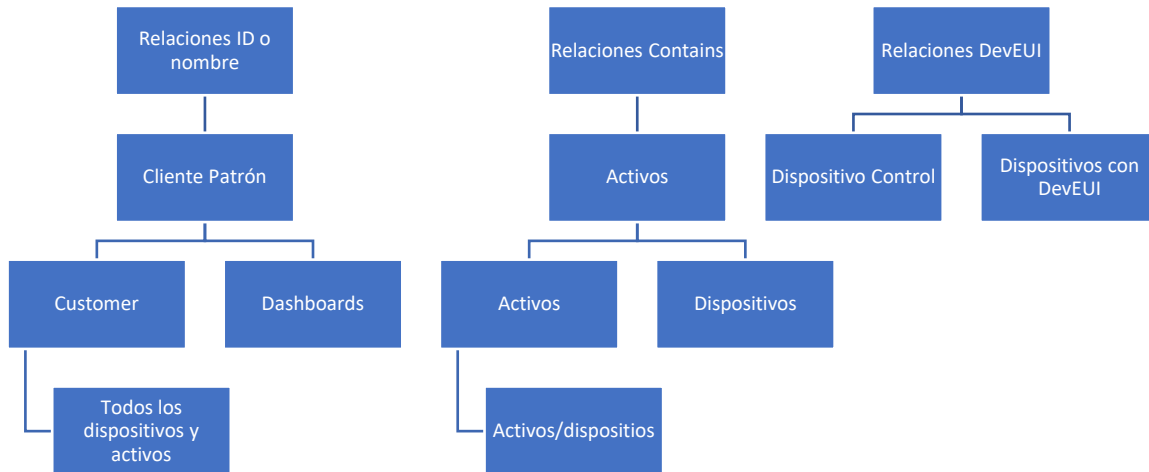


# Crear un nuevo tipo de dispositivo para myIoT

## 1. myIoT a vista de pájaro

- 1.1. **Usuario:** Los usuarios de myIoT tienen el rol de customers. Para poder desarrollar nuevos tipos de dispositivos necesitamos una cuenta con rol de tenant en la plataforma de desarrollo tb.iotopentech.io (solicítela a [juanfelixmateos@gmail.com](mailto:juanfelixmateos@gmail.com) si no dispone de ella).
- 1.2. **Activo Root:** Cada usuario de myIoT tiene un activo con este nombre, desde el que se establecen relaciones "From" de tipo "Contains" a todos los demás activos y dispositivos que cree el usuario dependiendo directamente de él. Básicamente sirve para mostrar la jerarquía de todos los activos/dispositivos en los dashboards.
- 1.3. **dispositivo Control:** Cada usuario de myIoT tiene un dispositivo con este nombre. Se utiliza fundamentalmente para recibir a través de telemetrías las operaciones que desea realizar el usuario pero no le están permitidas por tener rol de Customer, como crear nuevos activos/dispositivos o cambiar atributos de los dispositivos o del propio usuario (como su dirección de correo electrónico). Además, en el caso de dispositivos LoRaWAN podemos usar el dispositivo Control como destinatario de todos los mensajes (sin tener que configurar una integración específica para cada dispositivo), que él se encargará de redireccionar al dispositivo correspondiente extrayendo el devEUI del mensaje. Esto se consigue mediante relaciones "From" de tipo "[el devEUI del dispositivo]" desde el dispositivo Control a cada uno de los dispositivos LoRaWAN.
- 1.4. **Paneles:** Además de los 2 paneles siguientes, cada vez que un usuario cree un dispositivo/activo, se le asigna automáticamente el panel de tipo de dispositivo/activo correspondiente.
  - 1.4.1. **Configuración:** Sirve para configurar los datos de notificación al usuario (email, Telegram...), crear/configurar/delegar activos/dispositivos, reclamar dispositivos, y realizar un seguimiento del crédito.
  - 1.4.2. **Panel de control:** Sirve para acceder a los paneles de cada tipo de activo/dispositivo.
- 1.5. **cliente\_patron:** Este cliente se utiliza fundamentalmente para gestionar las versiones de los tipos de dispositivos (también para mantener una copia de seguridad de la contabilidad). Posee relaciones "From" de tipo "[el id del customer]" con todos los demás usuarios, y también relaciones "From" de tipo "[el nombre del tipo de dispositivo]" con todos los dashboards de tipos de dispositivos.
- 1.6. Crear un activo
- 1.7. Crear un dispositivo
- 1.8. Dashboard de tipo de dispositivo
- 1.9. Configurar un dispositivo

## 1.10. Delegar un dispositivo



## 2. Establecer el entorno de desarrollo

2.1. Descargar el repositorio: <https://github.com/loTopenTech/myloTopenTech/archive/master.zip>

2.2. Crear un Customer y, dentro de él, un usuario

2.2.1. Nombre customer: 00000001

2.2.1.1. Nombre usuario: [usuario@\[sunombre\].com](mailto:usuario@[sunombre].com) (Por ejemplo: [usuario@juanfelixmateos.com](mailto:usuario@juanfelixmateos.com))

Anote aquí el nombre elegido: \_\_\_\_\_

Anote aquí la contraseña elegida: \_\_\_\_\_

2.3. Importar paneles:

2.3.1. Panel de control

2.3.2. Dispositivo\_LDS01

2.3.2.1. Configuración: Configurar el alias del siguiente modo:

Edit alias

Alias name \*

customer

Resolve as multiple entities

Filter type \*

Single entity

Type

Current Customer

Default customer\*

00000001

SAVE

CANCEL

2.3.3.Activo\_IMAGE01

2.3.4.Activo\_MAP01

2.4. Crear el activo Root: 00000001\_ROOT

2.4.1.Asset Type: Root

2.4.2.Atributos de servidor

2.4.2.1. nombreEntidad: [String] \_ROOT

2.4.2.2. tipoEntidad: [String] ROOT

2.5. Crear el dispositivo Control: 00000001\_CONTROL

2.5.1.Device Type: System

2.6. Asignar el activo ROOT y el dispositivo Control al customer 00000001

2.7. Configurar en el customer 00000001 una relación From de tipo [id del activo ROOT] al activo 00000001\_ROOT

00000001

Customer details

DETAILS

ATTRIBUTES

LATEST TELEMETRY

ALARMS

EVENTS

RELATIONS

AUDIT LOGS

Direction

From

Outbound relations

+

<input type="checkbox"/> Type ↑	To entity type	To entity name	
<input type="checkbox"/> 704bc060-07c0-11eb-b82a-9b3a9efbb111	Asset	00000001_ROOT	<div></div> <div></div>

Page: 1

Rows per page: 5

1 · 1 of 1

2.8. Asignar los paneles Configuración y Panel de control al customer 00000001

2.9. Atributos del usuario

2.9.1.tiposDeActivos: [String] IMAGE01,MAP01

2.9.2.tiposDeDispositivos: [String] LDS01

2.9.3.credito: [Integer] 365

2.9.4.dispositivosPropios: [Integer] 0

2.9.5.dispositivosAsumidos: [Integer] 0

2.10. Importar reglas

2.10.1. notificaciones

2.10.2. actualizarConfigDashboardSingular

2.10.3. actualizarAccounting

2.10.4. enviarDownlink

2.10.5. crearDelegado

2.10.6. borrarEntidad (actualizarAccounting)

2.10.7. crearEntidad (actualizarConfigDashboardSingular)

2.10.8. editarEntidad (actualizarConfigDashboardSingular)

2.10.9. reclamarDispositivo (actualizarConfigDashboardSingular y actualizarAccounting)

2.10.10. configurarDelegacion (borrarEntidad)

2.10.11. Crear manualmente una regla llamada "Root Rule Chain Next"

2.10.12. Crear manualmente una regla llamada "entregaPatron"

2.10.13. Crear manualmente una regla llamada "configurarEntidad"

2.10.14. Crear manualmente una regla llamada "switchTipoDispositivo"

2.10.15. Root Rule Chain (entregaPatron, Root Rule Chain Next, editarEntidad, crearEntidad, borrarEntidad, configurarEntidad, reclamarDispositivo, crearDelegado, configurarDelegacion y switchTipoDispositivo)

2.10.15.1. Establecer esta regla como regla Raíz (y borrar la anterior Root Rule Chain si existiera)

2.10.16. LDS01 (enviarDownlink, Root Rule Chain y notificaciones)

2.10.17. Importar switchTipoDispositivo (LDS01), copiar todo su contenido, borrarla, y pegar el contenido copiado en la regla " switchTipoDispositivo " creada anteriormente.

2.10.18. Importar configurarEntidad (Root Rule Chain), copiar todo su contenido, borrarla, y pegar el contenido copiado en la regla " configurarEntidad " creada anteriormente.

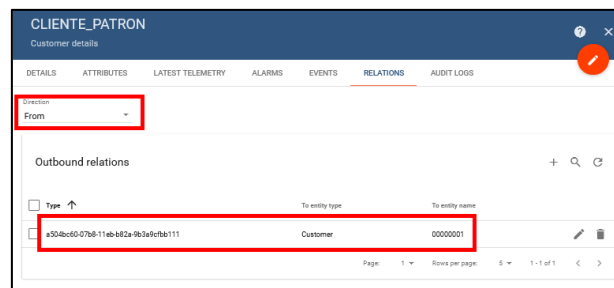
2.10.19. Importar entregaPatron (Root Rule Chain), copiar todo su contenido, borrarla y pegar el contenido en la regla "entregaPatron" creada anteriormente.

2.10.20. Importar Root Rule Chain Next (Root Rule Chain), copiar todo su contenido, borrarla y pegar el contenido en la regla " Root Rule Chain Next " creada anteriormente.

