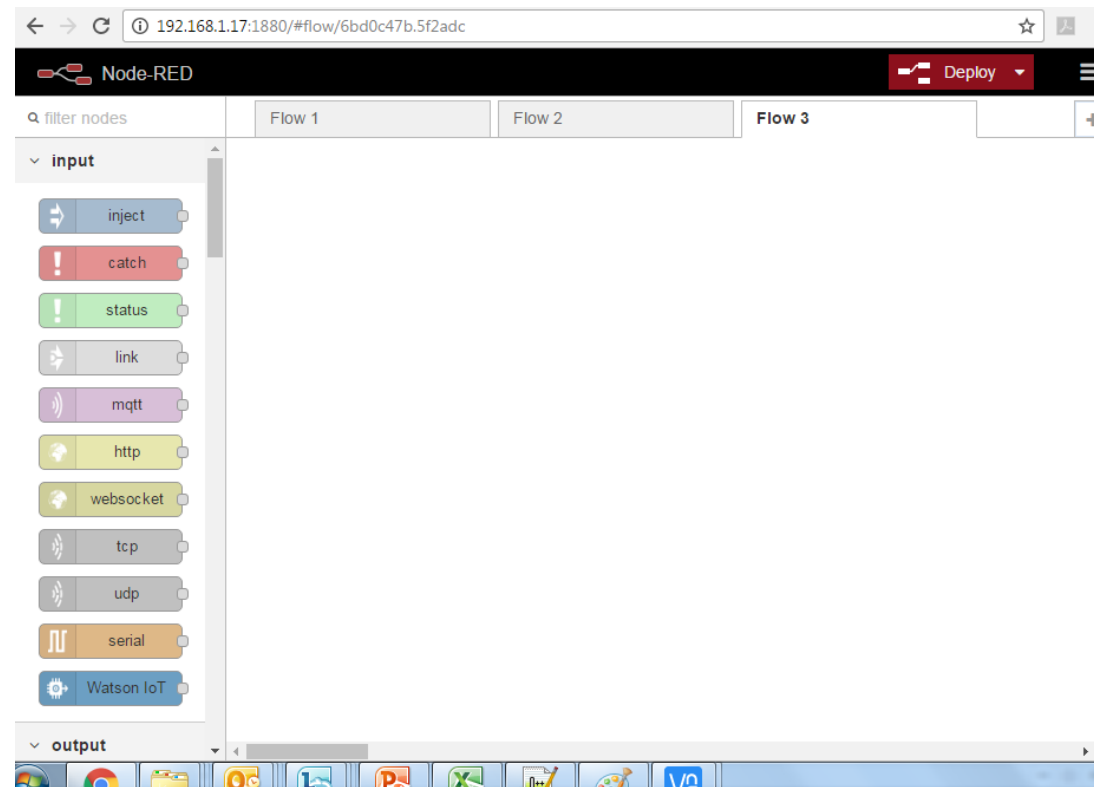
First Previous 1 Next Last

Taller. Flujo en Raspberry Pi (I)

Desarrollo del flujo en Raspberry Pi:

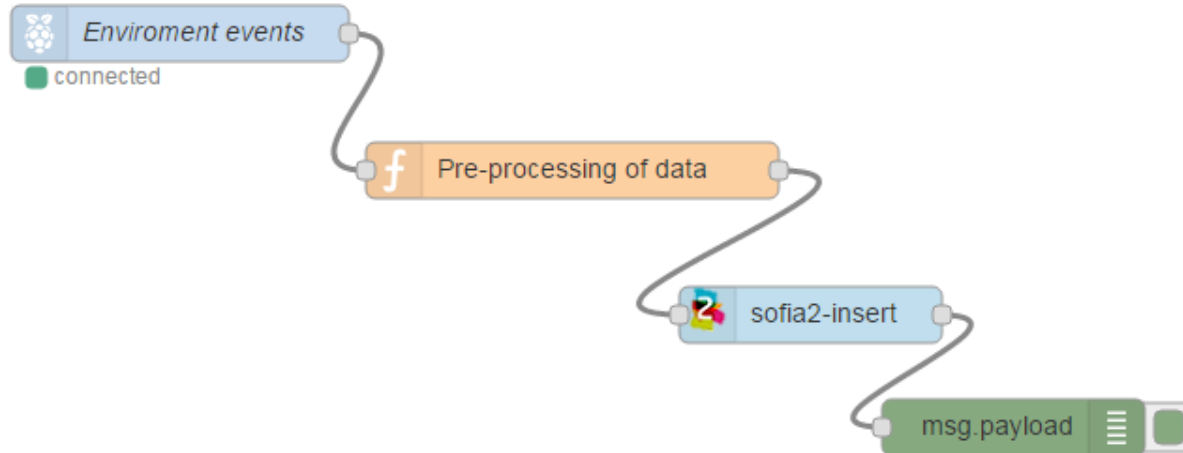
Con nuestra Raspberry configurada y NodeRED arrancado en ella accedemos al editor visual desde nuestro PC con un navegador: <http://<ipraspberrypi>:1880/>

Si no tenemos ningún flujo creado previamente se mostrará un editor vacío



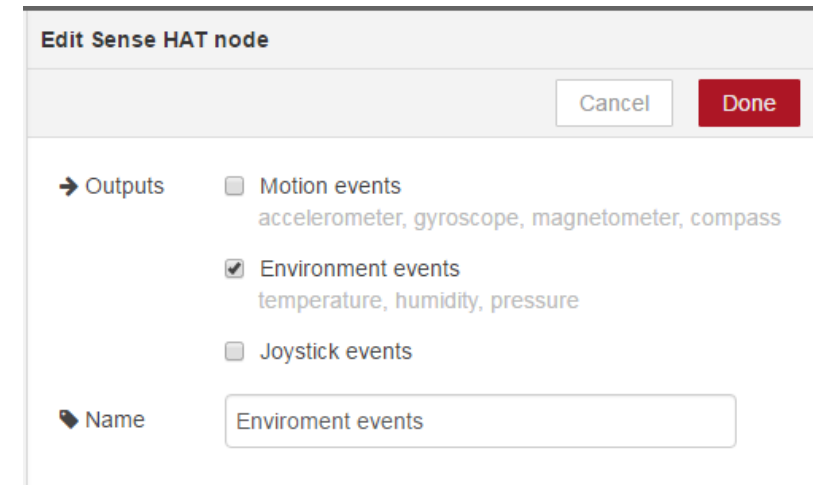
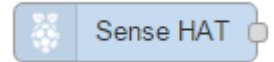
Taller. Flujo en Raspberry Pi (II)

El flujo que desarrollaremos en la Raspberry Pi tendrá este aspecto:



Nodo Environment events:

- Tipo en paleta: **Raspberry_Pi > SenseHat**
- Configurado para capturar eventos ambientales:

