ANEXO 1: GUÍA DE IMPLEMENTACION EN SOFIA2

En esta guía se detallará todos los pasos para simular y enviar datos de sensores desde una Raspberry Pi hasta el servicio Sofia2. La Raspberry Pi simulará cinco variables correspondientes a un panel solar:

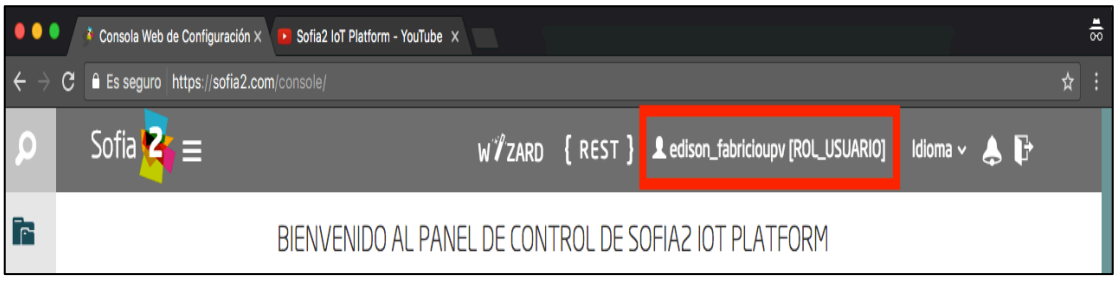
- Voltaje STC
- Voltaje NOCT
- Intensidad STC
- Intensidad NOCT
- Temperatura Panel.

La Raspberry Pi usada tiene las siguientes características:

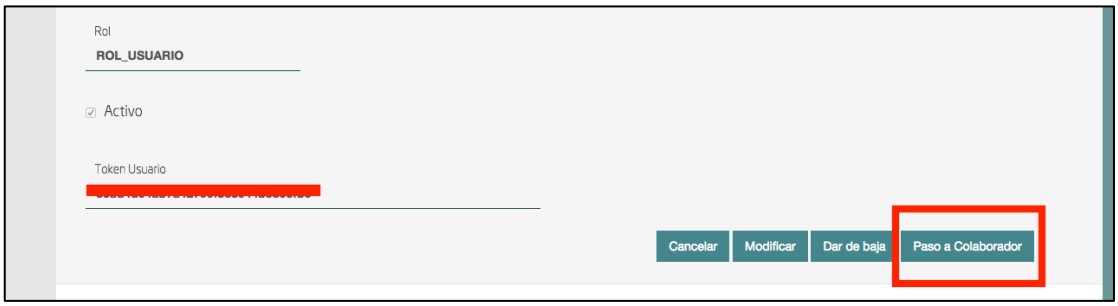
- Raspberry Pi 3
- Raspbian Jessie 8.0
- Linux 4.4.38-v7+
- Python 2.7

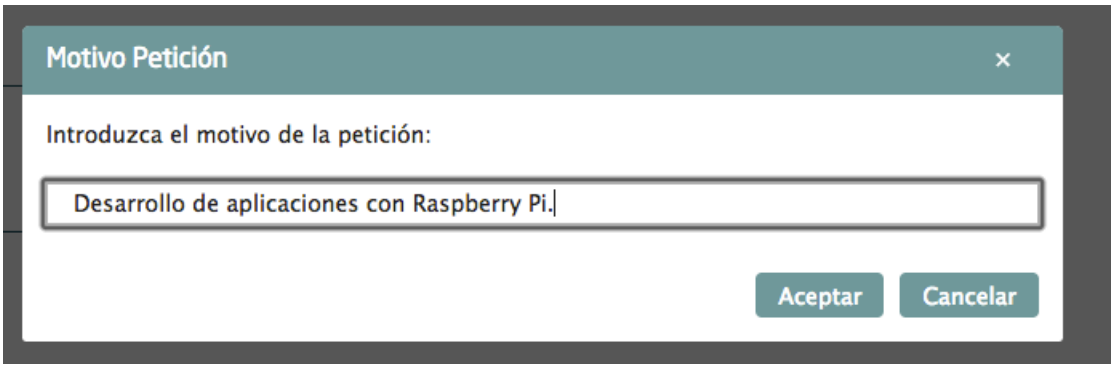
Lo primero que debemos hacer es crear una cuenta en Sofia2 en el siguiente link:
https://sofia2.com/console/login?lang=es

Una vez finalizado el registro debemos solicitar pasar de ***ROL_USUARIO*** a ***ROL_COLABORADOR***. En la pagina principal de Sofia2 damos clic en el texto que muestra nuestro ***ROL_USUARIO***.



Se abrirá una pagina con la información de nuestro usuario. Damos clic en ***Paso a Colaborador*** y rellenamos el cuadro de dialogo que se abre con el motivo de nuestra petición.





Enviamos la petición y esperamos la confirmación de paso a ***ROL COLABORADOR*** en nuestro correo electrónico registrado. Este proceso tarda de dos a cuatro días.

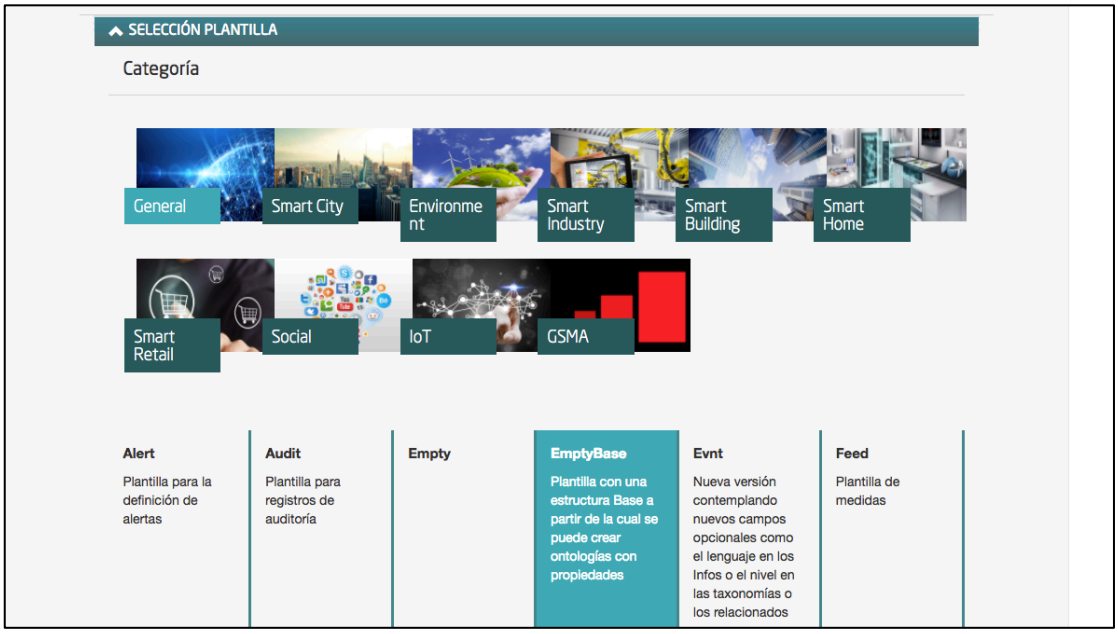
Teniendo ya nuestro ***ROL COLABORADOR*** debemos crear una ontología, la misma que representará el objeto Panel Solar. Este objeto tendrá como propiedades las cinco variables mencionadas en un principio. En el menú izquierdo vamos a ***ONTOLOGIAS >> Crear Ontología***

Elegimos la opción ***Creación Paso a Paso***.

Introducimos la información correspondiente:

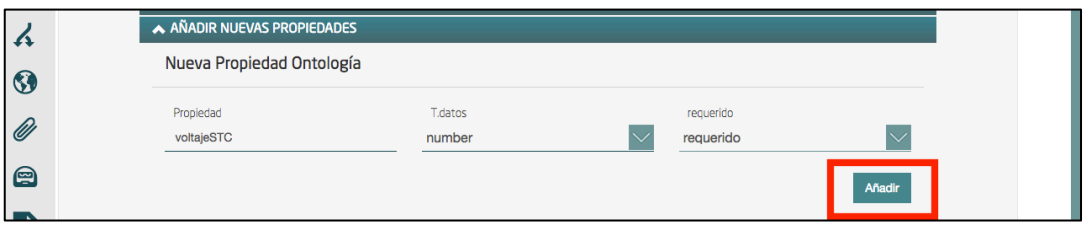
Nombre: Panelsolar021UPV
Versión de Plantilla Actual: Pùblica
Descripción: Panel Solar de 250W. Código: 021
Meta-Inf: panel021

En ***Configuración de Esquema*** elegimos ***Selección Plantilla >> General >> EmptyBase***



En la pestaña ***Añadir Nuevas Propiedades*** introducimos la siguiente información:

Propiedad: voltajeSTC
Tipo de datos: number
Requerido: requerido



Damos clic en ***Añadir***. Hacemos los mismo para las otras variables:

Propiedad: voltajeNOCT
Tipo de datos: number
Requerido: requerido

Propiedad: intensidadSTC
Tipo de datos: number
Requerido: requerido

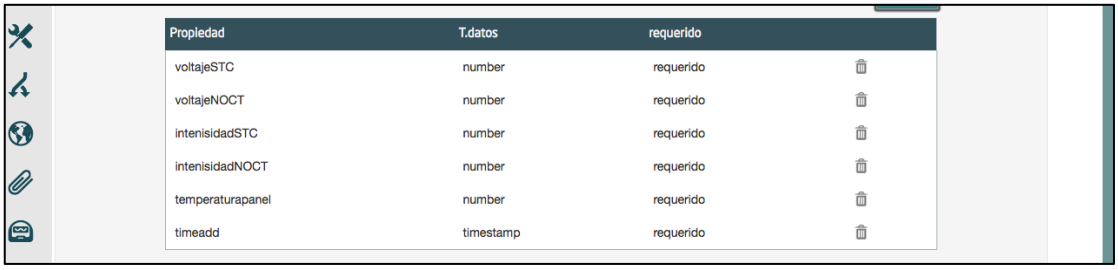
Propiedad: intensidadNOCT
Tipo de datos: number
Requerido: requerido

Propiedad: temperaturapanel
Tipo de datos: number
Requerido: requerido

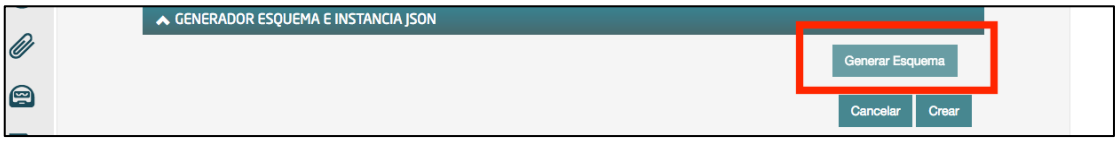
Además Sofia2 necesita saber el tiempo exacto en el cual la variable fue "sensada", entonces añadimos una propiedad más:

Propiedad: timeadd
Tipo de datos: timestamp
Requerido: requerido

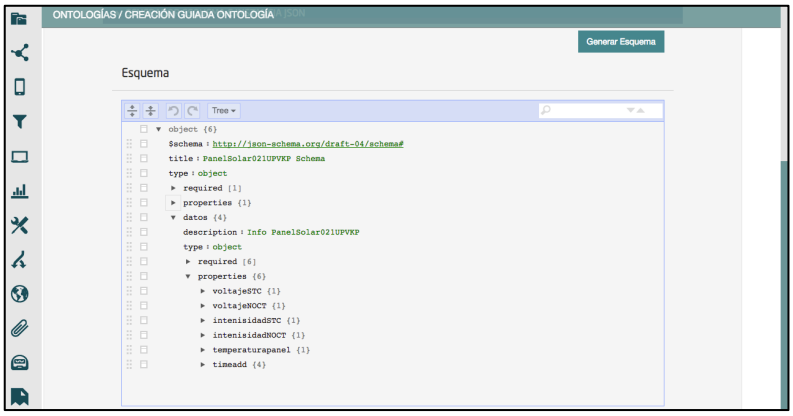
A la final debería quedar algo similar a esto.



Ahora en ***Generador Esquema e Instancia JSON*** damos clic en ***Generar Esquema***

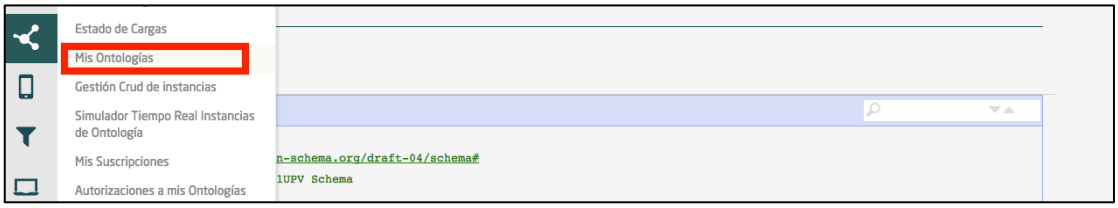


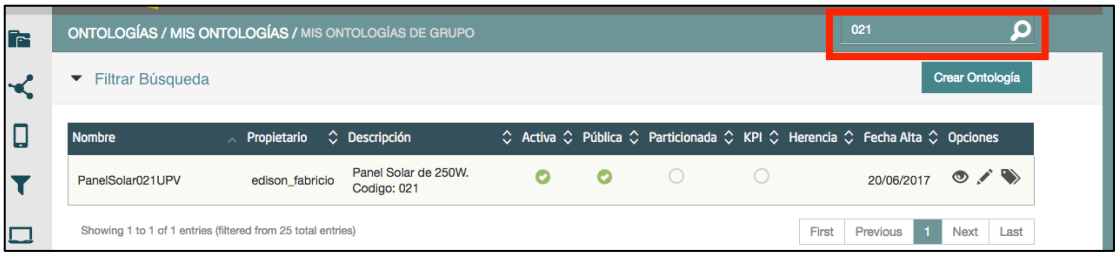
Debería generarse un esquema similar a este:



Damos clic en ***Generar Instancia*** y después en ***Crear***:

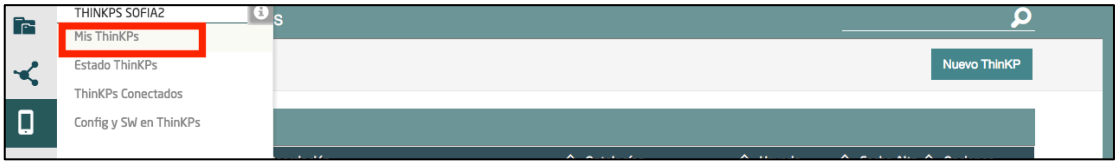
Con esto tenemos lista la ontología que representa al objeto panel solar con cinco propiedades. Para ver nuestra ontología vamos a ***Ontologias >> Mis ontologías***. La podemos buscar más fácilmente con la ayuda del buscador de ontologías.

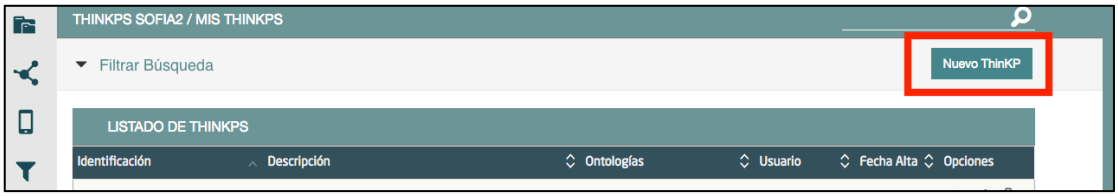




Una vez creada la ontología debemos crear una KP que consumirá la información generada por la ontología (la información de los sensores). Esto permitirá poder generar una instancia del KP que podremos llamar más tarde para consultar la información de la ontología/objeto.

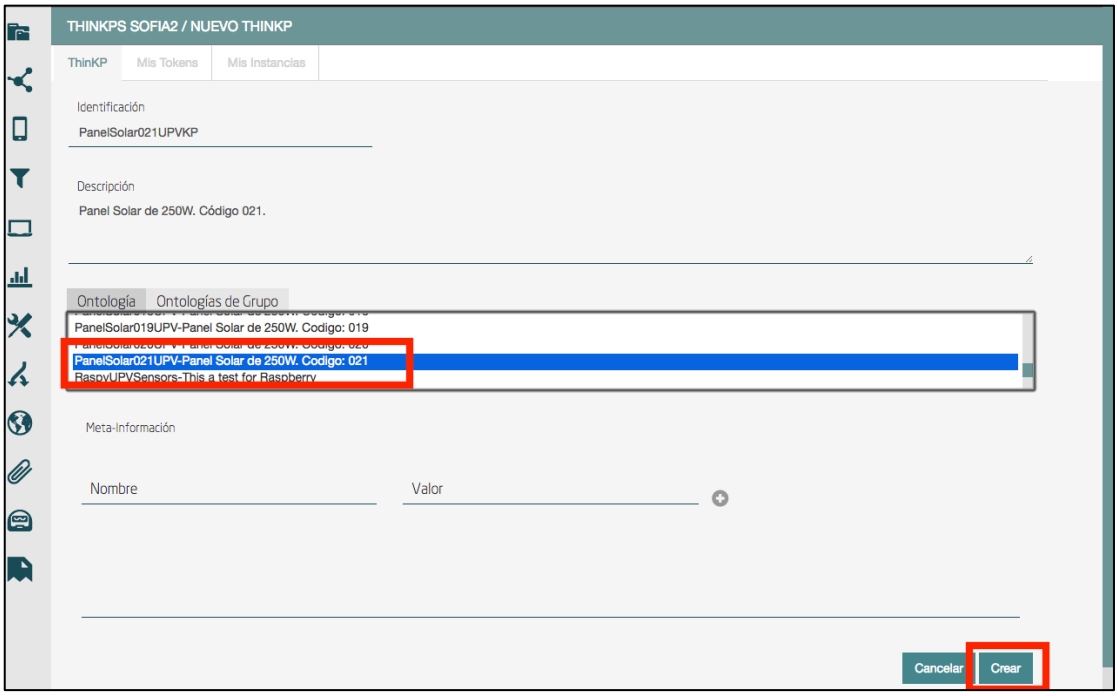
Vamos a ***THINKPS SOFIA2 >> Mis ThinKPs*** y damos clic en ***Nuevo***



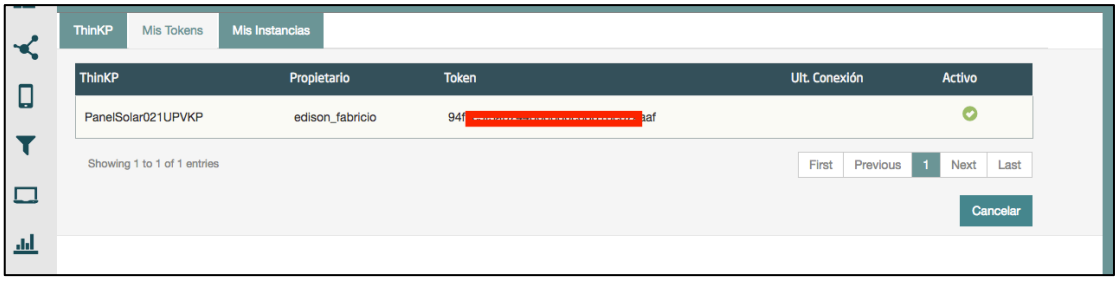


Introducimos la siguiente información y le damos clic en ***Crear***.