2.11. El cliente\_patron

2.11.1. Crear el cliente\_patron

2.11.2. Crearle una relación From al customer 00000001 de tipo el id del customer 00000001.



2.11.3. Crearle una relación From al dashboard Activo\_MAP01 de tipo "MAP01".

2.11.4. Crearle una relación From al dashboard Activo\_IMAGE01 de tipo "IMAGE01".

2.11.5. Crearle una relación From al dashboard Dispositivo\_LDS01 de tipo "LDS01".

2.11.6. Crearle los atributos del servidor siguientes:

2.11.6.1. IMAGE01\_config

```
<myIoT>
  <panel titulo="Configuración general"
    resumen="Configurar atributos de la entidad"
    nombreFormulario="General"
    labelBotonSubmit="Configurar">
    <item tipo="atributoInterno"
      nombreAtributo="urlImagenFondo"
      labelAtributo="URL de la imagen de fondo"
      tipoAtributo="texto" />
    </panel>
</myIoT>
```

### 2.11.6.2. LDS01\_config

```
<myIoT>
  <panel titulo="Configuración general" resumen="Configurar atributos de la entidad"
nombreFormulario="General">
    <item tipo="coordenadas" /> <item tipo="chirpstack" />
    <item tipo="alarma" nombreAlarma="cambioDeEstado" telemetria="DOOR_OPEN_STATUS"
labelAlarma="cambio de estado" tipoAlarma="opciones" labelAuxAlarma="Disparar al"
opciones="abrir/cerrar" />
    <item tipo="alarma" nombreAlarma="nivelDeBateria" telemetria="BAT_V" labelAlarma="nivel bajo de
batería" tipoAlarma="umbralMinimo" labelAuxAlarma="(V)" histeresis="min">
        <umbralMinimo size="10" min="0" max="5" step="0.1" /> </item>
        <histeresisMinimo min="0" max="5" step="0.1"/>
    <item tipo="alarma" nombreAlarma="duracionApertura" telemetria="LAST_DOOR_OPEN_DURATION"
labelAlarma="duracion (minutos)" tipoAlarma="biUmbral" labelAuxAlarma="de duración (minutos)"
histeresis="no">
        <umbralMinimo min="0" /> <umbralMaximo min="1" />
    </item>
    <item tipo="alarma" nombreAlarma="numeroDeAperturas" telemetria="DOOR_OPEN_TIMES"
labelAlarma="número de aperturas" tipoAlarma="umbralMaximo" labelAuxAlarma="de número de aperturas"
histeresis="no">
        <umbralMaximo min="1" /> </item>
    <item tipo="alarma" nombreAlarma="inactividad" />
  </panel>
  <panel tipo="devEUI" />
  <panel titulo="Heartbeat" resumen="Configurar periodo de envío de heartbeat"
nombreFormulario="Heartbeat" ultimoDownlink="heartbeat" requiereDelegacion="heartbeat">
    <item tipo="atributoCompartido"
        nombreAtributo="heartbeat"
        labelAtributo="Número de segundos entre heartbeats. Tenga en cuenta que debe resetear el
dispositivo para habilitar este valor."
        tipoAtributo="numero"
    >
        <atributosHTML size="10" step="1" min="0" max="16777215" />
    </item>
  </panel>
  <panel titulo="Borrar contador" resumen="Poner a cero el contador de aperturas"
nombreFormulario="Contador" labelBotonSubmit="Borrar contador" ultimoDownlink="borrarContador">
    <item tipo="atributoCompartido" nombreAtributo="borrarContador" tipoAtributo="boton" />
  </panel>
  <panel titulo="Resetear" resumen="Resetear dispositivo" nombreFormulario="Resetear"
labelBotonSubmit="Resetear" ultimoDownlink="resetear">
    <item tipo="atributoCompartido" nombreAtributo="resetear" tipoAtributo="boton" />
  </panel>
</myIoT>
```

### 2.11.6.3. LDS01\_delegate

```
<myIoT>
  <delegacion nombre="BAT_V" label="Permitir ver la tensión de la batería" />
  <delegacion nombre="DOOR_OPEN_STATUS" label="Permitir ver el estado de la puerta" />
  <delegacion nombre="DOOR_OPEN_TIMES" label="Permitir ver el contador de aperturas de la puerta" />
  <delegacion nombre="LAST_DOOR_OPEN_DURATION" label="Permitir ver la duración de la última apertura
de la puerta" />
  <delegacion nombre="heartbeat" label="Permitir cambiar el periodo de envío de heartbeats (tenga en
cuenta que para que el cambio sea efectivo se necesita resetear el dispositivo)" />
  <delegacion nombre="borrarContador" label="Permitir borrar el contador de aperturas de la puerta"
/>
  <delegacion nombre="resetear" label="Permitir resetear el dispositivo" />
</myIoT>
```

### 2.11.7. Asignarle al cliente\_patron los paneles Activo\_MAP01, Activo\_IMAGE01 y Dispositivo\_LDS01.

### 3. Crear un tipo de dispositivo nuevo (LHT65)

#### 3.1. Procedimiento general

3.1.1. Comunicar a la autoridad ([juanfelixmateos@gmail.com](mailto:juanfelixmateos@gmail.com)) del tipo de dispositivo nuevo que se va a crear (para que se reserve su nombre), y si va a ser un dispositivo público o privado.

3.1.2. Definir los nombres de las telemetrías

3.1.3. Definir las alarmas

3.1.4. Definir los atributos internos

3.1.5. Definir los atributos compartidos con los dispositivos subordinados (delegados)

3.1.6. Construir el atributo delegación (\_delegate)

3.1.7. Construir el atributo configuración (\_config)

3.1.8. Crear la cadena de reglas del tipo de dispositivo

3.1.9. Crear el dashboard del tipo de dispositivo

3.1.9.1. Si se requieren widgets nuevos, consultar con la autoridad ([juanfelixmateos@gmail.com](mailto:juanfelixmateos@gmail.com))

3.1.10. Entregar para su validación:

3.1.10.1. Atributo delegación

3.1.10.2. Atributo configuración

3.1.10.3. Cadena de reglas del tipo de dispositivo

3.1.10.4. Dashboard del tipo de dispositivo (incluida librería de widgets si fuera necesario)

3.1.10.5. Si se trata de un tipo de dispositivo público, idealmente debería entregarse también un archivo MD con la documentación para colocarlo en el repositorio. Pueden utilizarse como referencia los que ya hay en el repositorio (<https://github.com/loTopenTech/myloTopenTech/tree/master/TiposDeDispositivos>).

3.2. Añadir LHT65 al atributo tiposDeDispositivos del customer 00000001, que quedará del siguiente modo: LDS01,LHT65

3.3. Telemetrías: Son los nombres que asignamos a las series temporales que el dispositivo envía a la plataforma. Si el fabricante ofrece un payload decoder, puede resultar cómodo utilizar los mismos nombres para los campos, pero no es obligatorio porque myIoT realiza su propia decodificación a partir del payload raw. Los nombres de las telemetrías no pueden utilizar el carácter /.

Telemetría	Descripción
Ext_sensor	Tipo de sensor externo
BatV	Tensión de la batería
TempC_SHT	Temperatura del sensor interno
Hum_SHT	Humedad del sensor interno
TempC_DS	Temperatura del sensor conectado a la toma externa
Exti_pin_level	Estado (high o low) del sensor conectado a la toma externa
Exti_status	Indica si la telemetría ha sido desencadenada por una interrupción del sensor externo
ILL_lux	Nivel de iluminación del sensor conectado a la toma externa
ADC_V	Tensión del sensor analógico conectado a la toma externa
Exti_count	Pulsos contabilizados por el sensor conectado a la toma externa
No_connect	En los sensores externos (excepto el DS18B20) indica si el sensor está conectado o no

### 3.4. Atributos

#### 3.4.1. Tipos de atributos de activos o dispositivos

3.4.1.1. `__alarmas`: Actualmente myIoT sólo admite alarmas vinculadas a una única telemetría; estamos trabajando en crear alarmas mixtas que utilicen como criterios varias telemetrías del mismo dispositivo, o incluso de dispositivos diferentes. Los tipos de alarmas posibles son: `opciones/umbralMinimo/umbralMaximo/biUmbral`. Este tipo indica cómo se va a configurar el criterio de la alarma, pero no afecta a la lógica de la alarma en sí, que se definirá posteriormente en la cadena de reglas. Por ejemplo, una alarma de tipo `umbralMinimo` para una temperatura podría interpretarse en la cadena de reglas con una lógica que disparase la alarma si la temperatura bajase por debajo de ese umbral, o con una lógica que disparase la alarma si la diferencia de temperatura respecto a la medición anterior cayese por debajo de ese umbral (es decir, si la temperatura descendiese a un ritmo mayor del decremento indicado por el umbral). Adicionalmente existe una alarma de inactividad genérica, que no está vinculada a ninguna telemetría, sino a los mensajes de inactividad que genera automáticamente ThingsBoard. Las alarmas se almacenan en un atributo llamado `__alarma_[tipo de dispositivo]`; por ejemplo `__alarma_LHT65`.

Nombre alarma	Telemetría	Tipo alarma
nivelDeBateria	BatV	umbralMinimo
temperaturaSHT	TempC_SHT	biUmbral
humedadSHT	Hum_SHT	biUmbral
temperaturaDS	TempC_DS	biUmbral
nivel	Exti_pin_level	opciones (High/Low/Ambos)
interrupcion	Exti_status	opciones (Interrupción/Heartbeat/Ambos)
iluminacion	ILL_lux	biUmbral
tension	ADC_V	biUmbral
pulsos	Exti_count	umbralMaximo
conexion	No_connect	opciones (Conectado/No conectado/Ambos)



inactividad		
-------------	--	--

3.4.1.2. Internos: Son atributos que se utilizan internamente, por ejemplo, para indicar el URL de la imagen que queremos usar como fondo en el dashboard de un activo de tipo IMAGE01. El nombre de estos atributos debe comenzar forzosamente con un doble guión bajo (\_\_) para que myIoT los detecte y almacene. Adicionalmente, myIoT ofrece 2 tipos de atributos internos genéricos:

3.4.1.2.1. **coordenadas:** Permite configurar las coordenadas cuando el dispositivo se representa en un activo de tipo MAP01 o IMAGE01

3.4.1.2.2. **chirpstack:** En caso de que el dispositivo esté conectado a un servidor de red privado basado en ChirpStack, permite configurar el URL del servidor ChirpStack y el token JWT para enviar downlinks.

3.4.1.3. \_\_atributosCompartidos & \_\_ultimoDownlink: Son atributos que se comparten con los subordinados (delegados); por ejemplo, el periodo de envío de heartbeat. Es posible que algunos de ellos requieran un tratamiento adicional, como realizar un downlink para establecer el periodo de envío de heartbeat; este tratamiento especial se indica a través del atributo \_\_ultimoDownlink. Su configuración puede realizarse a través de elementos de los siguientes tipos: opciones/texto/numero/botón. Actualmente myIoT ofrece un atributo compartido genérico **devEUI**, que permite al dispositivo de control actuar como despachador para los demás dispositivos en base al devEUI configurado (esto nos permite, por ejemplo, incluir varios dispositivos, incluso de distinto tipo, en una misma aplicación de The Things Network y configurar una sola integración al dispositivo de control, en lugar de tener que configurar una integración individual para cada dispositivo). Con genérico nos referimos al hecho de que el propio sistema lo gestiona, sin que tengamos que desarrollar reglas adicionales (basta con utilizar como nombre para el atributo compartido devEUI y como valor para ultimoDownlink devEUI, como se verá posteriormente).