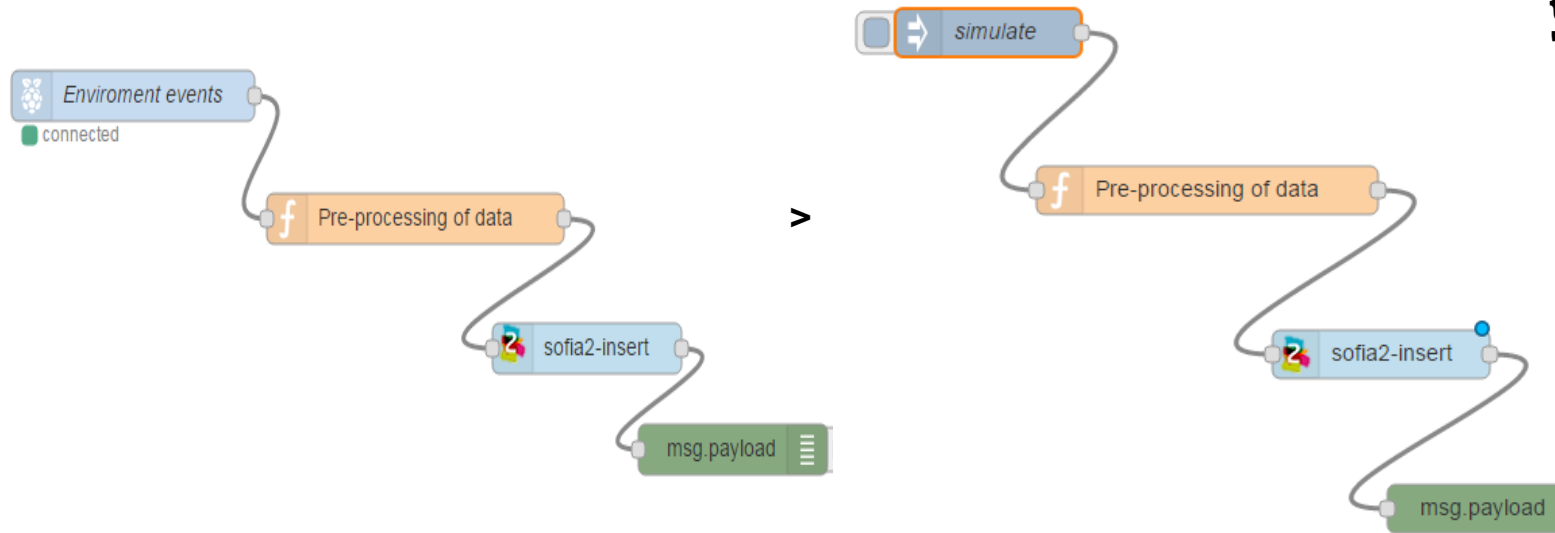


- Cada segundo inyecta al flujo un nuevo mensaje donde en el atributo payload incluye las propiedades
 - temperature
 - pressure
 - humidity

Taller. Flujo en Raspberry Pi (III)

Importante: ¿Y si no tenemos SenseHat?, si tenemos SenseHat pasamos a la siguiente página

Sustituimos el Nodo **Enviroment events** por un nodo que lo **simule**
(Existe un simulador de SenseHat, por si queréis experimentar mas allá del taller)



Nodo simulate:

- Tipo en paleta: **input > inject**
- Configurado para enviar un objeto JSON: `{"temperature":10, "humidity":20, "pressure":30}`:

Edit inject node Cancel Done

Payload `{ "temperature":10, "humidity":20, "pressure":30 }`

Topic

Repeat interval

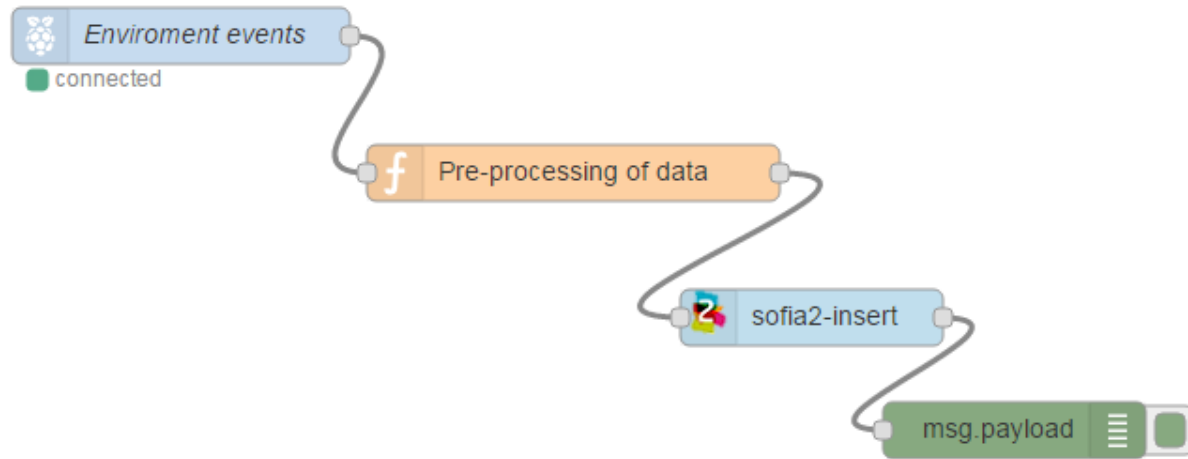
every seconds

☐ Inject once at start?

Name

Note: "interval between times" and "at a specific time" will use cron. See info box for details.

Taller. Flujo en Raspberry Pi (IV)



Nodo Pre-processing of data:

- Tipo en paleta: **function > function**
- Configurado con el código javascript para construir una instancia de la ontología **EvtRaspberryIoT<usuario>** (Sustituid cada uno vuestro usuario como elemento raíz del mensaje)



Edit function node

Cancel

Done

Name

Pre-processing of data

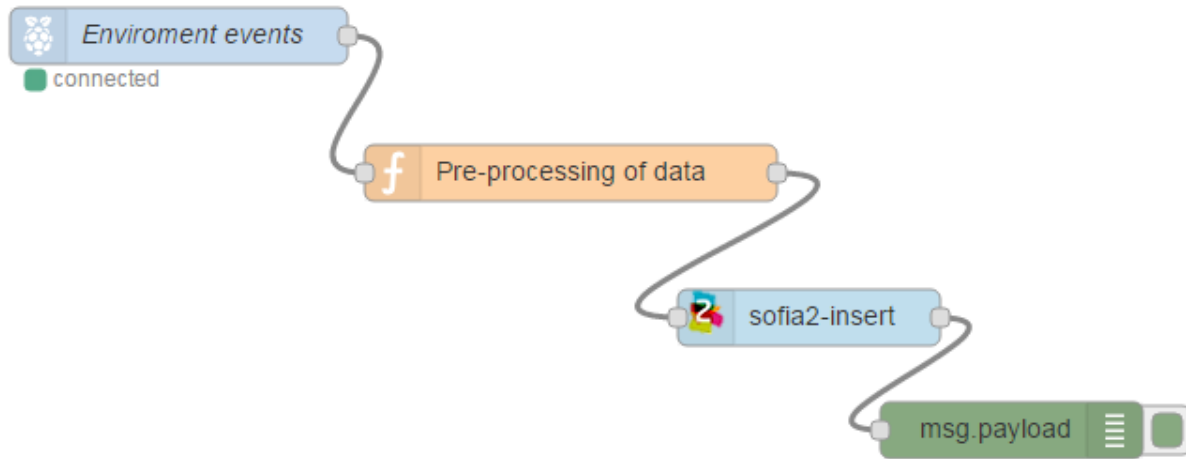
Function

```
1 //Nos creamos el objeto JSON que cumpla
2 //con la instancia de la ontología
3 var message = {EvtRaspberryIoT<usuario>:{}};
4 message.TallerIoTPrueba01=msg.payload;
5 msg.payload=message;
6 msg.payload=JSON.stringify(msg.payload);
7 return msg;
```