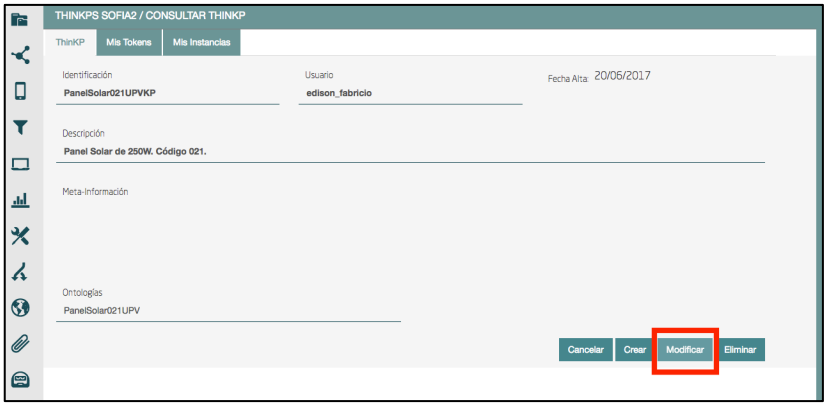
Identificación: PanelSolar021UPVKP
Descripción: Panel Solar de 250W. Código 021.
Ontología: PanelSolar021UPV-Panel Solar de 250W. Código 021



Una vez creado vamos la pestaña ***Mis Tokens*** donde estará el token necesario para desde la Raspberry Pi poder realizar la autenticación de la ontología.

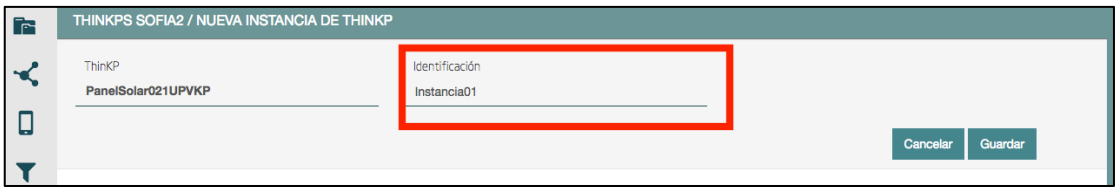


Ahora en la pestaña ***ThinKP*** le damos clic en ***Modificar***



En la pestaña ***Mis Instancias*** le damos al botón ***Nueva Instancia de ThinKP***

En ***Identificación*** ponemos ***Instancia01*** y guardamos los cambios.



Con esto tenemos todo listo para que la Raspberry Pi genere los datos. En la terminal de nuestra Raspberry Pi creamos un directorio donde descargaremos el API de Python para Sofia2.

```
$ mkdir sofia2-api
$ cd sofia2-api/
```

Una vez dentro del directorio creado descargamos el API con la siguiente orden:

```
$ git clone https://github.com/Sofia2/python-api.git
```

Después ingresamos a la carpeta que se ha descargado e instalamos las herramientas necesarias.

```
$ cd python-api/
$ sudo pip install -r requirements.txt
$ sudo ython setup.py install
```

Terminado ésto podemos descargar el código ejemplo para Sofia2.

```
$ git clone https://github.com/edisonupv/iot-servicesexamples.git
```

De los ficheros descargados el que nos interesa es el llamado ***panelSofia2.py*** el cual en su interior esta todo lo necesario para que la Raspberry Pi aleatoriamente genere los valores de las variables del panel solar, los adecue en cadena de texto en formato JSON y las envíe a Sofia2. Lo único a modificar es lo pertinente al nombre de la ontología, el token del KP y la instancia del mismo.

```
19
20      ONTOLOGY = "******************"              #Name of ontology that you were created before
21      TOKEN = "**********************************"  #Token of ontology
22      INSTANCE = "********************"             #Instance that you created for the KP that will consume the information
23                                                    #Example INSTANCE = "PanelSolar021UPVKP:Instancia01"
```

Hecho los cambios ejecutamos el programa.

$ python panel panelSofia2.py

El cual si todo sale bien debe arrojar un mensaje similar a este:

```
30.47
28.33
10.33
9.01
46.22
2017-06-20T19:12:18.000Z
INSERT INTO PanelSolar021UPV(voltajeSTC, voltajeNOCT, intensidadSTC, intensidadNOCT, temperaturapanel, timeadd) values (30.47,28.33,10.33,9.01,46.22, "{ '$date': '2017-06-20T19:12:18.000Z'}")
2017-06-20 19:12:18,700 WebsocketBasedSSAPEndpoint [INFO] - Waiting for the websocket connection to be established
2017-06-20 19:12:18,701 _SSAPWebsocketClient [INFO] - Websocket connection established
2017-06-20 19:12:18,908 WebsocketBasedSSAPEndpoint [DEBUG] - Data received:
2017-06-20 19:12:18,909 WebsocketBasedSSAPEndpoint [DEBUG] - {"body":{"data":"56ec8ce2-487c-41b3-ab3c-5771bc328147","ok":true,"error":null,"errorCode":null,"contentType":null},"messageId":"e28abe5f-309d-4e80-a49b-4bb03c544a02","sessionKey":"56ec8ce2-487c-41b3-ab3c-5771bc328147","ontology":"","direction":"RESPONSE","messageType":"JOIN"}
2017-06-20 19:12:19,861 WebsocketBasedSSAPEndpoint [DEBUG] - Data received:
2017-06-20 19:12:19,862 WebsocketBasedSSAPEndpoint [DEBUG] - {"body":{"data":"{\"_id\":ObjectId(\"59495772e4b0f660d2731473\")}","ok":true,"error":null,"errorCode":null,"contentType":null},"messageId":"07e10363-25a3-4576-afc9-b3ba9a5d5412","sessionKey":"56ec8ce2-487c-41b3-ab3c-5771bc328147","ontology":"PanelSolar021UPV","direction":"RESPONSE","messageType":"INSERT"}
A(n) INSERT message was received!
{
    "body":{
        "contentType":null,
        "data":{
            "_id":"ObjectId('59495772e4b0f660d2731473')"
        },
        "error":null,
        "errorCode":null,
        "ok":true
    },
    "direction":1,
    "messageId":"07e10363-25a3-4576-afc9-b3ba9a5d5412",
    "messageType":2,
    "ontology":"PanelSolar021UPV",
    "sessionKey":"56ec8ce2-487c-41b3-ab3c-5771bc328147"
}
.
----------------------------------------------------------------------
Ran 1 test in 2.124s

OK
```

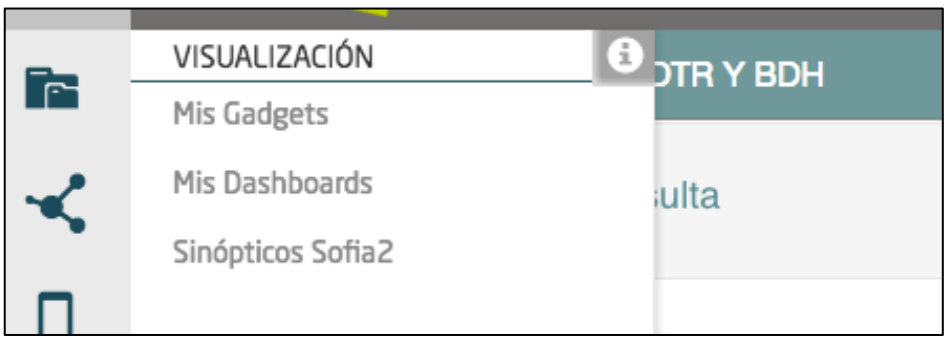
Ahora procedemos a ejecutar el script automáticamente con la herramienta crontab cada quince minutos. Para ello en la terminal ingresamos:

$ crontab –e

Ingresamos al final del archivo la siguiente línea

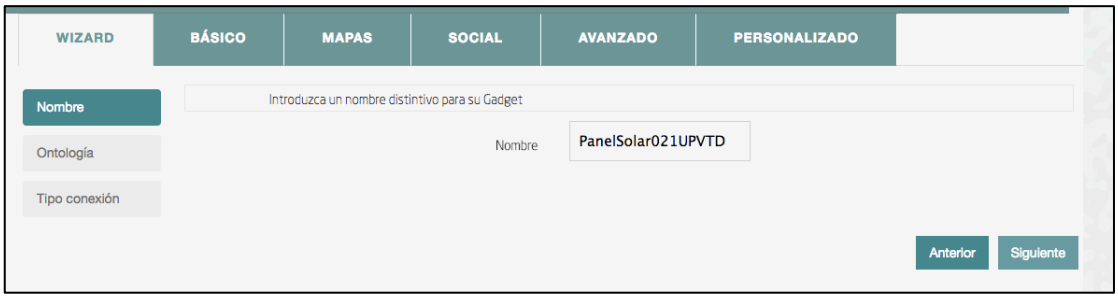
*/15 * * * * python /***rutadetucarchivo***/panelSofia2.py

Donde ***rutadetuarchivo*** es la ruta del script python en tu directorio Linux. Ahora con la Raspberry Pi enviando los datos cada quince minutos podemos ya generar las gráficas que representaran las variables. Por ello vamos a ***Visualización >> Mis Gadgets***

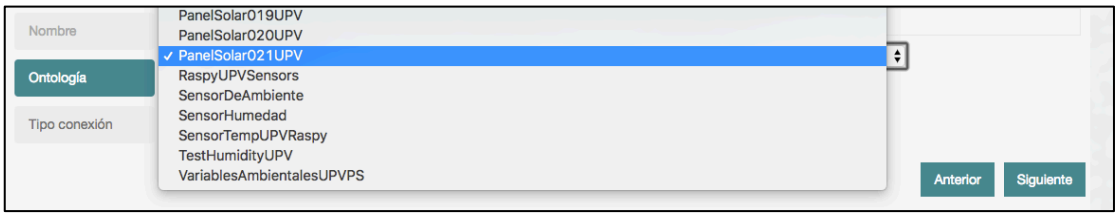


Damos clic en el botón ***Crear Gadget***.