Atributo compartido	ultimoDownlink	Tipo	Descripción
TDC	TDC	numero	Transmit Time Interval (segundos)
RESET	RESET	botón	Reset
CFM	CFM	opciones: activado desactivado	Confirm Status
CHE	CHE	opciones: [0..8]	Eight Channel Mode. No para EU.
CLRDTA	CLRDTA	boton	Borrar datos almacenados
RTP	RTP	number	Record Time Period (minutos)

DATE	DATE	fecha !!!!	Fecha
EXT.EXT	EXT !!!	opciones: ninguno temperatura interrupción iluminación conversor ADC contador	Tipo de sensor conectado a la toma externa
EXT.EXT_INT_FLANCO	EXT !!!	opciones: subida bajada ambos	En el caso de un sensor de interrupción, establece en qué flanco se produce la interrupción
EXT.EXT_ADC_TIMEOUT	EXT !!!	numero	En el caso de un conversor ADC, establece con qué antelación (milisegundos) se alimenta antes de realizar la medición
EXT.EXT_CNT_FLANCO	EXT !!!	opciones: subida bajada	En el caso de un sensor de tipo contador, establece en qué flanco se incrementa la cuenta
EXT.EXT_CNT_SET	EXT !!!	numero	En el caso de un sensor de tipo contador, establece el valor inicial de la cuenta

3.4.1.3.1. Algunos tipos de dispositivos admiten **comandos de actuación**; por ejemplo, una válvula admite comandos para cerrarla y abrirla. Estos comandos se gestionan exactamente igual que los atributos compartidos del punto anterior, con la diferencia de que no almacenan ningún atributo, sino que simplemente envían el comando. Están asociados a un valor de ultimoDownlink para poder tratarlos por la misma subrama de la cadena de reglas que los atributos compartidos, pero este valor realmente no se almacena en el atributo ultimoDownlink (para no dar información no deseada a un usuario delegado que no tenga concedida esta capacidad).

3.4.2.El atributo de delegación: Crear en el **cliente\_patron** un atributo de servidor llamado **LHT65\_delegate**. Este atributo contendrá un documento XML con el que myIoT generará el cuadro de diálogo de delegación del dispositivo. El esquema de este documento XML es el que

se muestra a continuación, en el que lo más destacable es que el nombre debe coincidir con el de la telemetría, atributo compartido o ultimoDownlink que queramos controlar con él.

```
<myIoT>
  <delegacion
    nombre="[identificador (si aplica a un solo elemento de telemetría o atributo compartido se recomienda utilizar el nombre de esa telemetría o atributo compartido)]"
    label="[Etiqueta que se muestra junto a la casilla de activación] " />
  ...
</myIoT>
```

```
<myIoT>
  <delegacion nombre="BatV" label="Permitir ver la tensión de la batería" />
  <delegacion nombre="TempC_SHT" label="Permitir ver la temperatura del sensor interno" />
  <delegacion nombre="Hum_SHT" label="Permitir ver la humedad del sensor interno" />
  <delegacion nombre="Exti_status" label="Permitir ver si el estado del sensor externo es resultado de una interrupción o un envío periódico (sólo para sensor de tipo interrupción)" />
  <delegacion nombre="Ext_sensor" label="Permitir ver el tipo de sensor externo conectado" />
  <delegacion nombre="Ext_sensor_value" label="Permitir ver el valor del sensor externo" />
  <delegacion nombre="No_connect" label="Permitir ver el estado de conexión del sensor externo (no aplicable al DS18B20)" />
  <delegacion nombre="TDC" label="Permitir configurar el intervalo de transmisión (heartbeat)" />
  <delegacion nombre="RESET" label="Permitir resetear el dispositivo" />
  <delegacion nombre="CFM" label="Permitir configurar que los uplinks sean con o sin ACK" />
  <delegacion nombre="CHE" label="Permitir elegir un subconjunto de 8 canales para regiones distintas a EU" />
  <delegacion nombre="CLRDTA" label="Permitir borrar los datos almacenados en la memoria interna del dispositivo" />
  <delegacion nombre="RTP" label="Permitir configurar el periodo de grabación de datos en la memoria interna del dispositivo" />
  <delegacion nombre="DATE" label="Permitir configurar la fecha del dispositivo" />
  <delegacion nombre="EXT" label="Permitir configurar el sensor externo" />
</myIoT>
```

3.4.3.El atributo de configuración: Nuestro siguiente paso será crear en el **cliente\_patron** un atributo de servidor llamado **LHT65\_config**, pero antes debemos entender que este atributo contendrá un documento XML con el que myIoT generará el cuadro de diálogo de configuración del dispositivo. El esquema de este documento XML es el siguiente:

```
<myIoT>
  <panel titulo="" resumen=""
    nombreFormulario=""
    labelBotonSubmit="[Configurar]"
    ultimoDownlink="[Sólo se utiliza en atributosCompartidos]"
    [tipo="devEUI"]>

    <item tipo="coordenadas/chirpstack/alarma/atributoCompartido/atributoInterno"

      requiereDelegacion="[nombre de la delegación requerida]"
```

```

nombreAlarma=""
telemetria=""
labelAlarma=""
tipoAlarma="opciones/umbralMinimo/umbralMaximo/biUmbral"
labelAuxAlarma="Disparar al/expresado en segundos..."
histeresis="no/min/max/bi"
opciones="A/B/C"

nombreAtributo=""
labelAtributo=""
tipoAtributo="opciones/texto/numero/boton">

    <umbralMinimo min="" max="" step="" --> atributos estándar HTML />
    <umbralMaximo min="" max="" step="" --> atributos estándar HTML />
    <histeresisMinimo min="" max="" step="" --> atributos estándar HTML />
    <histeresisMaximo min="" max="" step="" --> atributos estándar HTML />

    <atributosHTML min="" max="" step="" size="" pattern="" --> atributos estándar HTML />

</item>
</panel>
</myIoT>

```

3.4.3.1. En un primer panel, llamado General, agruparemos todos los atributos internos y de alarma. Este panel no requiere el atributo ultimoDownlink porque myIoT se encarga automáticamente de almacenar los atributos internos y de alarma.

3.4.3.2. Cada atributo compartido o conjunto de atributos compartidos que se configuren a través de un mismo downlink requerirá un panel propio.

3.4.3.3. En las siguientes imágenes se muestran algunos ejemplos:



Activar alarmas de nivel de sensor de interrupción ☒ Verdadero

Disparar al

```

<item tipo="alarma" nombreAlarma="nivel"
  telemetria="Exti_pin_level"
  labelAlarma="nivel de sensor de interrupción"
  tipoAlarma="opciones"
  labelAuxAlarma="Disparar al"
  opciones="HIGH/LOW/Ambos">
</item>

```

telegram ☐ Falso

IFTTT ☐ Falso

Evento IFTTT

EDITAR DELEGAR CONFIGURAR BORRAR

### Configurar LHT65

Configuración general Configurar atributos de la entidad

Device EUI Indicar el Device EUI del dispositivo

Device EUI expresado en hexadecimal

CONFIGURAR

Heartbeat Configurar periodo de envío de heartbeat

CANCELAR

EDITAR DELEGAR CONFIGURAR BORRAR

### Configurar LHT65

Configuración general Configurar atributos de la entidad

Device EUI Indicar el Device EUI del dispositivo

Heartbeat Configurar periodo de envío de heartbeat

Número de segundos entre heartbeats.

CONFIGURAR

CANCELAR

Este atributo nos permitirá en las reglas aplicar la lógica que requiera esta configuración

```

<panel titulo="Heartbeat"
  resumen="Configurar periodo de envío de heartbeat"
  nombreFormulario="TDC" ultimoDownlink="TDC">
  <item
    tipo="atributoCompartido"
    nombreAtributoCompartido="TDC"
    labelAtributoCompartido="Número de segundos entre heartbeats."
    tipoAtributoCompartido="numero"
  >
    <atributoCompartido size="10" step="1" min="0" max="16777215" />
  </item>
</panel>

```

EDITAR DELEGAR CONFIGURAR BORRAR

### Configurar LHT65

Configuración general Configurar atributos de la entidad

Device EUI Indicar el Device EUI del dispositivo

Heartbeat Configurar periodo de envío de heartbeat

Resetear Reiniciar el dispositivo

RESETEAR

CANCELAR

```

<panel titulo="Resetear"
  resumen="Reiniciar el dispositivo"
  nombreFormulario="Resetear"
  labelBotonSubmit="Resetear"
  ultimoDownlink="resetear">
  <item tipo="atributoCompartido" nombreAtributoCompartido="resetear" tipoAtributoCompartido="boton" />
</panel>

```

**3.4.3.4. El contenido completo del atributo config es el siguiente (no se han incluido las configuraciones de DATE ni EXT porque requieren un tratamiento de más bajo nivel, que se abordará posteriormente).**

```
<myIoT>
  <panel titulo="Configuración general" resumen="Configurar atributos de la entidad"
nombreFormulario="General" labelBotonSubmit="Configurar">
    <item tipo="coordenadas" />
    <item tipo="chirpstack" />
    <item tipo="alarma" nombreAlarma="nivelDeBateria" telemetria="BatV" labelAlarma="nivel bajo
de batería" tipoAlarma="umbralMinimo" labelAuxAlarma="(V)" histeresis="min">
        <umbralMinimo size="10" min="0" max="5" step="0.1" />
    </item>
    <item tipo="alarma" nombreAlarma="temperaturaSHT" telemetria="TempC_SHT"
labelAlarma="temperatura interior" tipoAlarma="biUmbral" labelAuxAlarma="(°C) [-40..80]"
histeresis="bi">
        <umbralMinimo size="10" min="-40" max="80" step="0.5" />
        <umbralMaximo size="10" min="-40" max="80" step="0.5" />
    </item>
    <item tipo="alarma" nombreAlarma="humedadSHT" telemetria="Hum_SHT" labelAlarma="humedad
interior" tipoAlarma="biUmbral" labelAuxAlarma="(%)" [0..100]" histeresis="bi">
        <umbralMinimo size="10" min="0" max="100" step="1" />
        <umbralMaximo size="10" min="0" max="100" step="1" />
    </item>
    <item tipo="alarma" nombreAlarma="temperaturaDS" telemetria="TempC_DS"
labelAlarma="temperatura exterior" tipoAlarma="biUmbral" labelAuxAlarma="(°C) [-55..125]"
histeresis="bi">
        <umbralMinimo size="10" min="-55" max="125" step="0.5" />
        <umbralMaximo size="10" min="-55" max="125" step="0.5" />
    </item>
    <item tipo="alarma" nombreAlarma="nivel" telemetria="Exti_pin_level" labelAlarma="nivel de
sensor de interrupción" tipoAlarma="opciones" labelAuxAlarma="Disparar con el nivel"
opciones="HIGH/LOW/Ambos" />

    <item
        tipo="alarma"
        nombreAlarma="interrupcion"
        telemetria="Exti_status"
        labelAlarma="motivo de envío del nivel del sensor de interrupción"
        tipoAlarma="opciones"
        labelAuxAlarma="Disparar si el envío es por"
        opciones="Interrupción/Heartbeat/Ambos"
    />

    <item tipo="alarma" nombreAlarma="iluminacion" telemetria="ILL_lux" labelAlarma="nivel de
iluminación" tipoAlarma="biUmbral" labelAuxAlarma="(lx)" histeresis="bi">
        <umbralMinimo size="10" min="0" max="100000" step="1" />
        <umbralMaximo size="10" min="0" max="100000" step="1" />
    </item>

    <item tipo="alarma" nombreAlarma="tension" telemetria="ADC_V" labelAlarma="nivel de tensión"
tipoAlarma="biUmbral" labelAuxAlarma="(V)" histeresis="bi">
        <umbralMinimo size="10" min="0" max="3.3" step="0.1" />
        <umbralMaximo size="10" min="0" max="3.3" step="0.1" />
    </item>

    <item tipo="alarma" nombreAlarma="pulsos" telemetria="Exti_count" labelAlarma="pulsos"
tipoAlarma="umbralMaximo" labelAuxAlarma="(n° de pulsos)" histeresis="no">
        <umbralMaximo size="10" min="0" max="65535" step="1" />
    </item>

    <item tipo="alarma" nombreAlarma="conexion" telemetria="No_connect" labelAlarma="fallo de
conexión con el sensor exterior" tipoAlarma="opciones" labelAuxAlarma="Disparar si el sensor está"
opciones="Desconectado/Conectado/Ambos" />

    <item tipo="alarma" nombreAlarma="inactividad" />
</panel>

<panel tipo="devEUI" />
```

```

    <panel titulo="Heartbeat" resumen="Configurar periodo de envío de heartbeat"
nombreFormulario="TDC" ultimoDownlink="TDC">
    <item tipo="atributoCompartido" nombreAtributo="TDC" labelAtributo="Número de segundos entre
heartbeats." tipoAtributo="numero">
        <atributosHTML size="10" step="1" min="0" max="16777215" />
    </item>
</panel>

    <panel titulo="Resetear" resumen="Reiniciar el dispositivo" nombreFormulario="Resetear"
labelBotonSubmit="Resetear" ultimoDownlink="resetear">
    <item tipo="atributoCompartido" nombreAtributo="resetear" tipoAtributo="boton" />
</panel>

    <panel titulo="Confirmar uplinks" resumen="Activar la confirmación de los uplinks"
nombreFormulario="CFM" ultimoDownlink="CFM">
    <item tipo="atributoCompartido" labelAtributo="Confirmación de uplinks" nombreAtributo="CFM"
tipoAtributo="opciones" opciones="activada/desactivada" />
</panel>

    <panel titulo="Subconjunto de canales" resumen="Sólo necesario para regiones LoRaWAN distintas a
Europa" nombreFormulario="CHE" ultimoDownlink="CHE">
    <item tipo="atributoCompartido" labelAtributo="Subconjunto de canales (consulte documentación
comandos AT)" nombreAtributo="CHE" tipoAtributo="opciones" opciones="0/1/2/3/4/5/6/7/8" />
</panel>

    <panel titulo="Borrar almacenamiento" resumen="Borrar los datos almacenados en la memoria interna
del dispositivo" nombreFormulario="CLRDTA" labelBotonSubmit="Borrar" ultimoDownlink="CLRDTA">
    <item tipo="atributoCompartido" nombreAtributo="CLRDTA" tipoAtributo="boton" />
</panel>

    <panel titulo="Periodo de grabación" resumen="Configurar el periodo de grabación de datos en la
memoria interna" nombreFormulario="RTP" ultimoDownlink="RTP">
    <item tipo="atributoCompartido" nombreAtributo="RTP" labelAtributo="Periodo de grabación en
minutos (0 para deshabilitar)" tipoAtributo="numero">
        <atributosHTML size="10" step="1" min="0" max="65535" />
    </item>
</panel>
</myIoT>