Outputs

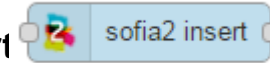
1

Taller. Flujo en Raspberry Pi (V)



Nodo sofia2-insert:

- Tipo en paleta: **sofia2> sofia2-insert**
- Configurado con:
 - La ontología en la que insertar dato del atributo **payload** del mensaje recibido del nodo anterior
EvtRaspberryIoT<usuario> (Sustituid cada uno vuestra ontología).
 - Server: La instancia de Sofia2 donde Insertar, y que tenemos que configurar previamente pulsando



Edit sofia2-insert node

Cancel
Done

Name

☒ Insert the data manually

Ontology

Server

sofia2-insert > Edit sofia2-config node

Delete
Cancel
Update

Protocol

Endpoint

ThinkKP

Instance

Token

Renovate session every minutes

Taller. Crear Flujo en Sofia2 (I)

Acceder a la consola de administración de Sofia2 <http://sofia2.com/console>

Seleccionar Mis Flujos



>

Comprobar que tenemos una instancia del motor de flujos en nuestro dominio de aplicación

MIS DOMINIOS

Crear Dominio

Identificación	Estado	Uso CPU	Memoria [MB]	Opciones	Modo demo	Tiempo restante
pruebatallermeetup01		--	--		<input type="checkbox"/>	--

Showing 1 to 1 of 1 entries

FirstPrevious1NextLast

Arrancarla si está parada, seleccionando la opción Arrancar

Identificación	Estado	Uso CPU	Memoria [MB]	Opciones	Modo demo	Tiempo restante
pruebatallermeetup01		0	93		<input checked="" type="checkbox"/>	11:43

Taller. Crear Flujo en Sofia2 (II)

Una vez arrancada la instancia podemos entrar en el editor de NodeRED pulsando Ver Flujos

Identificación

Identificación	Estado	Uso CPU	Memoria [MB]	Opciones	Modo demo	Tiempo restante
pruebatallerteetup01		0	93			06:11

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

MIS FLUJOS

Ver Flujos

Sofia 2

WZARD { REST } rbarrio_col [ROL_COLABORADOR] Idioma

filter nodes

Flow 1

input

- inject
- catch
- status
- link
- mqtt
- http
- websocket
- tcp
- udp
- script topic
- sofia2 kp subscribe
- ssan process

ssap-process-request

publish Tweet?

getTemperaturaVariation

sofia2-kp-insert

switch

Tweet

Madrid - Weather

Saving weather data

msg.payload

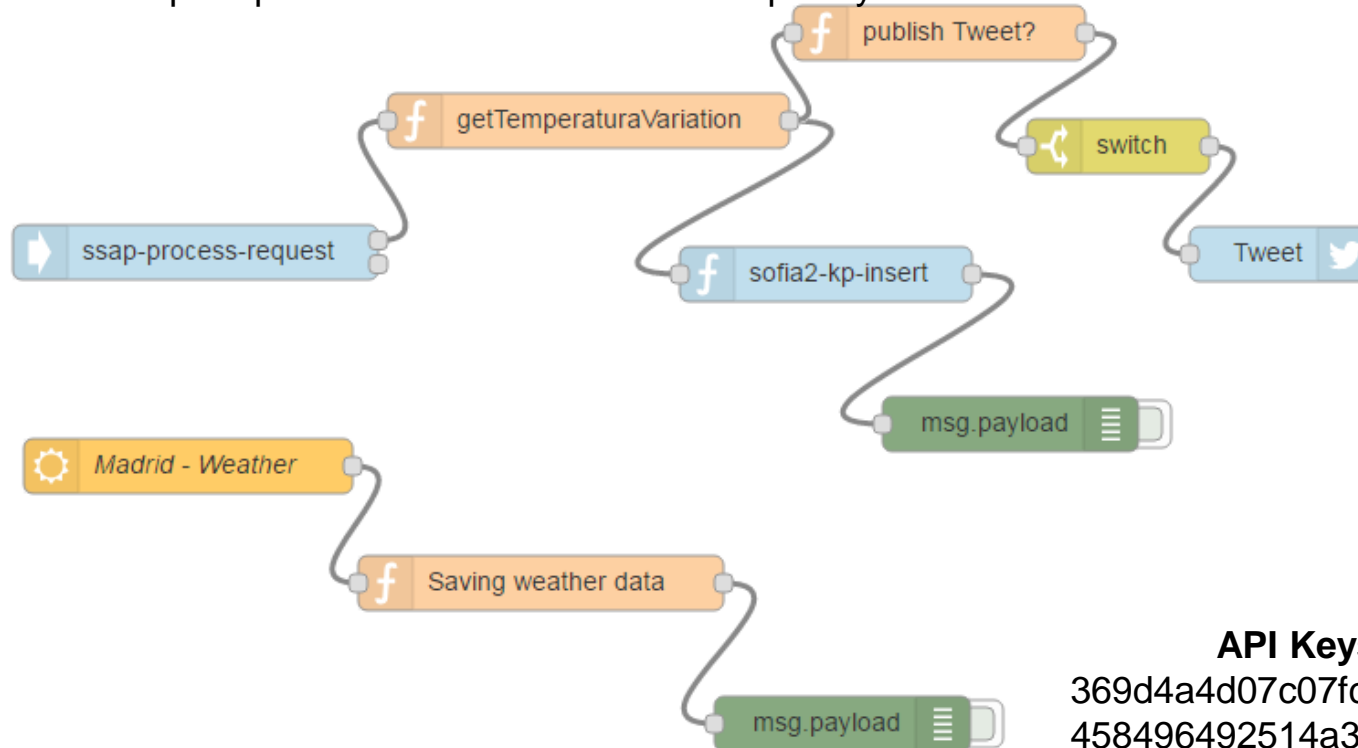
msg.payload

Taller. Flujo en Sofia2 (I)

El flujo que desarrollaremos en Sofia2 tendrá este aspecto:

Se trata de dos flujos paralelos:

- Uno para que el flujo disponga de datos de temperatura en Madrid.
- Otro para procesar eventos desde la Raspberry