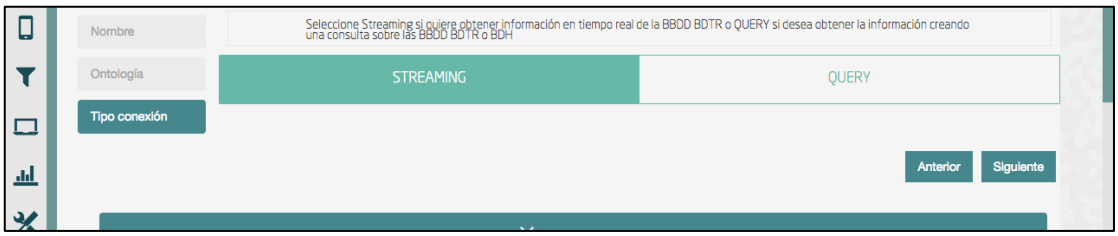
En la nueva ventana elegimos la pestaña ***Wizard*** y de ahí en el campo nombre ponemos: ***PanelSolar021UPVTD*** y le damos clic en ***Siguiente***



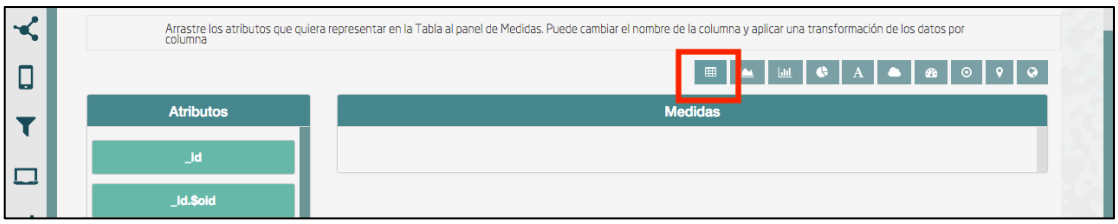
Aquí elegimos la ontología que deseamos graficar, en este caso la ontología ***PanelSolar021UPV*** y después clic en ***Siguiente***.

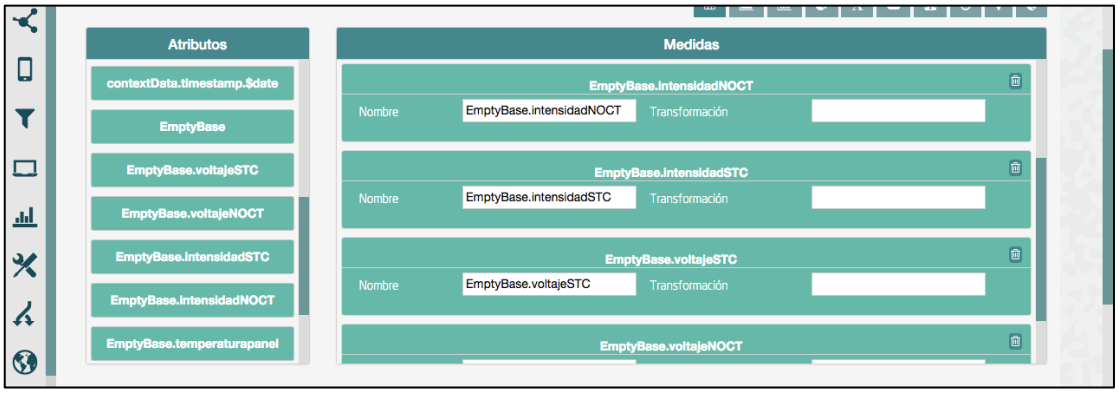


Elegimos la opción ***Streaming*** y después clic en ***Siguiente***.

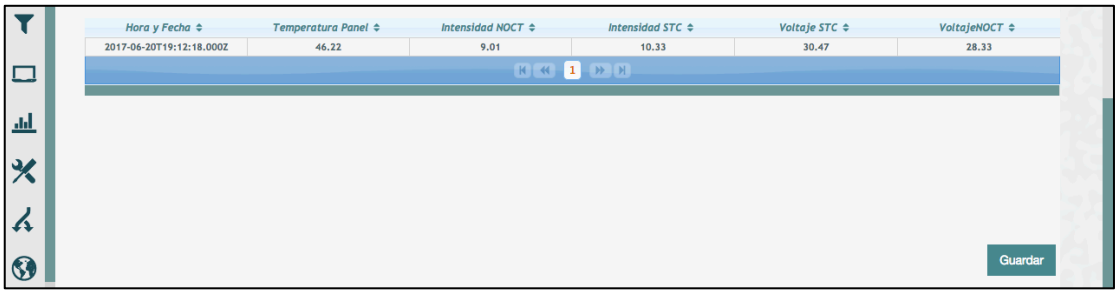


Aquí elegimos la opción de ***Tabla*** y después arrastramos los atributos que queremos representar en al panel de Medidas. Puedes cambiar el nombre de la columna a uno mas conveniente a tu proyecto.

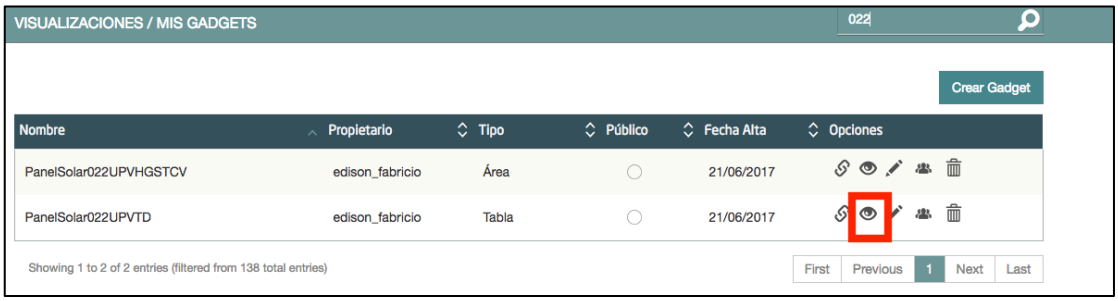




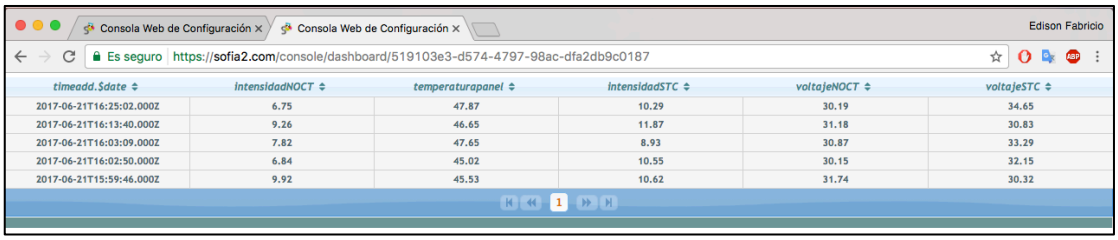
Después le damos clic en ***Guardar***



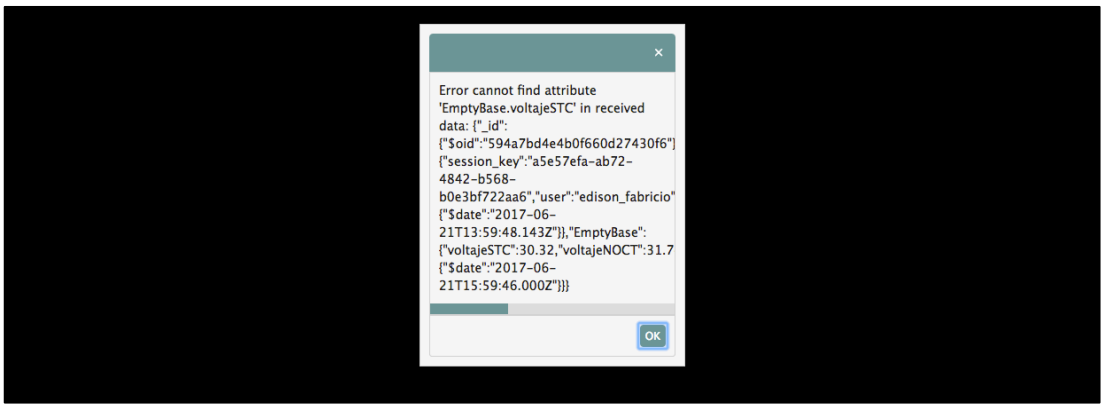
Para visualizar la tabla debemos en el apartado de ***Mis Gadgets*** buscar el gadget que hemos creado y dar clic en ***Ver Gadget***



Deberíamos ver algo similar a esto.