```

3.5. La cadena de reglas: Aquí es donde se ejecuta toda la "lógica de negocio" del tipo de dispositivo. Tendremos que crear una regla cuyo nombre sea **LHT65**, pero en lugar de partir de cero, se recomienda copiar la regla LDS01.

3.5.1.Vincular la regla LHT65 recién creada en la regla switchTipoDispositivo.



3.5.2.El aspecto inicial de la cadena de reglas es el que se muestra a continuación, y sobre ella iremos realizando las modificaciones necesarias para adaptarla al nuevo tipo de dispositivo LHT65. Se han resaltado las zonas en las que actuaremos.





```

}
newMsg.Hum_SHT = parseInt(newMsg.payload_raw.substring(8,
12), 16) / 10.0;
auxiliar = parseInt(newMsg.payload_raw.substring(12, 14),
16) & 0x7F;
switch (auxiliar) {
case 0:
    newMsg.Ext_sensor = "No external sensor";
    break;
case 1:
    newMsg.Ext_sensor = "Temperature Sensor";
    auxiliar = parseInt(newMsg.payload_raw.substring(14,
18),
16);
    if (auxiliar >= 65536) {
        newMsg.TempC_DS = (auxiliar - 65536) / 100.0;
    } else {
        newMsg.TempC_DS = auxiliar / 100.0;
    }
    break;
case 4:
    newMsg.Ext_sensor = "Interrupt Sensor send";
    if (parseInt(newMsg.payload_raw.substring(14,
16),
16) == 1) {
        newMsg.Exti_pin_level = 'High';
    } else {
        newMsg.Exti_pin_level = 'Low';
    }
    if (parseInt(newMsg.payload_raw.substring(16,
18),
16) == 1) {
        newMsg.Exti_status = 'True';
    } else {
        newMsg.Exti_status = 'False';
    }
    if (parseInt(newMsg.payload_raw.substring(12, 14),
16) & 0x80 == 0x80) {
        newMsg.No_connect = 'Sensor no connection';
    } else {
        newMsg.No_connect = 'Sensor connection';
    }
    break;
case 5:
    newMsg.Ext_sensor = "Illumination Sensor";
    newMsg.ILL_lux = parseInt(newMsg.payload_raw
.substring(14, 18),
16);
    if (parseInt(newMsg.payload_raw.substring(12, 14),
16) & 0x80 == 0x80) {
        newMsg.No_connect = 'Sensor no connection';
    } else {
        newMsg.No_connect = 'Sensor connection';
    }
    break;
case 6:
    newMsg.Ext_sensor = "ADC Sensor";
    newMsg.ADC_V = parseInt(newMsg.payload_raw
.substring(14, 18),
16) / 1000.0;
    if (parseInt(newMsg.payload_raw.substring(12, 14),
16) & 0x80 == 0x80) {
        newMsg.No_connect = 'Sensor no connection';
    } else {
        newMsg.No_connect = 'Sensor connection';
    }
    break;
case 7:
    newMsg.Ext_sensor = "Interrupt Sensor count";
    newMsg.Exti_count = parseInt(newMsg.payload_raw
.substring(14, 18),

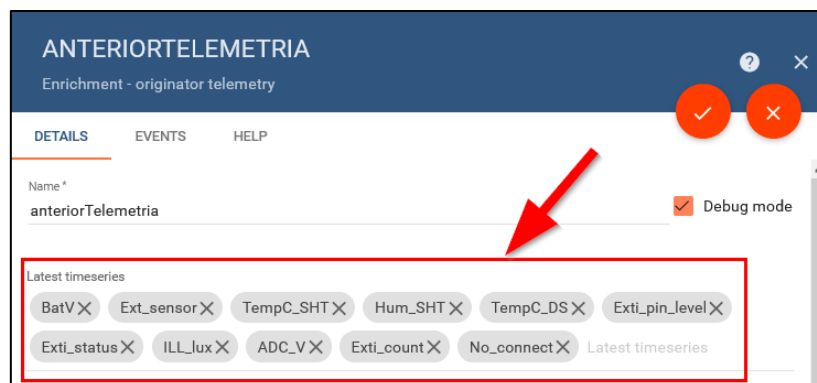
```

```

        16);
    if (parseInt(newMsg.payload_raw.substring(12, 14),
        16) & 0x80 == 0x80) {
        newMsg.No_connect = 'Sensor no connection';
    } else{
        newMsg.No_connect = 'Sensor connection';
    }
    break;
}
metadata.nombresTelemetrias = "BatV/TempC_SHT";
metadata.correspondenciaDelegacionesTelemetrias = JSON
.stringify({
    "BatV": ["BatV"],
    "TempC_SHT": ["TempC_SHT"],
    "Hum_SHT": ["Hum_SHT"],
    "Exti_status": ["Exti_status"],
    "Ext_sensor": ["TempC_DS", "Exti_pin_level",
        "ILL_lux", "ADC_V", "Exti_count"
    ],
    "No_connect": ["No_connect"]
});
//Final de la conversión de la carga de pago RAW en telemetrias

```

**3.5.3.2. Originator Telemetry: anteriorTelemetry:** En este nodo tenemos que cargar las últimas telemetrías recibidas para que, en caso de que alguna de ellas no esté delegada, no se actualice nuevamente con el valor "No delegado" (esta actualización podría desvelar información que no queremos delegar; por ejemplo, no hemos delegado el estado de una puerta, pero el usuario delegado podría recibir actualizaciones de que su estado ha cambiado).



**3.5.3.3. Script: Tooltip + Downlink:** En este nodo construiremos el tooltip que queramos mostrar en los activos de tipo IMAGE01 o MAP01 cuando el dispositivo les tenga como padres.

```

//Inicio de construcción del tooltip
var newMsg = {};
newMsg.tooltip = "<b>" + year + "-" + month + "-" + date +
    " " + hours + ":" + minutes + ":" + seconds +
    "</b><br/>" + "<b>Temp. interna: </b>" + (msg
    .TempC_SHT == undefined ? 'No delegado' : msg
    .TempC_SHT) + " °C" +
    "<br/><b>Hum. interna: </b>" + (msg.Hum_SHT ==
    undefined ? 'No delegado' : msg.Hum_SHT) + " %" +
    "<br/><b>Sensor externo: </b>";
if (msg.TempC_DS != undefined) {
    newMsg.tooltip += msg.TempC_DS + " °C";
} else if (msg.Exti_pin_level != undefined) {
    newMsg.tooltip += msg.Exti_pin_level;
} else if (msg.ILL_lux != undefined) {

```


```

    newMsg.tooltip += msg.ILL_lux + " lx";
} else if (msg.ADC_V != undefined) {
    newMsg.tooltip += msg.ADC_V + " V";
} else if (msg.ExtI_count != undefined) {
    newMsg.tooltip += msg.ExtI_count + " pulsos";
}
newMsg.tooltip += "<br/><b>Batería: </b>" + (msg
.BatV == undefined ? 'No delegado' : (msg.BatV) +
" V");
//Final de construcción del tooltip

```

3.5.3.4. Ya tenemos la parte principal de la regla modificada. Vamos a verificar su funcionamiento simulando el envío de una telemetría, que además nos servirá para revisar el procedimiento de integración con The Things Network.

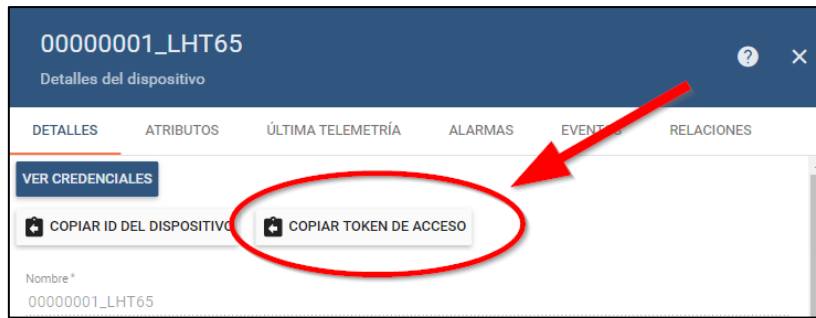
3.5.3.4.1. Creamos un dispositivo nuevo de tipo LHT65 con la cuenta customer [usuario@\[sunombre\].com](mailto:usuario@[sunombre].com) (a través de su panel Configuración) que configuramos anteriormente.