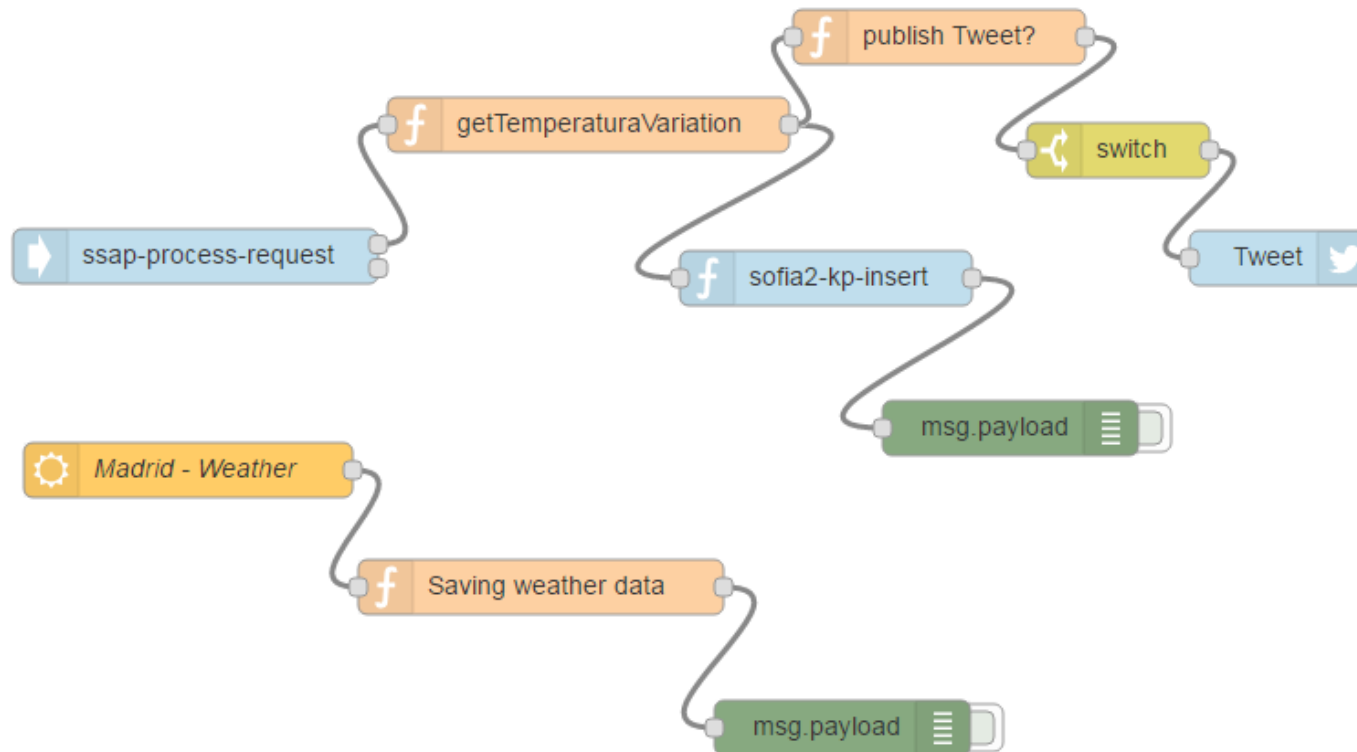
Nodo Madrid - Weather:

- Tipo en paleta: **weather > openweathermap**
- Configurado para obtener datos meteorológicos de Madrid:

API Keys disponibles:

369d4a4d07c07fcf75deb9117c20744d
 458496492514a31cdd9e57e6b5568236
 e79303d0c4b17ea81e8a68e47cf43adf
 d4f7ee91843679808906d56554a45268
 242c46302d7312874a9012b4794dee6f
 c5657966c20aa92912e4ee8df9c04803

Taller. Flujo en Sofia2 (II)



Nodo Saving weather data:

- Tipo en paleta: **function > function**
- Configurado para almacenar la temperatura recuperada de Open Weather en el contexto del flujo, para que cualquier otro nodo la pueda consultar en el futuro.

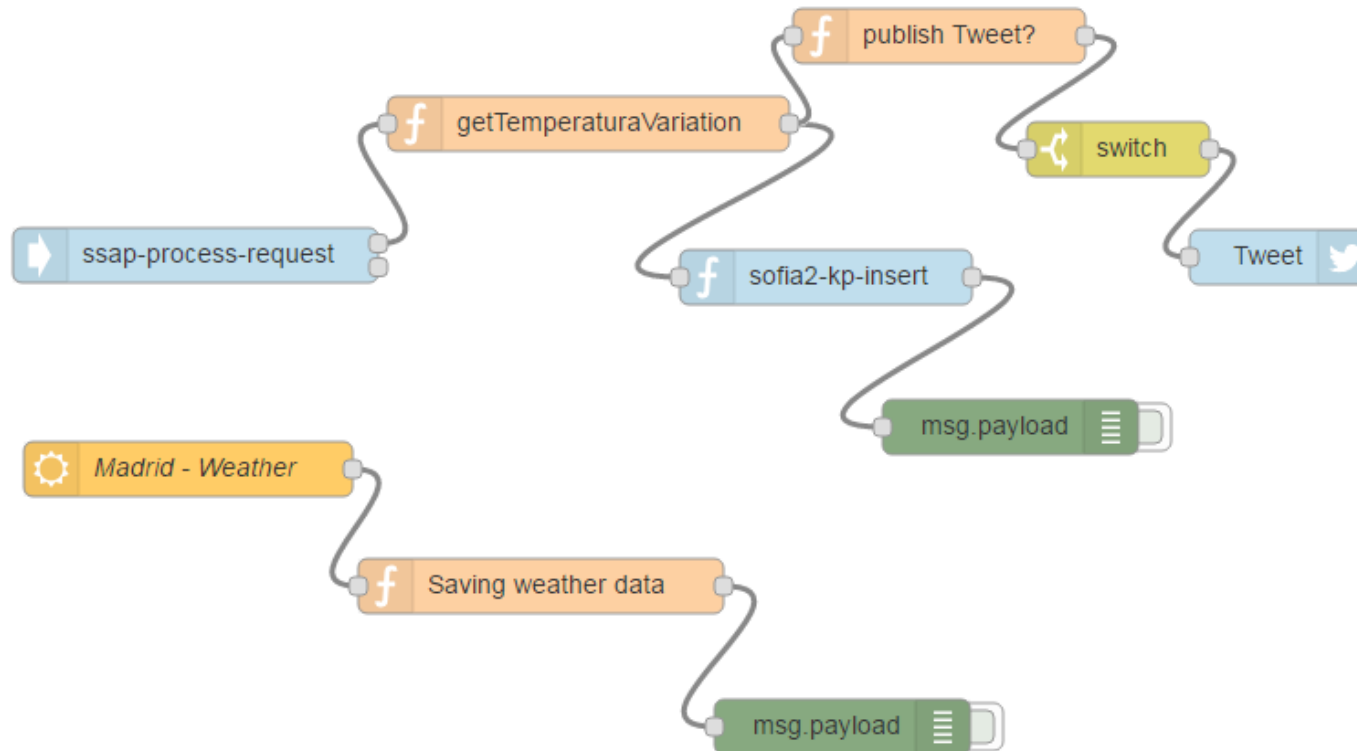
Name

Function


```

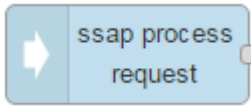
1 var temp = msg.payload.tempc;
2 flow.set("temp", temp);
3 msg.payload=msg.payload.tempc;
4 return msg;
    
```

Taller. Flujo en Sofia2 (III)



Nodo ssap-process-request:

- Tipo en paleta: **input > ssap process request**
- Configurado para iniciar el flujo con los eventos entrantes en la plataforma desde la Raspberry Pi, cada vez que inserta nuevos datos en la ontología:



Edit ssap-process-request node

Cancel Done

Name

Message address

Type message

Ontologies available

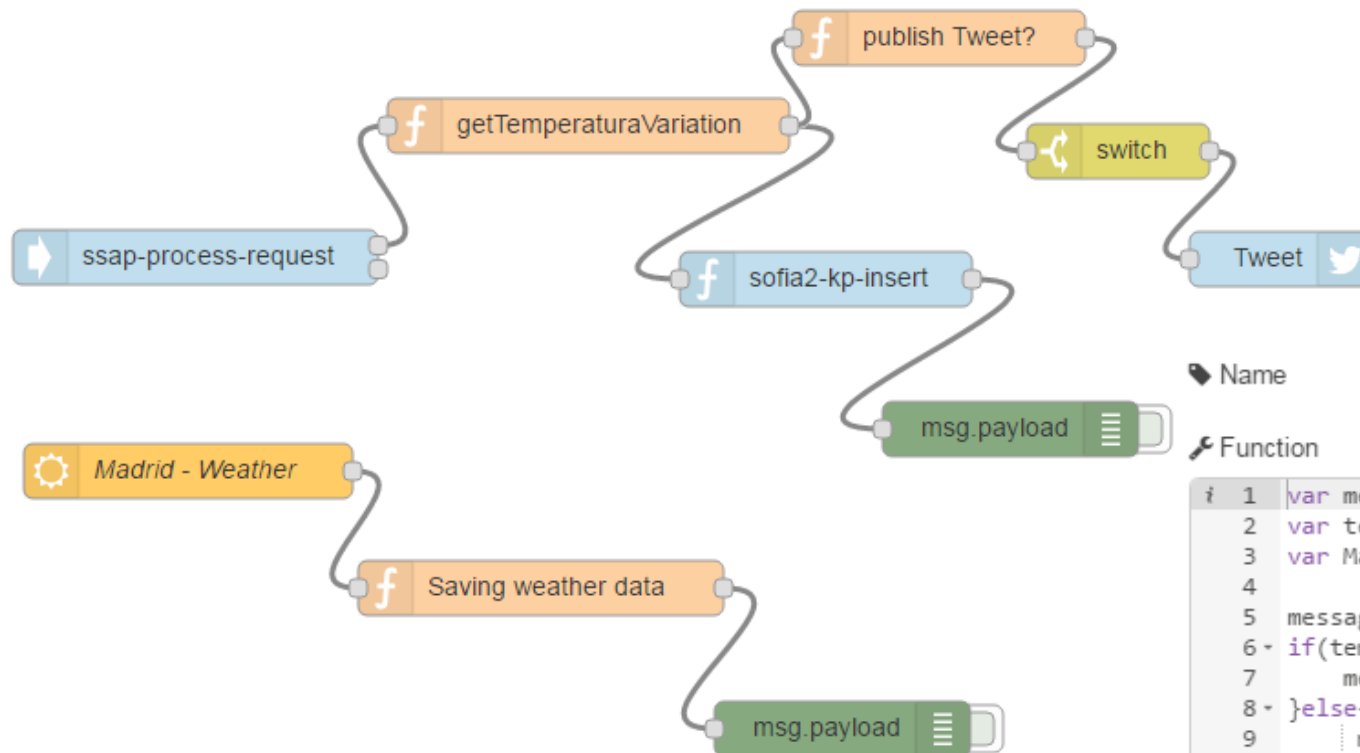
Ontology

ThinkKP available

ThinkKP

KP instance

Taller. Flujo en Sofia2 (IV)



Nodo getTemperaturaVariation:

- Tipo en paleta: **function > function**
- Configurado para comparar la temperatura recibida desde la raspberry y la almacenada en el contexto del flujo (Open Weather). Y componer una nueva instancia para insertar en la ontologia DifTempDashboard<usuario> que alimentará nuestro Dashboard

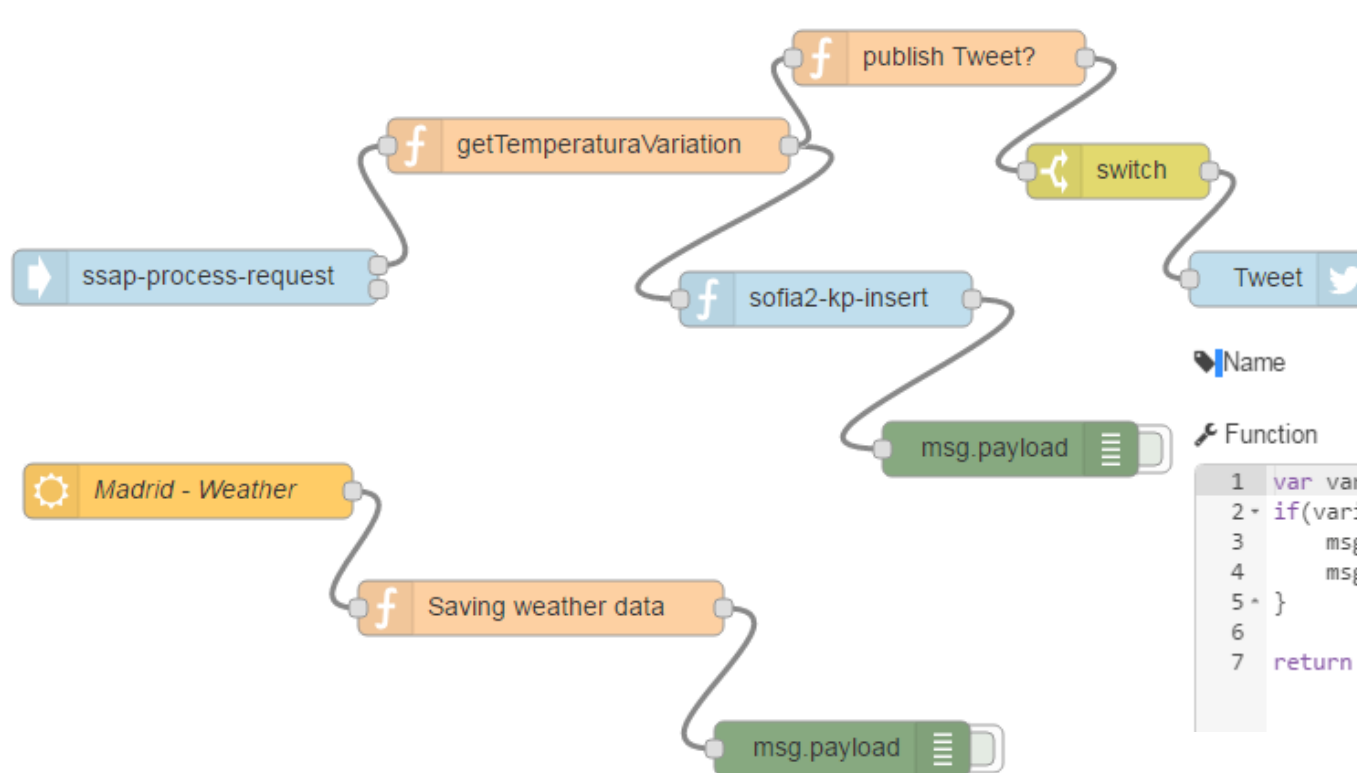
Name: getTemperaturaVariation

Function:

```

1 var message = JSON.parse(msg.payload)
2 var temperature = JSON.parse(message.data).TallerIoTPrueba01.temperature;
3 var MadridTemperatura = flow.get("temp") || 22;
4
5 message = {"VariacionTemperatura":{}};
6 if(temperature>MadridTemperatura){
7   message.VariacionTemperatura.variacion=temperature-MadridTemperatura;
8 }else{
9   message.VariacionTemperatura.variacion=MadridTemperatura-temperature;
10 }
11
12 msg.payload=JSON.stringify(message);
13 msg.ontology="VariacionTemperatura";
14
15 return msg;
  