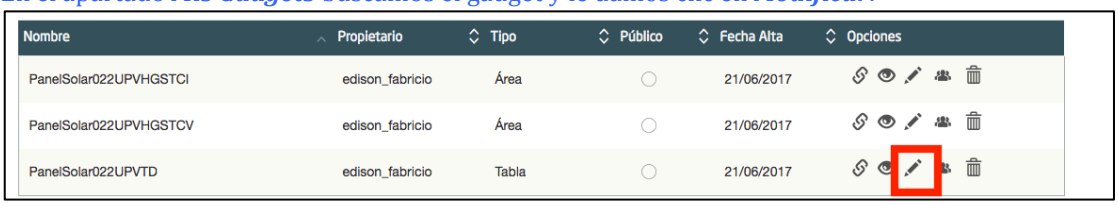


NOTA:
Si tenemos un error similar a la figura de abajo se debe a un bug de Sofia2 al momento de implementar la ontología en su base de datos.

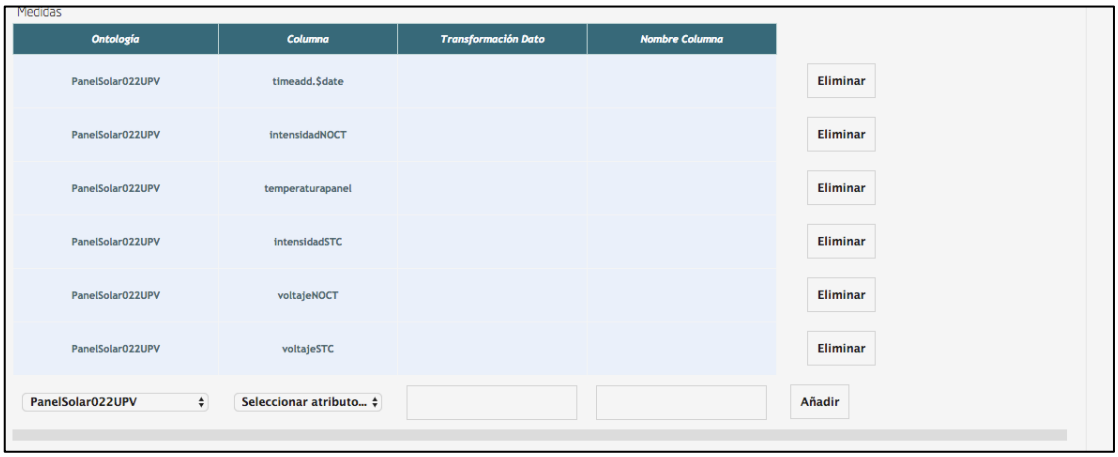


Tendremos que hacer lo siguiente:

En el apartado ***Mis Gadgets*** buscamos el gadget y le damos clic en ***Modificar***.

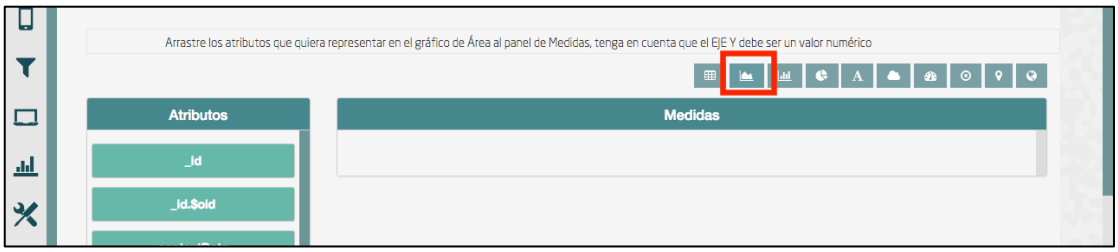


Después tendremos que asegurarnos que las medidas tengan este formato

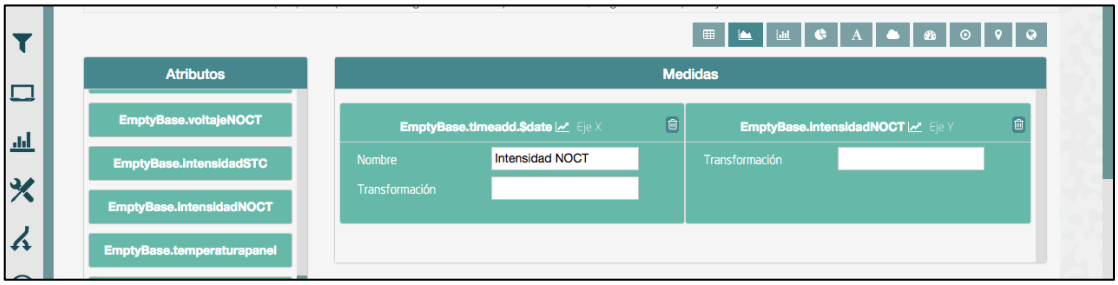


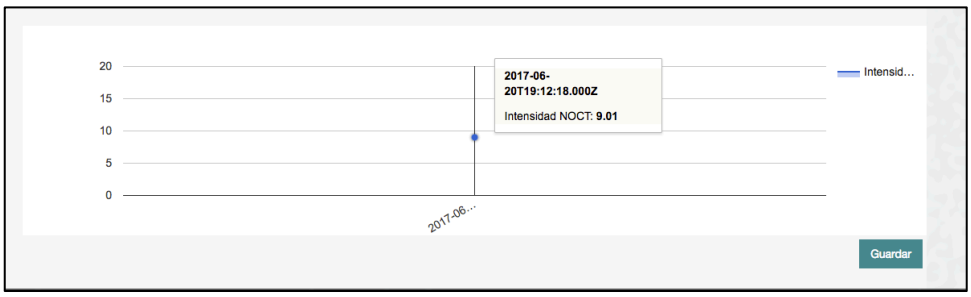
Si no es así las eliminamos y luego las añadimos con el formato correcto.

Ahora vamos a crear una grafica tipo histograma. Para ello creamos una nuevo gadget y le ponemos de nombre ***PanelSolar021UPVHGSTCV***, elegimos la ontología PanelSolar021UPV y de tipo ***Streaming***. Y en el tipo de grafico elegimos la opción ***Area***.



Aquí arrastramos los atributos a graficar, en este caso los atributos de ***intensidadNOCT*** y ***timeadd***. Damos clic en Guardar.



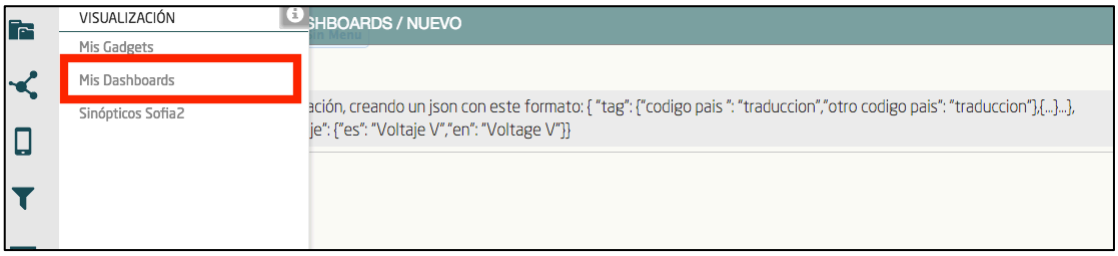


De igual manera que el primer gadget podemos observar este histograma con la opción ***Ver Gadget.***

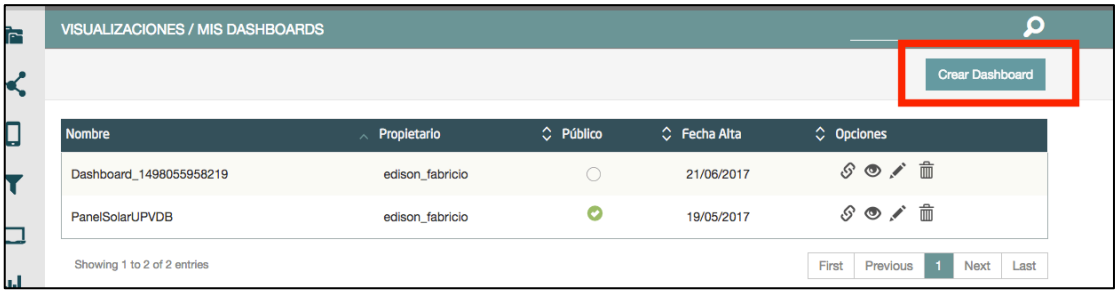
NOTA: Si ocurre el mismo error del primer gadget, seguir los mismos pasos y asegurarse que los campos tengan el formato correcto.

Realizar los mismos pasos para crear las demos histogramas para las demás variables. Recomendamos probar otras graficas que estén disponibles en Sofia2.