3.5.3.4.2. Verificamos que los paneles Delegar y Configurar se han "parseado" correctamente a partir de los atributos LHT65\_delegate y LHT65\_config que creamos anteriormente en el cliente patrón, y que se han copiado automáticamente en el cliente 00000001 cuando éste ha creado el dispositivo de tipo LHT65.



3.5.3.4.3. Copiamos el token de acceso al dispositivo, que necesitaremos al crear la integración en The Things Network.



3.5.3.4.4. Creamos una aplicación nueva en TTN y, dentro de ella, un dispositivo (sus nombres y demás configuraciones son indiferentes para esta prueba).

3.5.3.4.5. Creamos en la aplicación una integración de tipo HTTP configurada como se muestra en la siguiente imagen.

3.5.3.4.6. En el panel Overview del dispositivo creado en TTN, simular el uplink de la carga de pago cbe508fd01bb01021f7fff.

3.5.3.4.7. Si todo es procedimiento anterior se ha realizado correctamente, al revisar los eventos de nodo RAW2JSON de la rama de telemetrías deberíamos ver un mensaje de entrada y su correspondiente salida, en la que encontraremos los

campos de la telemetría correctamente extraídos de la carga de pago en bruto que hemos enviado.

The screenshot displays the Thingiverse interface. On the left, a workflow editor shows a sequence of steps: 'check alarm status', 'check existence fields', 'check relation', 'gps geofencing filter', 'message type', 'message type switch', 'originator type', 'originator type switch', 'script', and 'switch'. The 'script' step is highlighted with a red circle and the number 1. The main panel shows a 'RAW2JSON' transformation script. Below the script, a table of events is displayed. The table has columns: Event time, Server, Type, Entity, Message Id, Message, Relation, Type, Data, Metadata, and Error. Two events are listed, both from 'ubuntu-2gb-nbg1-1'. The first event is an 'OUT' type with a 'Success' relation. The second event is an 'IN' type with a 'TTN' relation. A red circle and the number 2 highlight the 'Data' column. A red arrow points from the 'Data' column to a 'Data' viewer window. The viewer shows a JSON object with various fields, including 'payload\_raw' and 'payload'. A red circle and the number 3 highlight the 'payload' field.

Event time	Server	Type	Entity	Message Id	Message	Relation	Type	Data	Metadata	Error
2020-10-13 19:46:25	ubuntu-2gb-nbg1-1	OUT	DEVICE	27c083f5-1...	POST_TELE...	Success		...	...	
2020-10-13 19:46:25	ubuntu-2gb-nbg1-1	IN	DEVICE	27c083f5-1...	POST_TELE...	TTN		...	...	

```

{
  "hardware_serial": "0000000000000000",
  "counter": 0,
  "port": 1,
  "metadata": {
    "class": "2020-10-13T17:46:25.186679532"
  },
  "payload_raw": "0x0000000000000000",
  "payload": {
    "freq": 3.045,
    "freq_offset": 23.03,
    "new_snr": 44.3,
    "rx_sensor": "temperature sensor",
    "temp": 25.5,
    "temp_offset": 0.0
  },
  "thingiverse": {
    "downloadLink": {
      "downloadUrl": "https://integrations.thethingnetwork.org/ttn-eu-api/v2/down/jfateos_curse_myiot/jfateos_tkkey=ttn-account-v2.FK03_sp-hdStB1"
    },
    "ttn_dev_id": "jfateos_curse_myiot_kh85"
  }
}

```

3.5.4. Rama downlink: En esta rama se tratan los atributos compartidos y los comandos de actuación. La diferencia entre ambos es muy sutil: básicamente los atributos compartidos almacenan siempre un valor en `__ultimoDownlink` (además de, según el caso, una propiedad en `__atributosCompartidos`) y los comandos de actuación no. Por ejemplo, un atributo compartido podría ser la periodicidad de envío de heartbeats de un nodo o una orden de reseteo, mientras que un comando de actuación podría ser la solicitud de cerrar una válvula. En esta rama sólo tendremos que preocuparnos de 2 nodos: operaciones y prepararDownlink.

3.5.4.1. **Nodo switch: operaciones:** En este nodo se determina si estamos ante un atributo compartido o un comando de actuación. En el primer caso se deriva la ejecución tanto a la subrama guardarAtributo (para guardar el atributo `___ultimoDownlink` y, si corresponde, el valor correspondiente en `___atributosCompartidos`), como a la subrama propia del downlink (u otra operación) necesario para ese atributo compartido. En el segundo caso, el de un comando de actuación, sólo se deriva la ejecución a la subrama propia del downlink (no a la de guardar el atributo). Para nuestro caso, el código de este nodo quedaría así:

```
switch (msg.ultimoDownlink) {
    case 'TDC':
        return ['guardarAtributo', 'TDC'];
        break;
    case 'RESET':
        return ['guardarAtributo', 'RESET'];
        break;
    case 'CFM':
        return ['guardarAtributo', 'CFM'];
        break;
    case 'CHE':
```

```

        return ['guardarAtributo', 'CHE'];
        break;
    case 'CLRDTA':
        return ['guardarAtributo', 'CLRDTA'];
        break;
    case 'RTP':
        return ['guardarAtributo', 'RTP'];
        break;
}

```

**3.5.4.2. Nodos script:prepararDownlink:** En estos nodos se realiza realmente la operación que requiera el atributo compartido o comando de actuación, que generalmente será enviar un downlink. A continuación se muestra el código correspondiente a TDC, y los demás pueden consultarse en la regla terminada, que está disponible en el repositorio.

```

var payload = {};

payload.port = 2; //cualquiera
payload.confirmed = false;
payload.schedule = 'replace';

var valor = parseInt(msg.___atributosCompartidos.TDC)
    .toString(16);
var pad = '000000';

payload.payload_raw = "01" + pad.substring(0, pad
    .length -
    valor.length) + valor;
msg = {
    "payload": JSON.stringify(payload),
    "uuid": msg.uuid
};
return {
    msg: msg,
    metadata: metadata,
    msgType: msgType
};

```

**3.5.5. Rama alarmas:** En esta rama definiremos la lógica de cada una de las alarmas que identificamos anteriormente para las telemetrías. Es necesario entender que en ThingsBoard las alarmas se crean, y a continuación pueden ser atendidas (ACK) y borradas (bien por el usuario, o bien mediante reglas). Mientras una alarma está activa (después de crearla), incluso aunque el usuario la marque como atendida, no se generarán alarmas del mismo tipo. En otras palabras: es necesario que la alarma activa se borre para re-armar ese tipo de alarma. Cada subrama de alarma está compuesta por 4 nodos que desembocan en el nodo Notificaciones. El primero es de tipo script y sirve para almacenar en los metadatos la información necesaria para componer el mensaje de notificación de la alarma (por ejemplo, si es una alarma de exceso de temperatura, almacena en los metadatos la temperatura y que el tipo de alarma es "exceso de temperatura". Es importante que esta información esté en los metadatos, porque en el segundo nodo, que es de tipo switch, es dónde se encuentra la lógica de la alarma y se determina si hay que crearla o borrarla, dando paso a continuación al tercer nodo, que es el que efectivamente crea o borra la alarma y sustituye el cuerpo del mensaje por la propia información de la alarma. Almacenando la información en los metadatos hemos conseguido que llegue al cuarto nodo, que es de tipo script, y sirve para componer el mensaje de la/s notificación/es. Obsérvese que las alarmas biUmbral tienen 2 subramas, pues una misma telemetría podría a la vez crear y borrar una alarma (por ejemplo, si tenemos un umbral mínimo de 20 y uno máximo de 25, se ha generado una alarma de "temperatura muy baja" por

una telemetría de valor 18 y, a continuación, recibimos una telemetría de valor 26, debería borrarse la alarma anterior de "temperatura muy baja" y generar una nueva de "temperatura demasiado alta". Téngase en cuenta que la subrama de la alarma "inactividad" no requiere ninguna modificación porque se apoya en el sistema intrínseco de alarmas de inactividad de ThingsBoard (no requiere una lógica expresa).

**3.5.5.1. Switch: Alarmas:** En este nodo derivaremos la ejecución a la/s sub-rama/s responsable/s de tratar cada alarma, validando previamente que el mensaje contiene la telemetría necesaria y que la alarma está activada. El código quedaría del siguiente modo:

```
var respuesta = [];
if (metadata.hasOwnProperty('ss__alarma_'+metadata.deviceType)) {
    var obj = JSON.parse(metadata['ss__alarma_'+metadata.deviceType]);

    if
(obj.hasOwnProperty('nivelDeBateria') && msg.hasOwnProperty('BatV') && obj.nivelDeBateria.enable===true)
    {
        respuesta.push('nivelDeBateria');
    }
    if
(obj.hasOwnProperty('temperaturaSHT') && msg.hasOwnProperty('TempC_SHT') && obj.temperaturaSHT.enable===true) {
        respuesta.push('temperaturaSHT');
    }
    if
(obj.hasOwnProperty('humedadSHT') && msg.hasOwnProperty('Hum_SHT') && obj.humedadSHT.enable===true) {
        respuesta.push('humedadSHT');
    }
    if
(obj.hasOwnProperty('temperaturaDS') && msg.hasOwnProperty('TempC_DS') && obj.temperaturaDS.enable===true)
    {
        respuesta.push('temperaturaDS');
    }
    if (obj.hasOwnProperty('nivel') && msg.hasOwnProperty('Exti_pin_level') && obj.nivel.enable===true) {
        respuesta.push('nivel');
    }
    if
(obj.hasOwnProperty('interrupcion') && msg.hasOwnProperty('Exti_status') && obj.interrupcion.enable===true)
    {
        respuesta.push('interrupcion');
    }
    if
(obj.hasOwnProperty('iluminacion') && msg.hasOwnProperty('ILL_lux') && obj.iluminacion.enable===true) {
        respuesta.push('iluminacion');
    }
    if (obj.hasOwnProperty('tension') && msg.hasOwnProperty('ADC_V') && obj.tension.enable===true) {
        respuesta.push('tension');
    }
    if (obj.hasOwnProperty('pulsos') && msg.hasOwnProperty('Exti_count') && obj.pulsos.enable===true) {
        respuesta.push('pulsos');
    }
    if (obj.hasOwnProperty('conexion') && msg.hasOwnProperty('No_connect') && obj.conexion.enable===true)
    {
        respuesta.push('conexion');
    }
    //La alarma de inactividad es genérica para todos los tipos de dispositivos y no requiere
    modificación
    if
(obj.hasOwnProperty('inactividad') && (msgType=='INACTIVITY_EVENT' || msgType=='ACTIVITY_EVENT') && obj.ina
ctividad.enable===true) {

        respuesta.push('inactividad');
    }
}
```

```
return respuesta;
```

3.5.5.2. A continuación habría que modificar los nodos de cada sub-rama de alarma; esta modificación dependerá del tipo de alarma (opciones, umbralMinimo, umbralMaximo...). Aquí vamos a desarrollar el caso de la alarma "temperaturaDS" que es de tipo biUmbral y con biHisteresis (en la regla terminada, que está disponible en el repositorio, pueden consultarse otros tipos). Como tiene 2 umbrales, necesitaremos 2 subramas con la misma etiqueta "temperaturaDS", una para el umbral máximo y otra para el umbral mínimo.