```

Taller. Flujo en Sofia2 (V)



Nodo publish Tweet?:

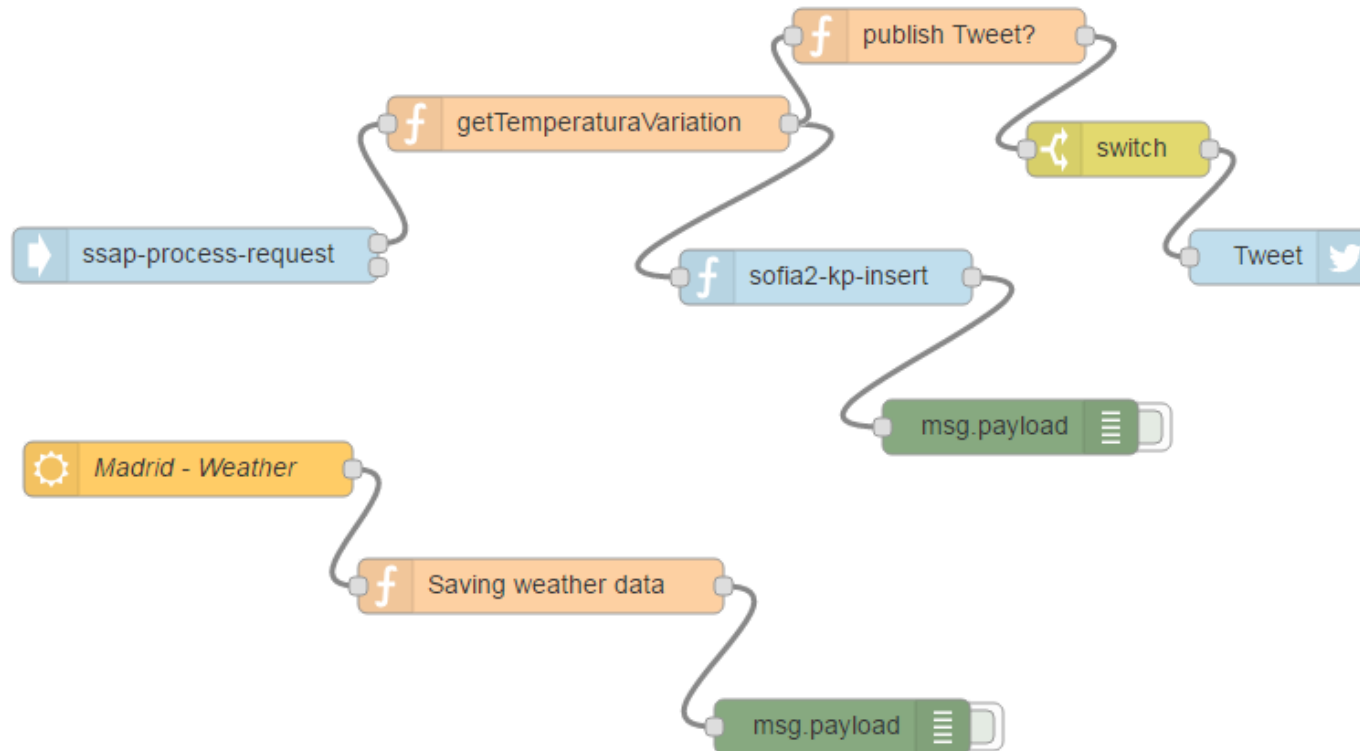
- Tipo en paleta: **function > function**
- Configurado como una regla para decidir si publicar un Tweet o no en función de si la diferencia supera un umbral

Name publish Tweet?

Function

```
1 var variacion = JSON.parse(msg.payload).VariacionTemperatura.variacion;
2 if(variacion>10){
3     msg.tweet=true;
4     msg.payload="La variación de temperatura es superior a 10 grados el día " + new Date();
5 }
6
7 return msg;
```

Taller. Flujo en Sofia2 (VI)



Nodo switch:

- Tipo en paleta: **function > switch**
- Configurado para propagar el mensaje si la propiedad tweet es true.