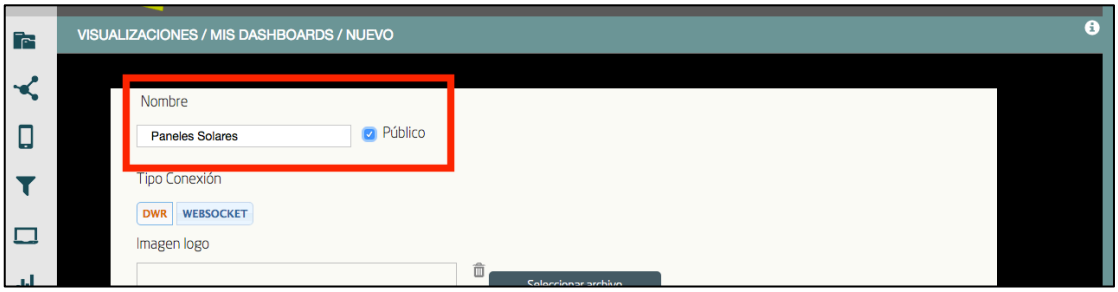
Es posible también construir un DashBoard donde reunir los diferentes tipos de gadgets que hemos creado. Para ello vamos a ***Visualización >> Mis Dashboards***



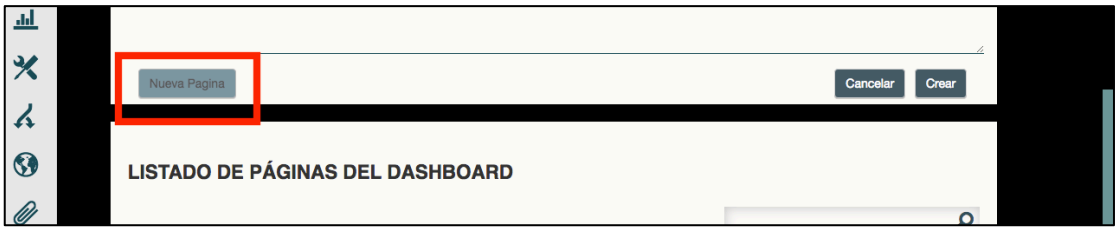
Y le damos clic en ***Crear Dashboard***



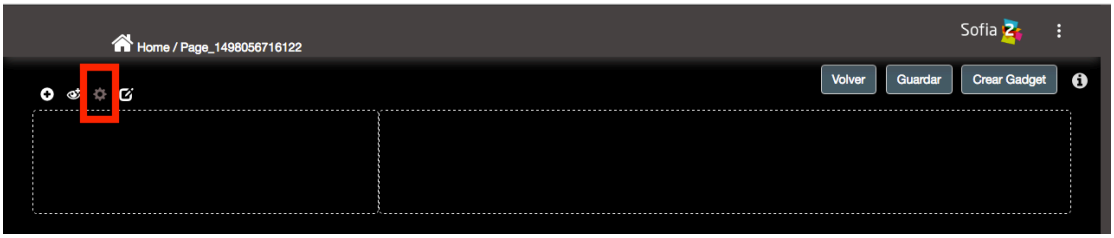
Ingresamos el nombre y lo hacemos publico.



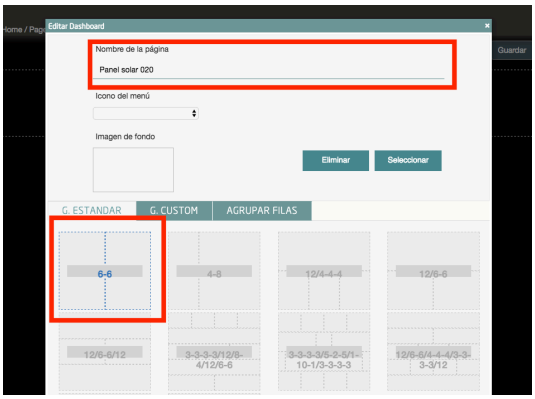
En la parte inferior damos clic en ***Nueva Pagina***



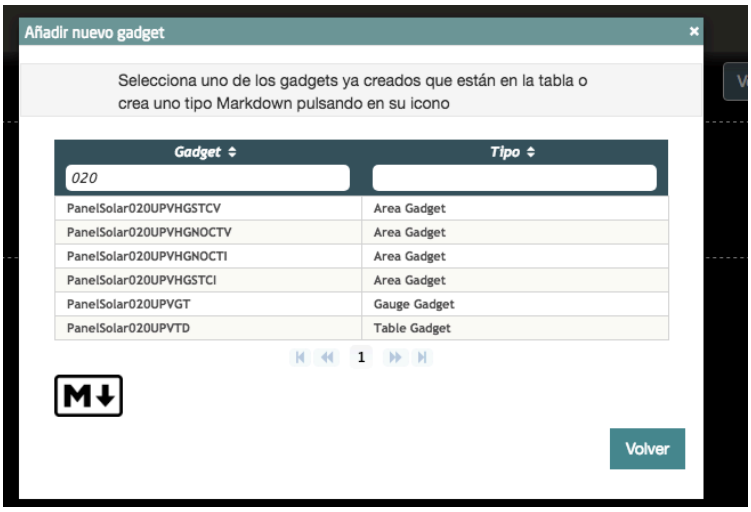
Se nos abrirá una nueva pagina. Aquí dar clic en la opción ***Configurar Pagina***



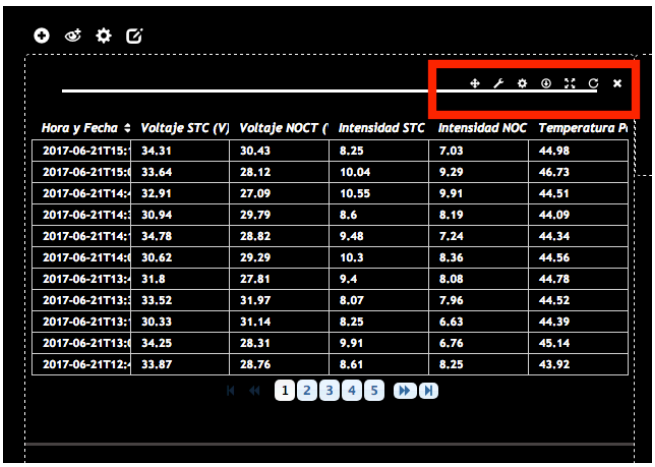
Después ingresar un nombre y elegir la opción de formato de la pagina de 6-6



Después le damos clic en ***Volver***. Ahora elegimos la opción ***Añadir nuevo gadget***. Se nos abrirá un ventana con los gadgets que hemos creado y haciendo clic sobre ellos se irán añadiendo.

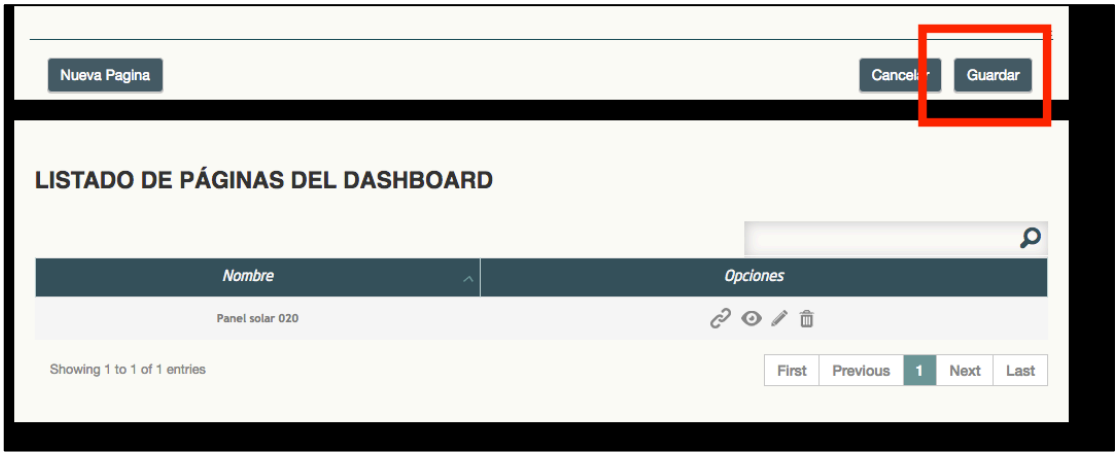


Cada gadget tiene un pequeño menú con varias opciones, por ejemplo la de poder mover el gadget para una mejor organización.

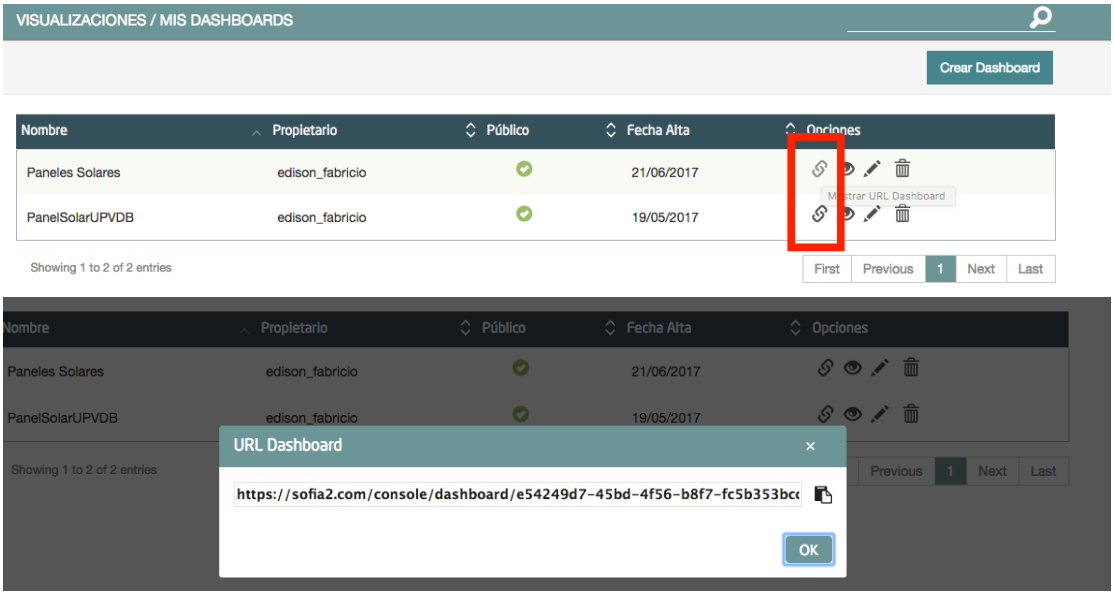


Cuando tengas todo listo le das clic en ***Guardar*** y después en ***Volver.***

Ahora le das clic en nuevamente en ***Guardar***.