3.5.5.2.1. Nodos script de almacenamiento en los metadatos de la información necesaria para construir el mensaje de la alarma. Almacenamos la telemetría, la configuración de la alarma, un nombre para el tipo de alarma y los umbrales. Para el caso de la primera sub-rama (temperatura máxima) quedaría como se muestra a continuación, y para el de temperatura mínima sería igual salvo cambiando el tipoAlarma.

```
var obj = JSON.parse(metadata['ss___alarma_' + metadata.deviceType]);

//Cargamos la alarma en el mensaje para que el nodo siguiente
//pueda ejecutar la "lógica de negocio"
msg.alarma = obj.temperaturaDS;

//Cargamos los datos de interés en los metadatos para construir los mensajes
//de alarma tras el nodo que genera/borrar la alarma
metadata.TempC_DS = msg.TempC_DS;
//alarmaActual se usa en la cadena de reglas de Notificaciones
//Para extraer información. No puede ir por msg porque se
//pierde en el nodo de generación de alarma
metadata.alarmaActual = JSON.stringify(obj.temperaturaDS);
metadata.tipoAlarma = "Temperatura exterior superior al umbral máximo";
metadata.umbralMinimo = msg.alarma.umbralMinimo;
metadata.umbralMaximo = msg.alarma.umbralMaximo;
return {
  msg: msg,
  metadata: metadata,
  msgType: msgType
};
```

3.5.5.2.2. Nodos switch para crear o clear las alarmas: Aquí es donde se ejecuta la lógica de la alarma. Para la subrama de temperatura máxima quedaría así:

```
if (metadata.TempC_DS > msg.alarma.umbralMaximo) {
  return ['crear'];
} else if (metadata.TempC_DS < msg.alarma.umbralMaximo - msg.alarma.histeresisMaximo) {
  return ['clear'];
}
```

3.5.5.2.3. Nodos script para construir los mensajes de las notificaciones. En estos nodos construiremos el mensaje que necesitan cada uno de los tipos de notificaciones que tenemos actualmente (mail, Telegram e IFTTT). Para el caso de creación de la alarma de superación de la temperatura máxima quedaría así:

```
metadata.mensajeTelegram="El dispositivo " +
metadata.deviceName.substring(metadata.deviceName.indexOf("_")+1)+" ha generado una alarma de tipo
"+metadata.tipoAlarma+". La temperatura es "+metadata.TempC_DS+" °C y el umbral es de
"+metadata.umbralMaximo+" °C.";

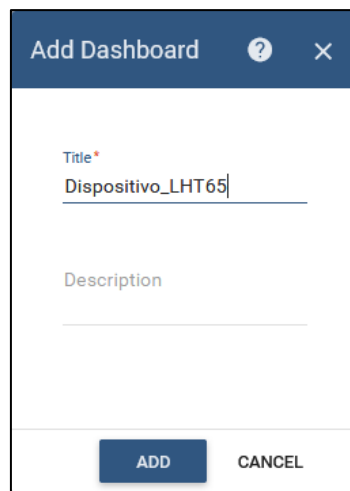
metadata.mensajeIfttt='{"value1":"' + metadata.deviceName.substring(metadata.deviceName.indexOf("_")+1)
+ '", "value2":"' + metadata.TempC_DS + '"}';
```



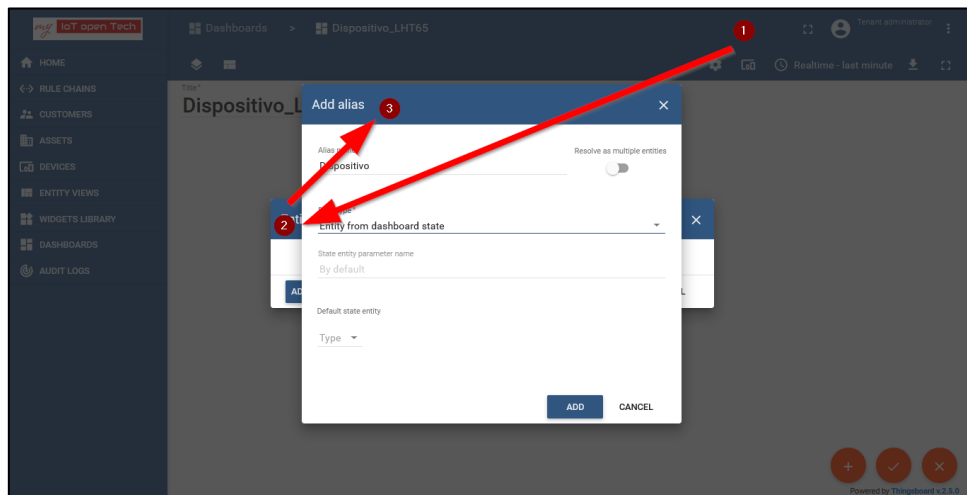
```
metadata.mensajeEmail=metadata.mensajeTelegram;  
metadata.asuntoEmail="[IoT open Tech]  
"+metadata.deviceName.substring(metadata.deviceName.indexOf("_")+1)+": Alarma "+metadata.tipoAlarma;  
return {msg: msg, metadata: metadata, msgType: msgType};
```

3.5.5.2.4. Completar el resto de sub-ramas de alarmas. En realidad es básicamente un proceso de "copia/pega" en el que hay que prestar atención a los detalles resaltados en amarillo en los tres fragmentos de código anteriores.

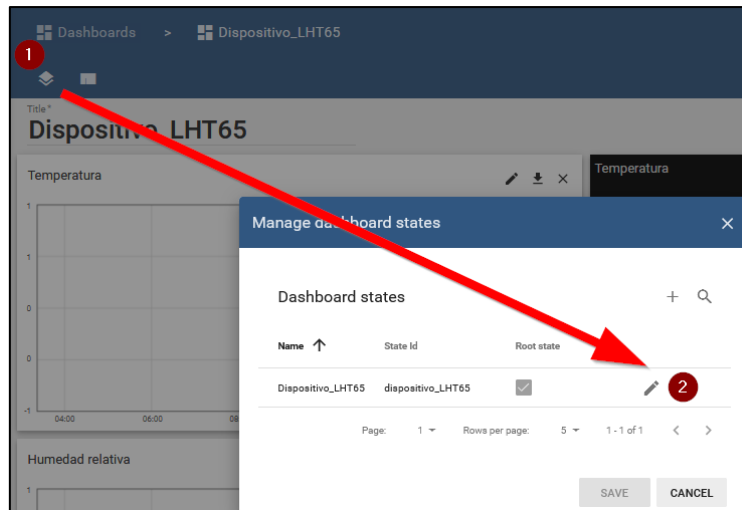
3.6. El dashboard: Cuando el customer haga clic sobre un dispositivo del tipo LHT65, myIoT intentará abrir un panel llamado "Dispositivo\_LHT65". Empezar creando este panel, y a continuación, definir el alias que tome la entidad activa del estado del panel.



The screenshot shows a modal dialog titled "Add Dashboard". It has a "Title" field with the text "Dispositivo\_LHT65" and a "Description" field which is currently empty. At the bottom of the dialog are two buttons: "ADD" and "CANCEL".



3.6.1. Configurar el estado raíz con el nombre Dispositivo\_LHT65.

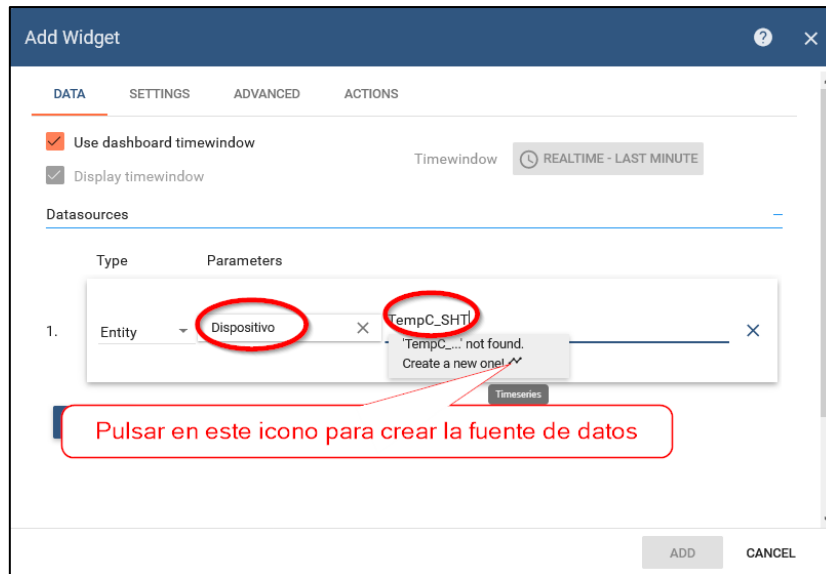


3.6.2. Asignar el panel Dispositivo\_LHT65 tanto al cliente patrón como al cliente 00000001. Lo normal sería asignárselo sólo al cliente patrón, pues a partir de éste lo adquiriría cualquier otro customer que crease un dispositivo de este tipo. Pero, en este caso, como ya hemos creado anteriormente un dispositivo LHT65 con el cliente 00000001 para probar el funcionamiento de la sub-rama de telemetría de la cadena de reglas, tendremos que asignárselo también a él.

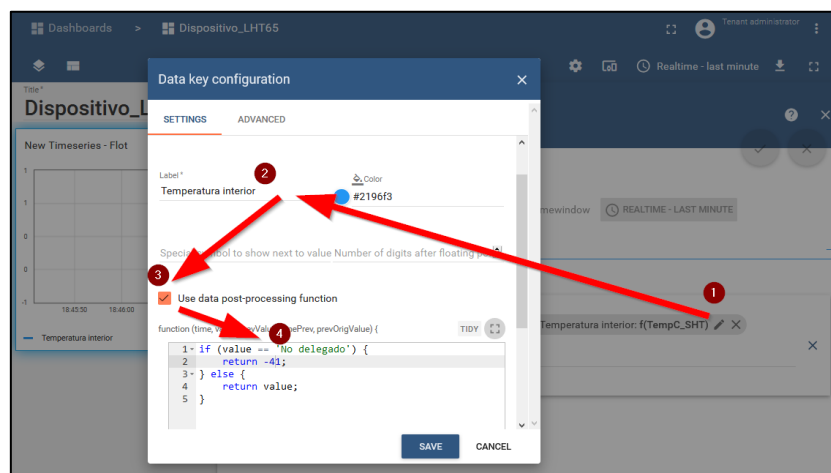
3.6.3. Inserción de widgets. En esta sección veremos cómo añadir una pareja de widgets para el sensor de temperatura interior (TempC\_SHT): uno con una gráfica del histórico de valores, y otro con un reloj analógico/digital que represente el valor actual. El procedimiento es muy similar para el resto de telemetrías, y puede utilizar como referencia el dashboard terminado que está disponible en el repositorio. Insertar un widget de tipo Charts > Timeseries.