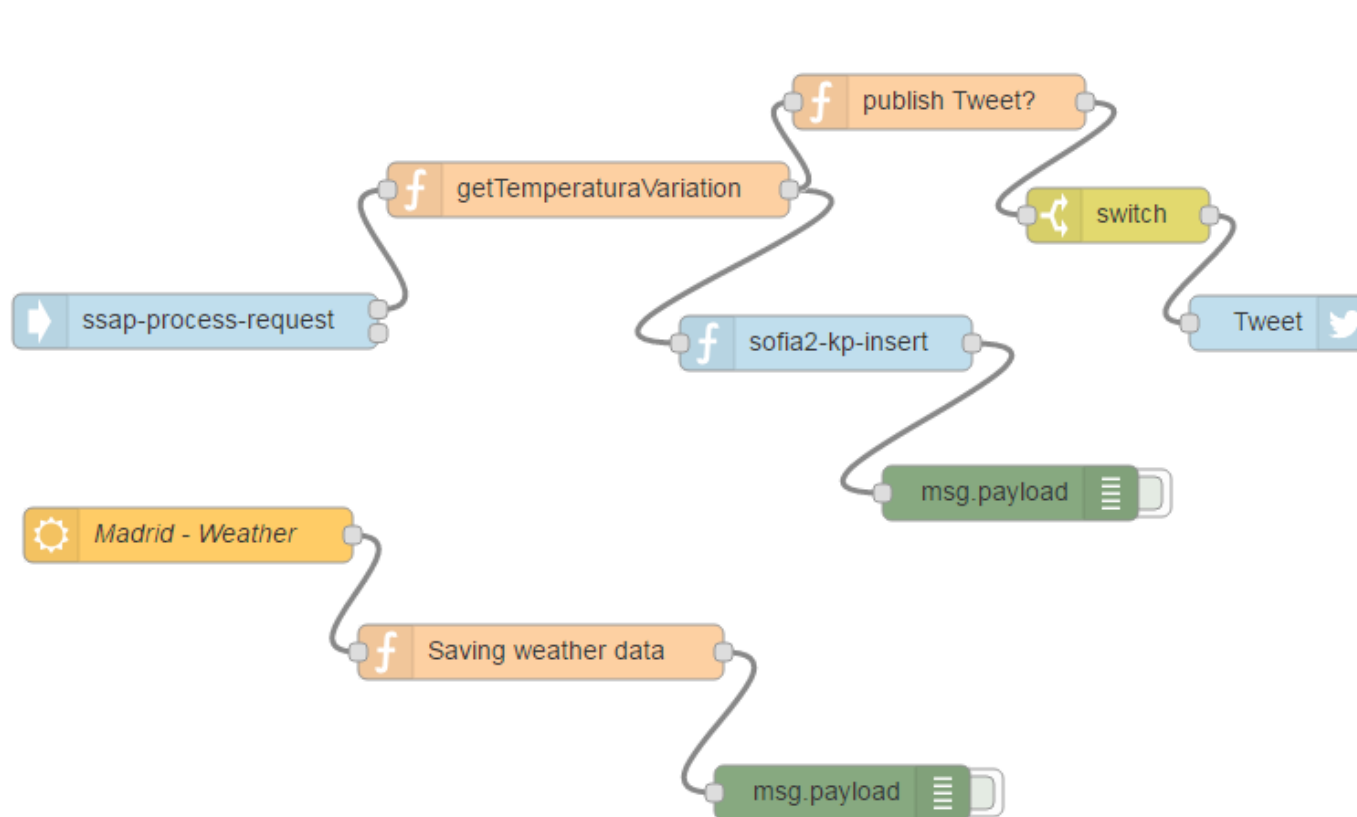


Nodo Tweet:

- Tipo en paleta: **social > Twitter**
- Configurado para enviar el Tweet con la cuenta configurada

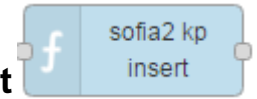


Taller. Flujo en Sofia2 (VIII)



Nodo sofia2-kp-insert:

- Tipo en paleta: **function > sofia2-kp-insert**
- Configurado para insertar la instancia con la variación de temperatura propagada en el nodo anterior. Para ello utiliza los atributos:
 - **payload:** JSON con la instancia a insertar
 - **ontology:** Ontología en la que insertar



Name	<input type="text" value="Name"/>
Token	<input type="text" value="120b5766f8d34a16b4c7687bb78bbe6d"/>
ThinkKP available	<input type="text" value="KPVariacionTemperatura"/>
ThinkKP	<input type="text" value="KPVariacionTemperatura"/>
KP instance	<input type="text" value="01"/>

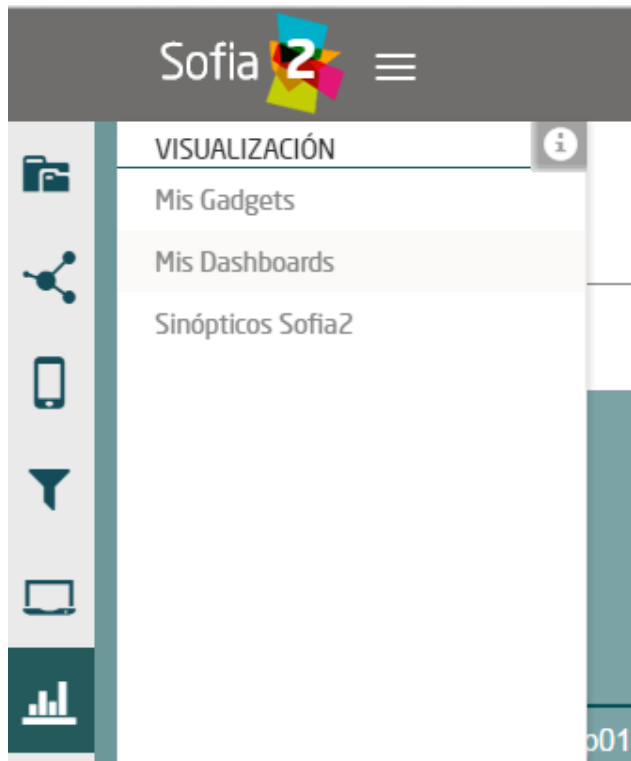
Los datos insertados por este nodo, son los que alimentan el dashboard que veremos a continuación

Taller. Creación Dashboard(I)

A continuación crearemos el Dashboard en Sofia2, para representar los datos guardados en las ontologías

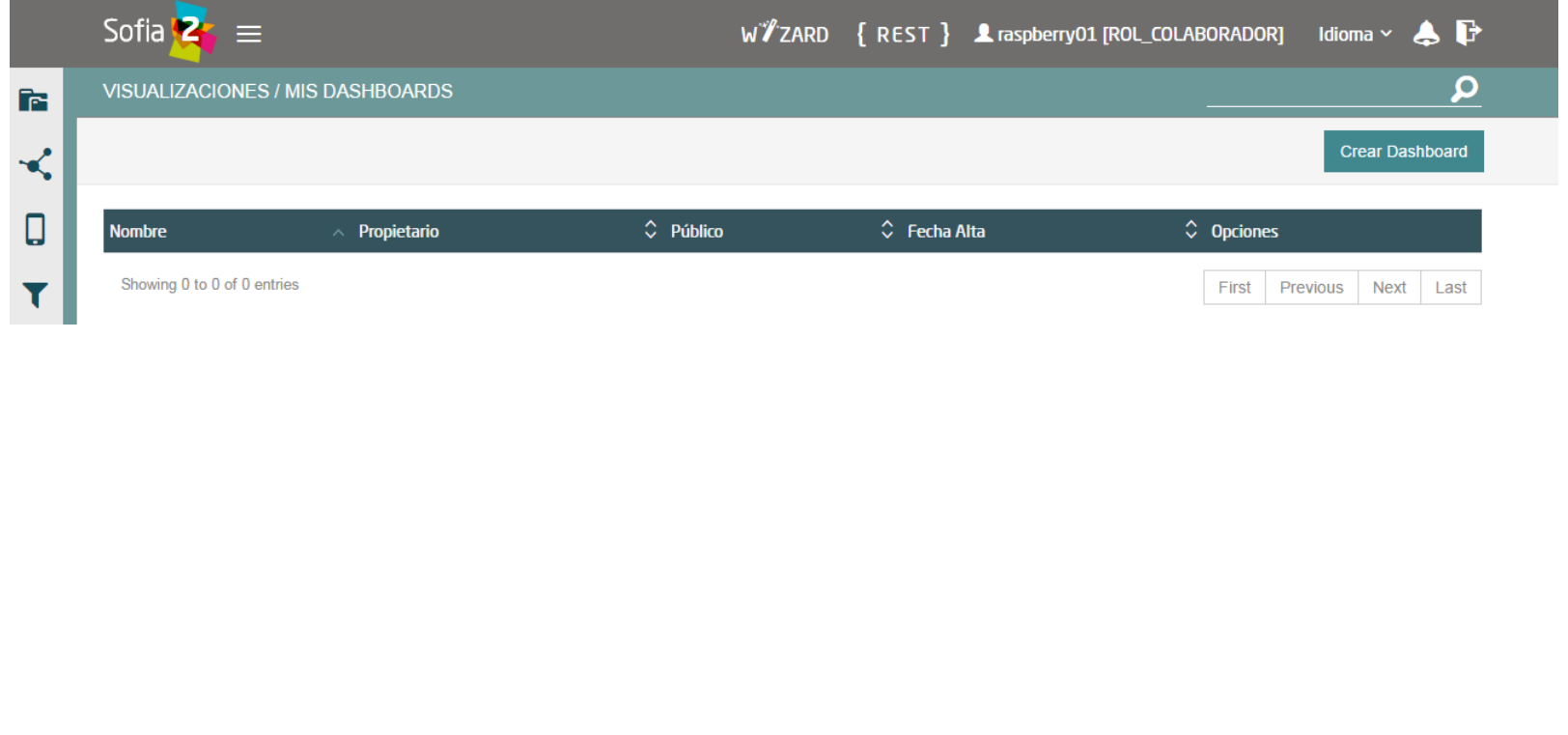
En la consola de administración de Sofia2:

Mis Dashboards



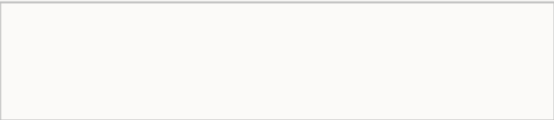
>

Crear Dashboard



Taller. Creación Dashboard(II)


Asignar al Dashboard el nombre **DashboardSenseHat<usuario>**, seleccionar el **estilo** del dashboard y pulsar **Nueva Página**



Nombre

Dashboard_SenseHat_Enviroments ☐ Público

Imagen logo

 [Seleccionar archivo](#)

Estilo

[Default Theme](#) [Blue Green Theme](#) [Dark Theme](#) [Minsait Theme](#) [Dark Blue Theme](#) [Custom Style](#)

Menu

[Vertical](#) [Horizontal](#)

[Nueva Pagina](#)

Una vez creada la página pulsamos en **Crear Gadget**, ya que **un Dashboard se compone a al menos un Gadget**.
En nuestro caso crearemos un Gadget para cada medida que queramos representar.



Taller. Creación Dashboard(III)

- Seleccionaremos el tipo de Gadget **Básico** > **Área**
- Le asignaremos un **nombre** y seleccionaremos el **ThinkKP** que representa el **Gadget** en Sofia2 a efectos de consultar datos. En nuestro caso **NodeRed<usuario>**

Sofia 2

WIZARD { REST } raspberry01 [ROL_COLABORADOR] Idioma

VISUALIZACIONES / CREAR GADGET

Nombre

Temperatura

KP

NodeRedRasp01

Obtener datos en directo

Obtener datos por query

Máximos valores a representar

20

Medidas

Ontología	Eje X	Transformación dato X	Eje Y	Transformación dato Y	Nombre Serie
EvtRaspberryIoTRasp01	ContextData.Timestamp		temperature		Temperatura

Taller. Creación Dashboard(III)

Para un gadget tipo **Área**: añadiremos las **medidas que se quieren representar** en este gadget, para ello hay que seleccionar la **ontología** de la que extraer datos y se deberán seleccionar los campos de dicha ontología que serán representados tanto en el **eje X** como en el **eje Y**.

Medidas

Ontología	Eje X	Transformación dato X	Eje Y	Transformación dato Y	Nombre Serie
EvtRaspberryIoTRasp01	ContextData.Timestamp		temperature		

Token

Para un gadget tipo **Gauge**: añadimos el dato a representar y si es necesario el **rango de valores** entre los que va a oscilar.

Una vez añadidas las medidas seleccionamos el **Token** activo que se corresponda con el **ThinkP** y pulsamos en **Crear Gadget**.

Repetir estos pasos con cada dato que se quiera representar en el Dashboard.

Medidas

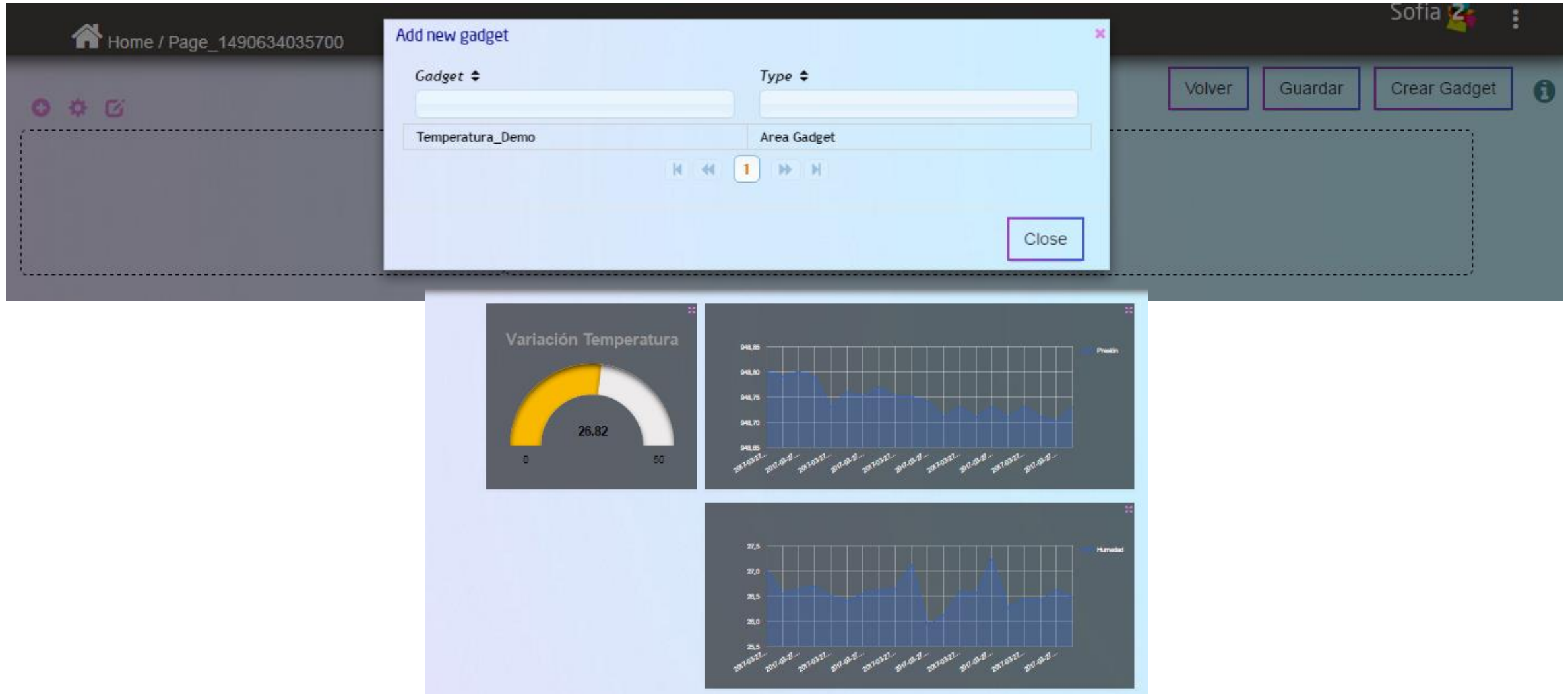
Ontología	Valor	Transformación Dato	Mínimo Rango	Máximo Rango	Nombre
DifTempDashboardRasp07	variacion		0	100	

Token

Taller. Creación Dashboard(IV)

Componer el dashboard:

Una vez creados los gadgets, sólo hay que añadirlos al Dashboard.





http://twitter.com/SOFIA2_Platform



<http://about.sofia2.com>



plataformasofia2@indra.es



<http://sofia2.com>

Muchas gracias!!!