El dashboard puede generar un link publico donde alguien externo a Sofia pueda observar el mismo. Para ello vamos a ***Visualización >> Mis Dashboards***, buscamos nuestro dashboard y le damos clic en ***Mostrar URL Dashboard***



Estos mismo procedimiento se puede aplicar para las demás ontologías con las mismas variables u otras diferentes. Así como también los gadgets que representen los valores de las variables de las ontologías.

En este canal de Youtube se puede encontrar varios tutoriales de cómo usar Sofia2: https://www.youtube.com/channel/UC6VGV_lN9gB2mJcPdIYHB0A

En la siguiente pagina se puede encontrar información de los diferentes API de Sofia para las diferentes plataformas: https://github.com/Sofia2

ANEXO 2: GUÍA DE IMPLEMENTACION EN FIWARE

En esta guía se detallará todos los pasos para simular y enviar datos de sensores desde una Raspberry Pi hasta el servicio Fiware. La Raspberry Pi simulará cinco variables correspondientes a un panel solar:

- Voltaje STC
- Voltaje NOCT
- Intensidad STC
- Intensidad NOCT
- Temperatura Panel.

La Raspberry Pi usada tiene las siguientes características:

- Raspberry Pi 3
- Raspbian Jessie 8.0
- Linux 4.4.38-v7+
- Python 2.7

Lo primero que debemos hacer es crear una cuenta en la plataforma Fiware-Lab en el siguiente link: https://account.lab.fiware.org. Una vez tengamos la cuenta descargamos la herramienta FIGWAY en nuestra Raspberry e ingresamos a la carpeta descargada:

```
$ git clone https://github.com/telefonicaid/fiware-figway.git
$ cd fiware-figway/
$ cd python-IDAS4/
```

Luego ejecutamos el script ***get_token.py*** e ingresamos nuestra cuenta y contraseña registrada anteriormente para poder obtener un token que servirá para autenticarnos en el Context Broker e IDAS que proporciona Fiware. Este token lo guardamos ya que más adelante nos será útil.

```
$ python get_token.py
```

Ahora es necesario configurar el archivo config.ini ya que aquí se almacena la información referente a nuestro usuario, token, dirección IP del context Broker, dirección IP de IDAS, servicios a usar e información del dispositivo registrado. Para ello introducimos la orden:

```
$ nano config.ini
```

Una vez abierto el fichero introducimos la siguiente información:

```
[user]
username=edison***************com                #Tu username registrado en Fiware
token=85*******************************aa          #Tu Token obtenido con el scritp get_token.py
[contextbroker]
host=130.206.80.40
port=1026
OAuth=no
fiware_service=bus_auto
fiware-service-path=/
[idas]
host=130.206.80.43
adminport=4061
ul20port=7896
OAuth=no
fiware-service=OpenIoT
fiware-service-path=/
apikey=4jggokgpepnvsb2uv4s40d59ov
[local]
host_type=RaspberryPiupv
host_id=************************                   #Username o MAC de su Raspberry Pi
```

Guardamos y cerramos los cambios. Podemos realizar una comprobación podemos ejecutar el script ***CheckVersion.py*** y obtener un resultado similar a este.

$ cd ContextBroker/
$ python CheckVersion.py

Una vez realizada esta comprobación podemos empezar a registras nuestras entidades (en este caso los objetos Paneles Solares) y enviar los valores de sus atributos (en este caso los valores de las variables). Para ello se puede descargar un script en el repositorio https://github.com/edisonupv/iot-servicesexamples con el nombre ***panelFiware.py***. Este script contiene ya todo lo necesario para conectarse con el Context Broker y generar ya las variables aleatorias correspondiente a los valores del objeto panel solar. Este script debe ser copiado en el directorio donde estén los archivos de la herramienta FIGWAY específicamente en la ruta:

tudirectoriohome/fiware-figway/python-IDAS4/ContextBroker

Donde ***tudirectoriohome*** es la ruta donde esta descargado el FIGWAY esto para no tener problemas al momento de encontrar los API's y archivos de configuración necesarios para ejecutar el script. Dentro del script ***panelFiware.py*** tenemos que configurar lo siguiente:

```
ENTITY_ID= "panelsolarupv"                    #The ID of the object
ENTITY_TYPE="panelsolar"                      #The Type of the Object

……
……

#La ruta de tu archivo config.ini
CONFIG_FILE = "tudirectoriohome/fiware-figway/python-IDAS4/config.ini"
```

Donde ***ENTITY_ID*** y ***ENTITY_TYPE*** son los nombres que tu le asignas a tu objeto panel solar. Con esto listo es hora de ejecutar el script, para ello introducimos la orden :

$ python panelFiware.py

Debemos obtener una respuesta similar a esta

Podemos además comprobar el estado de la entidad y sus atributos usando el script ***GetEntityById.py*** seguido del ID de la entidad creada y registrada en el Context Broker, en este caso ***panelsolarupv.***

$ python GetEntityById.py panelsolarupv011

La respuesta debe mostrar cada una de los atributos generados y enviados por el script anterior. La respuesta debe ser algo similar a esto:


```
* Status Code: 200
* Response:
{
  "contextResponses" : [
    {
      "contextElement" : {
        "type" : "panelsolar",
        "isPattern" : "false",
        "id" : "panelsolarupv011",
        "attributes" : [
          {
            "name" : "intensidadNOCT",
            "type" : "float",
            "value" : "6.5"
          },
          {
            "name" : "intensidadSTC",
            "type" : "float",
            "value" : "11.5"
          },
          {
            "name" : "temperaturapanel",
            "type" : "float",
            "value" : "46.68"
          },
          {
            "name" : "voltajeNOCT",
            "type" : "float",
            "value" : "31.58"
          },
          {
            "name" : "voltajeSTC",
            "type" : "float",
            "value" : "30.53"
          }
        ]
      },
      "statusCode" : {
        "code" : "200",
        "reasonPhrase" : "OK"
      }
    }
  ]
}
```

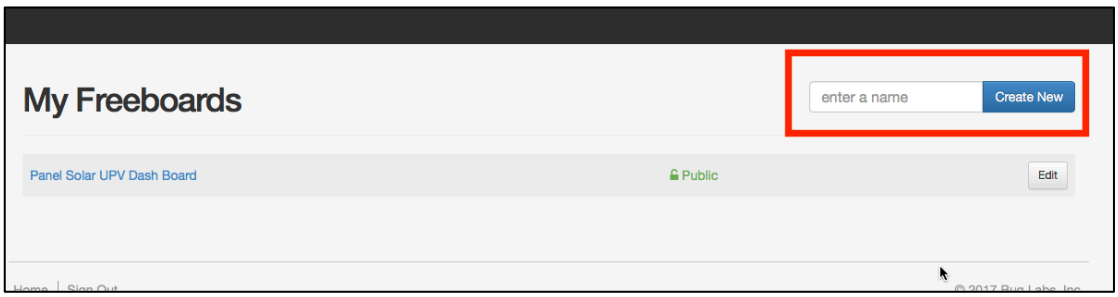

Ahora podemos comenzar a enviar periódicamente los valores de los atributos pertenecientes a la entidad ***panelsolarupv*** y que esta registrada en el Context Broker. Para este fin haremos uso de la herramienta Linux ***crontab***. Ingresamos a ella con la siguiente orden:

$ crontab –e

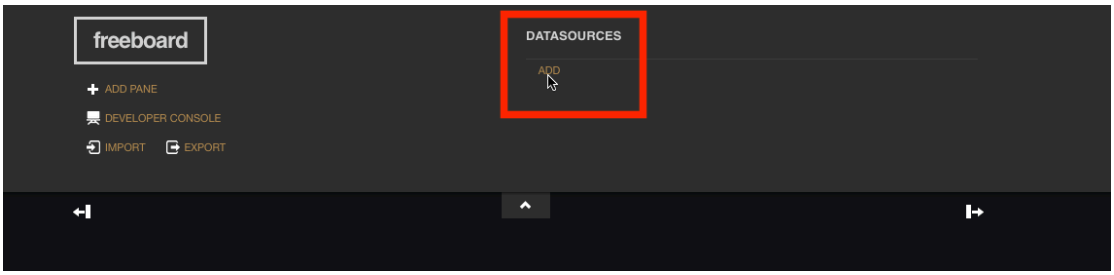
Añadimos la siguiente línea:

***/15 * * * * python tudirectoriohome/fiware-figway/python-IDAS4/ContextBroker/panelsolarupv.py**

Ahora haremos uso de la herramienta gratuita en línea FREEBOARD que permite hacer consultas a las entidades registradas en el Context Broker y visualizarlas de una maneras mas grafica y vistosa. Para ello nos registramos en el siguiente link https://freeboard.io/signup. Una vez tenemos la cuenta registrada, en la opción My Freeboards le damos a crear un nuevo DashBoard. Recuerda primero ingresar un nombre.