3.6.4. Crear la fuente de datos para el widget, que será la telemetría TempC\_SHT de la entidad representada por el alias "Dispositivo".



3.6.5. Si un usuario delegado accede a este panel y no tiene delegada esta telemetría, el valor de la telemetría será "No delegado". Para que este valor se muestre en la zona inferior del eje de ordenadas de la gráfica vamos a convertirlo en -41, que es un grado menos que el valor mínimo que puede enviar el sensor SHT. Hacer clic sobre el icono de configuración de la fuente de datos (icono con forma de lápiz), y asignarle la siguiente función de post-procesamiento (también aprovechar para asignarle el label "Temperatura interior").

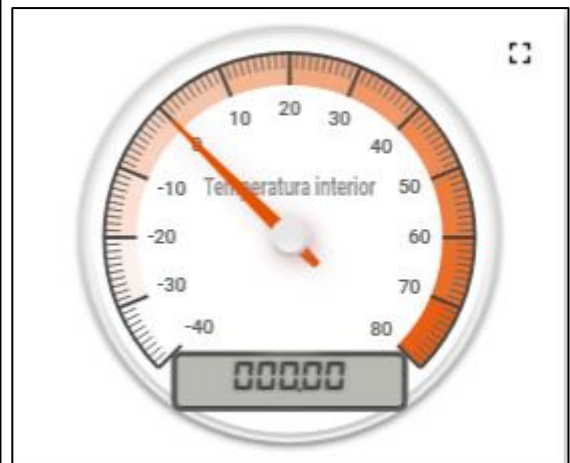
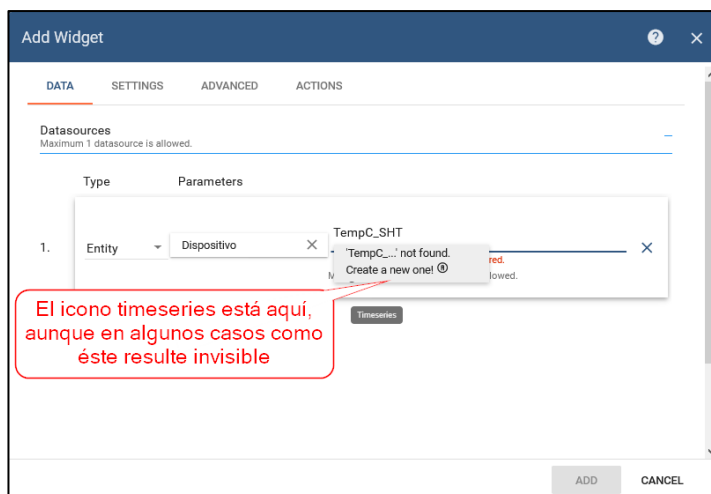


3.6.6. En el panel Settings, indicar "Temperatura interior" como Título del widget.

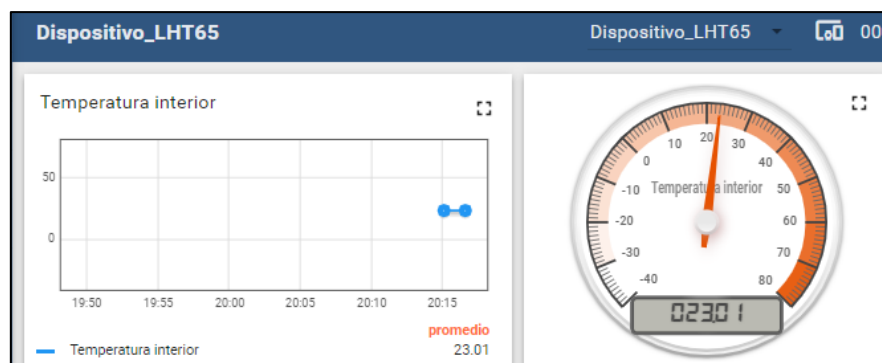
3.6.7. En el panel Advanced, indicar tanto en la función de valor de Tooltip (Tooltip value format function) como en la función de formato de ticks (Text formatter function) el código que se muestra a continuación. Si se desea, pueden configurarse otros parámetros avanzados como los valores mínimo/máximo del eje Y (-41 y 80, respectivamente para el sensor SHT).



3.6.8. Insertar un widget Analog gauges > Temp y configurar su origen de datos exactamente igual que se hizo en el widget anterior. En este caso no nos servirá de nada la función de post-procesamiento porque este widget no nos permite mostrar textos (sólo números). Se puede configurar el fondo de escala de 40 a 80 con 12 divisiones principales para obtener un resultado similar al que se muestra a continuación.



3.6.9. Simular nuevamente el envío de la carga de pago cbe508fd01bb01021f7fff desde The Things Network y comprobar que el panel refleja el valor de 23.01 °C.



3.7. Cuestiones pendientes: Tenemos pendiente de terminar la sección de configuración de los atributos DATE y EXT porque el parseador del atributo \_config no reconoce items de tipo fecha/hora, ni items de opciones en los que sus valores disponibles varíen en función de los valores elegidos en otras opciones. Sin embargo, el parseador puede inhibirse, tanto a nivel de panel como a nivel de item; esto quiere decir que si el parseador encuentra en el atributo \_config un elemento que no es de tipo <panel> en el lugar donde debería ir un panel, lo copiará directamente en el código HTML del formulario de configuración (igualmente para <item>).

3.7.1. Completar, en los customers patrón y 00000001, el atributo LHT65\_config como se muestra a continuación (obsérvese que el parseador no puede aceptar el carácter &, por lo que es necesario codificarlo como una entidad &amp;):

```
...
<panel titulo="Periodo de grabación" resumen="Configurar el periodo de grabación de datos en la
memoria interna" nombreFormulario="RTP" ultimoDownlink="RTP">
  <item tipo="AtributoCompartido" nombreAtributo="RTP" labelAtributo="Periodo de grabación en
minutos (0 para deshabilitar)" tipoAtributo="numero"> <atributosHTML size="10" step="1" min="0"
max="65535" /> </item>
</panel>
<panel titulo="Fecha y hora" resumen="Configurar fecha y hora" nombreFormulario="DATE"
ultimoDownlink="DATE">
  <div class="row" layout="row">
    <section layout="row" layout-align="start start">
      <mdp-date-picker ng-model="vm.configuracion.____atributosCompartidos.DATE" mdp-
placeholder="Fecha"></mdp-date-picker>
      <mdp-time-picker ng-model="vm.configuracion.____atributosCompartidos.DATE" mdp-
placeholder="Hora" mdp-auto-switch="true"></mdp-time-picker>
    </section>
  </div>
</panel>
<panel titulo="Sensor externo" resumen="Elegir el tipo de sensor externo" nombreFormulario="EXT"
ultimoDownlink="EXT">
  <div class="row" layout="row">
    <md-input-container style="margin: 0px; margin-top: 10px;">
      <label>Tipo de sensor</label>
      <md-select ng-model="vm.configuracion.____atributosCompartidos.EXT" placeholder="Tipo de
sensor" class="md-no-underline">
        <md-option value="0">Ninguno</md-option> <md-option value="1">Temperatura</md-option>
<md-option value="4">Interruptor</md-option> <md-option value="5">Iluminación</md-option> <md-option
value="6">Convertor ADC</md-option>
        <md-option value="7">Contador de pulsos</md-option>
      </md-select>
    </md-input-container>
    <md-input-container ng-if="vm.configuracion.____atributosCompartidos.EXT=='4'" style="margin:
0px; margin-top: 10px;">
      <label>Flanco</label>
      <md-select ng-required="vm.configuracion.____atributosCompartidos.EXT=='4'" ng-
model="vm.configuracion.____atributosCompartidos.EXT_INT_FLANCO" placeholder="Flanco" class="md-no-
underline">
        <md-option value="1">Ambos</md-option> <md-option value="2">Bajada</md-option> <md-
option value="3">Subida</md-option>
      </md-select>
    </md-input-container>
    <md-input-container ng-if="vm.configuracion.____atributosCompartidos.EXT=='6'" style="margin:
0px; margin-top: 10px;">
      <label>Precalentamiento (ms)</label> <input type="number" step="1" min="0" max="65535"
ng-model="vm.configuracion.____atributosCompartidos.EXT_ADC_TIMEOUT" ng-
required="vm.configuracion.____atributosCompartidos.EXT=='6'" />
    </md-input-container>
    <md-input-container ng-if="vm.configuracion.____atributosCompartidos.EXT=='7'" style="margin:
0px; margin-top: 10px;">
      <label>Flanco o pulsos iniciales</label>
      <md-select ng-model="vm.configuracion.____atributosCompartidos.EXT_CNT_FLANCO"
placeholder="Flanco" class="md-no-underline" ng-
required="vm.configuracion.____atributosCompartidos.EXT=='7'">
```

```

        <md-option value="0">Bajada</md-option> <md-option value="1">Subida</md-option><md-
option value="2">Establecer pulsos iniciales</md-option>
    </md-select>
</md-input-container>
    <md-input-container ng-if="vm.configuracion.___atributosCompartidos.EXT=='7' &amp;&amp;
vm.configuracion.___atributosCompartidos.EXT_CNT_FLANCO=='2'" style="margin: 0px; margin-top: 10px;">
        <label>Pulsos iniciales </label>
        <input type="number" step="1" min="-1" max="65535" ng-
model="vm.configuracion.___atributosCompartidos.EXT_CNT_SET" ng-
required="vm.configuracion.___atributosCompartidos.EXT=='7' &amp;&amp;
vm.configuracion.___atributosCompartidos.EXT_CNT_FLANCO=='2'" />
    </md-input-container>
</div>
</panel>
</myIoT>

```

3.7.2. Funciones. Hay ocasiones en la que la inhibición del parseador no es suficiente y necesitamos insertar en el cuadro de diálogo de configuración, además de código HTML, código JavaScript. Por ejemplo, ocurre con el atributo compartido DATE que acabamos de incluir. El código HTML (mdp-date-picker y mdp-time-picker) que permite configurar este atributo requiere una variable JavaScript de tipo Date. Si necesitamos incluir funciones JavaScript en el archivo de configuración podremos hacerlo colocándolas al principio, como un array JSON de pares {"nombreFuncion": "cuerpoFuncion"}, y dentro de un elemento <funciones>. El nombre de función "inicializacion" está reservado; la función que definamos con este nombre se ejecutará automáticamente al instanciar el cuadro de diálogo de configuración (es justo lo que necesitamos para el atributo DATE). Incluir el siguiente elemento funciones al principio de los atributos LHT65\_config del cliente patrón y el cliente 00000001.

```

<funciones>
[{"nombre": "inicializacion", "codigo": "$scope.vm.configuracion.___atributosCompartidos.DATE=new
Date($scope.vm.configuracion.___atributosCompartidos.DATE);"}]</funciones>
<myIoT>
    <panel titulo="Configuración general" resumen="Configurar atributos de la entidad"
nombreFormulario="General" labelBotonSubmit="Configurar">
        <item tipo="coordenadas" /> <item tipo="chirpstack" />
    ...

```

3.7.3. Sub-rama downlink. Necesitamos atender estos 2 nuevos atributos (DATE y EXT) en la sub-rama downlink de la cadena de reglas del dispositivo LHT65.

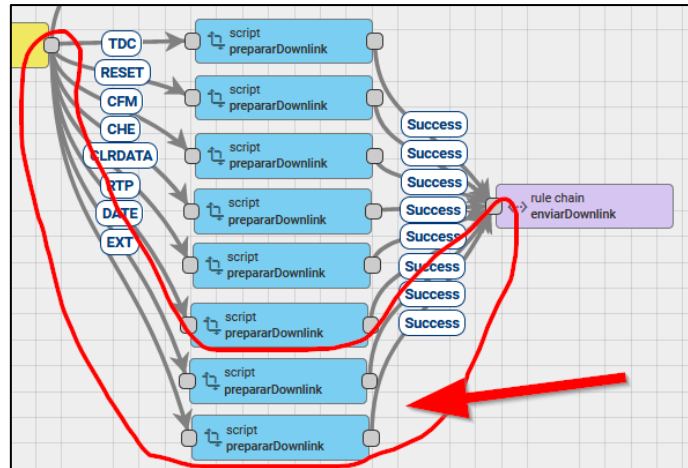
3.7.3.1. Añadir el siguiente código en el nodo switch:operaciones:

```

...
case 'RTP':
    return ['guardarAtributo', 'RTP'];
    break;
case 'DATE':
    return ['guardarAtributo', 'DATE'];
    break;
case 'EXT':
    return ['guardarAtributo', 'EXT'];
    break;
}

```

3.7.3.2. Añadir los 2 nodos siguientes con el código que se indica:



### Nodo DATE

```
var payload = {};

payload.port = 2; //cualquiera
payload.confirmed = false;
payload.schedule = 'replace';

var valor = new Date(msg.___atributosCompartidos.DATE);
var pad = "00";
payload.payload_raw = "A1";
var agno = String(valor.getFullYear()).substring(2);
var mes = String(valor.getMonth() + 1);
var dia = String(valor.getDate());
var hora = String(valor.getHours());
var minuto = String(valor.getMinutes());
var segundo = String(valor.getSeconds());
payload.payload_raw += agno;
payload.payload_raw += pad.substring(0, mes
    .length -
    mes.length) + mes;
payload.payload_raw += pad.substring(0, dia
    .length -
    mes.length) + dia;
payload.payload_raw += pad.substring(0, hora
    .length -
    mes.length) + hora;
payload.payload_raw += pad.substring(0, minuto
    .length -
    mes.length) + minuto;
payload.payload_raw += pad.substring(0, segundo
    .length -
    mes.length) + segundo;
msg = {
    "payload": JSON.stringify(payload),
    "uuid": msg.uuid
};
return {
    msg: msg,
    metadata: metadata,
    msgType: msgType
};
```

### Nodo EXT

```
var payload = {};

payload.port = 2; //cualquiera
payload.confirmed = false;
payload.schedule = 'replace';

var pad = '0000';
```

```

var tipoEXT = parseInt(msg.___atributosCompartidos.EXT.EXT);

switch (tipoEXT) {
  case 0:
    payload.payload_raw = "A200";
    break;
  case 1:
    payload.payload_raw = "A201";
    break;
  case 4:
    var flanco = parseInt(msg.___atributosCompartidos
      .EXT.EXT_INT_FLANCO);
    payload.payload_raw = "A2040" + String(flanco);
    break;
  case 5:
    payload.payload_raw = "A205";
    break;
  case 6:
    var timeout = parseInt(msg.___atributosCompartidos
      .EXT_ADC_TIMEOUT).toString(16);
    payload.payload_raw = "A206";
    payload.payload_raw += pad.substring(0, pad.length -
      timeout.length) + timeout;
    break;
  case 7:
    switch (msg.___atributosCompartidos.EXT
      .EXT_CNT_FLANCO) {

      case "0":
        payload.payload_raw = "A20700";
        break;
      case "1":
        payload.payload_raw = "A20701";
        break;
      case "2":
        var contadorInicial = parseInt(msg
          .___atributosCompartidos.EXT
          .EXT_CNT_SET
        ).toString(16);
        payload.payload_raw = "A20702";
        payload.payload_raw += pad.substring(0, pad
          .length -
          contadorInicial.length) +
          contadorInicial;

        break;
    }
    break;
}

msg = {
  "payload": JSON.stringify(payload),
  "uuid": msg.uuid
};
return {
  msg: msg,
  metadata: metadata,
  msgType: msgType
};

```

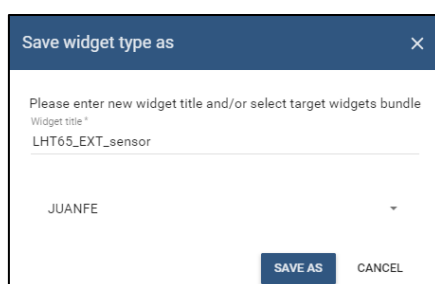
3.8. Introducción a la modificación y creación de widgets: ThingsBoard incluye una colección de widgets variada, pero inevitablemente habrá situaciones en las que tengamos que crear widgets personalizados. La información para desarrollar nuevos widgets está disponible en <https://thingsboard.io/docs/user-guide/contribution/widgets-development-before-3.0>. Esta labor requiere conocimientos sobre HTML, JavaScript (AngularJS) y CSS. Una buena forma de introducirse a la creación de widgets es partir del widget Cards > HTML Value Card, en lugar de empezar desde



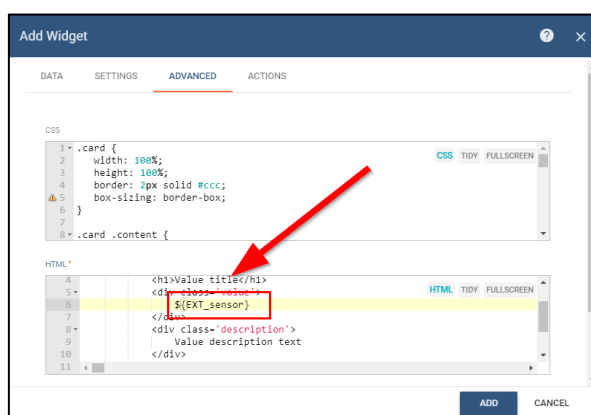
cero. Lo interesante de este widget es que funciona como un lienzo en blanco, en el que podemos configurar mediante HTML/CSS lo que queremos mostrar. Como ejemplo vamos a crear un widget para nuestro tipo de dispositivo LHT65 en el que se muestre el tipo de sensor externo conectado, y el color de fondo del widget cambie en función de este valor.

3.8.1. Crear un nuevo paquete de widgets con su nombre.

3.8.2. Copiar (Save As, en la esquina superior derecha) el widget Cards > HTML Value Card en el paquete [su nombre] (en myIoT disponemos de una colección de widgets llamada IoTTopenteEch en la que se encuentran todos los widgets personalizados que hemos creado).



3.8.3. Insertar este nuevo widget en el panel Dispositivo\_LHT65, elegir como origen de datos Dispositivo > EXT\_sensor y, en el panel Advanced, dentro de la sección HTML, sustituir la etiqueta predeterminada (My\_value) por EXT\_sensor. En este caso estamos usando como etiqueta el propio nombre del origen de datos; ante esta situación ThingsBoard sustituirá la etiqueta por el valor del origen de datos. También eliminar la división (div) de la descripción y la imagen. El código HTML debe quedar como se muestra a continuación.



```
<div class='card'>
  <div class='content'>
    <div class='column'>
      <h1>Tipo de sensor externo</h1>
      <div class='value'>
        ${Ext_sensor}
      </div>
    </div>
  </div>
</div>
</div>
```

### 3.8.4. En el código JavaScript del widget, dentro de la función updateHTML insertar el código resaltado a continuación:

```
function updateHtml() {
    var $injector = self.ctx.$scope.$injector;
    var utils = $injector.get('utils');
    var types = $injector.get('types');
    var text = self.ctx.html;
    var updated = false;
    for (var v in self.ctx.replaceInfo.variables) {
        var variableInfo = self.ctx.replaceInfo.variables[v];
        var txtVal = '';
        if (variableInfo.dataKeyIndex > -1) {
            var varData = self.ctx.data[variableInfo.dataKeyIndex].data;
            if (varData.length > 0) {
                var val = varData[varData.length - 1][1];
                if (isNumber(val)) {
                    txtVal = padValue(val, variableInfo.valDec, 0);
                } else {
                    txtVal = val;
                    //Aquí trato el caso del color de fondo
                    switch (val) {
                        case "No external sensor":
                            self.ctx.$containerParent[0].parentElement.parentElement.style.backgroundColor =
'white';
                            break;
                        case "Interrupt Sensor send":
                            self.ctx.$containerParent[0].parentElement.parentElement.style.backgroundColor
='#FF9AA2';
                            break;
                        case "Illumination Sensor":
                            self.ctx.$containerParent[0].parentElement.parentElement.style.backgroundColor
='#FFDAC1';
                            break;
                        case "ADC Sensor":
                            self.ctx.$containerParent[0].parentElement.parentElement.style.backgroundColor
='#E2F0CB';
                            break;
                        case "Interrupt Sensor count":
                            self.ctx.$containerParent[0].parentElement.parentElement.style.backgroundColor
='#B5EAD7';
                            break;
                        case "Temperature Sensor":
                            self.ctx.$containerParent[0].parentElement.parentElement.style.backgroundColor
='#C7CEEA';
                            break;
                    }
                }
            }
        }
        } else if (variableInfo.isEntityName) {
            if (self.ctx.defaultSubscription.datasources
```

### 3.8.5. Comprobar que el color de fondo del widget cambia simulando el envío desde TTN de una carga de pago de tipo sensor de temperatura (CBE508FD01BB01021F7FFF) y otra de tipo sensor de interrupción (CB040B55025A0401007FFF).

- 3.9. Actualización de un tipo de dispositivo: En myIoT puede haber tipos de dispositivos públicos o privados. Los públicos son los que están disponibles para toda la comunidad, y los privados son los que desarrolla una entidad (empresa, organismo público, particular...) para que sólo puedan usarlos algunos customers (generalmente será el caso de un integrador que quiera ofrecer estos tipos de dispositivos privados a sus clientes). Cualquier usuario de myIoT puede crear un dispositivo de tipo público. Por el contrario, los dispositivos de tipo privado sólo se pueden adquirir por reclamación; en otras palabras: el integrador tendrá que preaprovisionar los dispositivos en myIoT (contactando con la autoridad) y sus clientes los reclamarán.