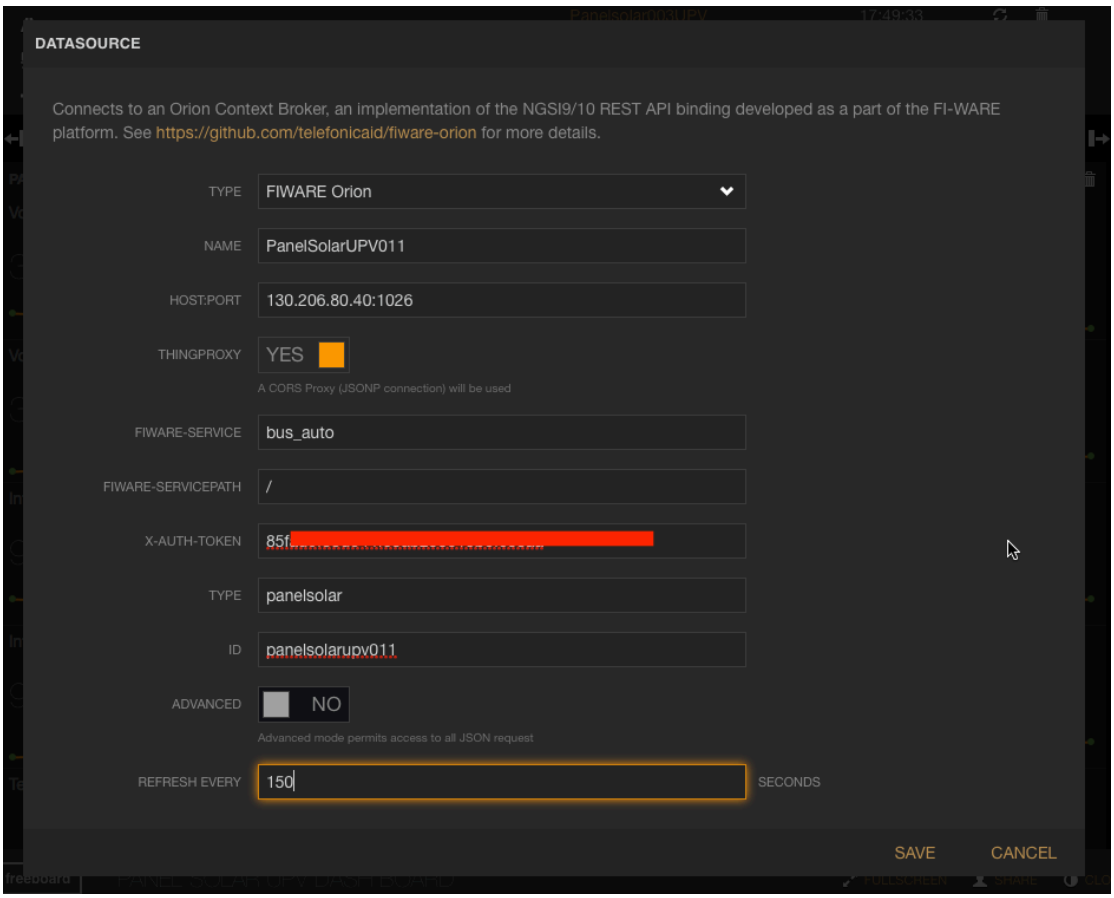


Una vez hecho lo anterior se mostrara una pagina, aquí debemos añadir el ***DataSource*** que corresponde a la entidad registrada en el ORION Context Broker.

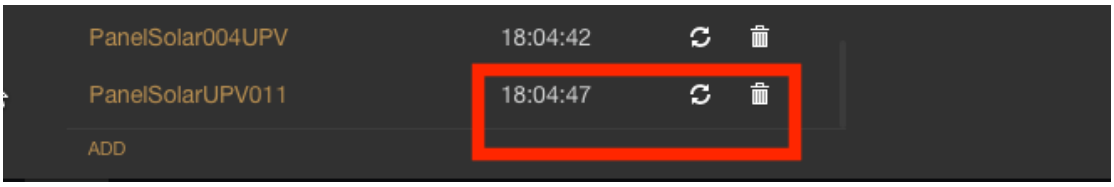


Le damos clic en ***ADD*** y en ***Type*** elegimos ***FIWARE Orion*** y rellenamos la información solicitada de la siguiente manera:

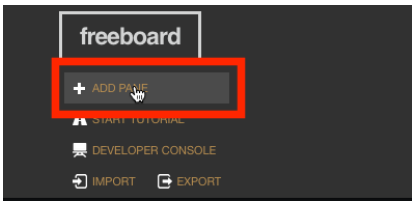
NAME: El nombre que identificara al DataSource.
HOST- PORT: Aquí configurar la dirección IP y el puerto del Orion Context Broker. En este caso la dirección *130.206.80.40:1026*
FIWARE-SERVICE: bus_auto
FIWARE-SERVICEPATH: /
X-AUTH-TOKEN: El token obtenido con el script get_token.py
TYPE: El tipo de entidad registrado en el Context Broker. En este caso es de tipo *panelsolar*
ID: Identificador de la entidad registrada en el Context Broker. En este caso es *panelsolarupv*
REFRESH EVERY: Cantidad en segundos en que se realizara la consulta al Context Broker sobre los atributos de la entidad.



Guardamos este **DataSource** y si todo esta bien tendría que mostrarse el valor del ***Last Update*** que confirma que FreeBoard puede realizar la consulta de la entidad sin inconvenientes.



Ahora vamos a graficar los atributos de la entidades asociados al ***DataSource***. Para ello vamos a ***ADD PANE*** y veremos como se añade un nuevo panel en el cual le damos clic en el signo **(+)**

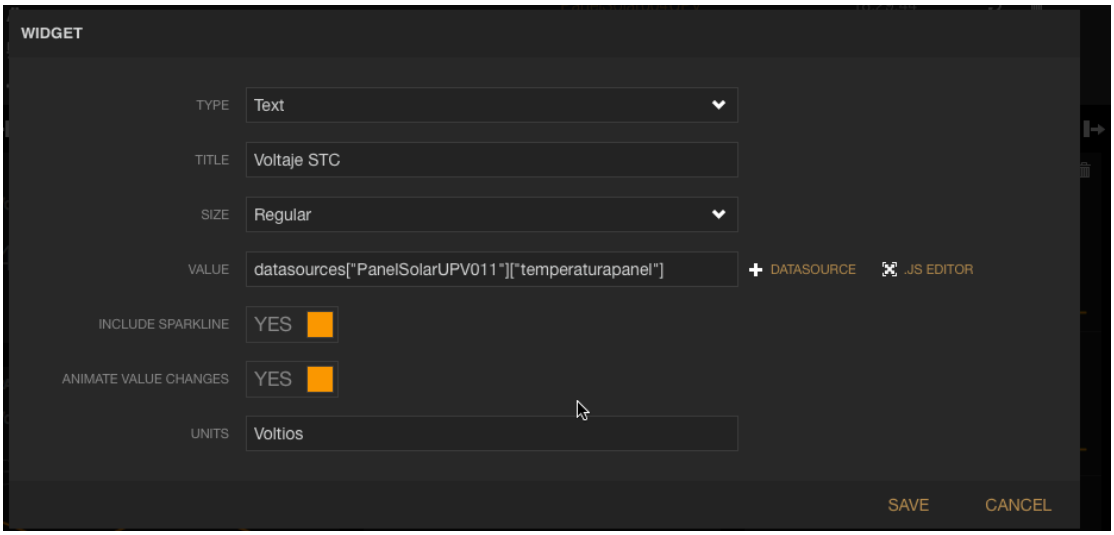


Se abrirá una nueva ventana donde podremos seleccionar el tipo de Widget que será la representación grafica del atributo de la entidad. En esta ventana ingresamos las siguientes opciones:

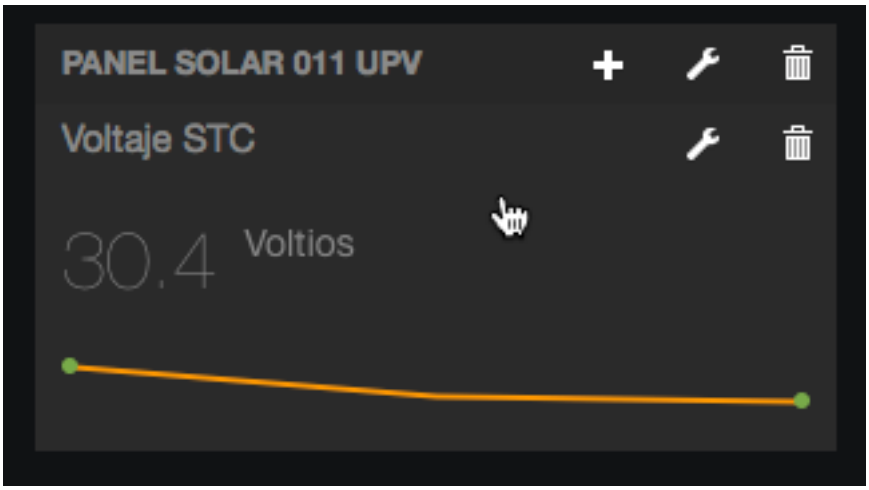
TYPE: Elegimos un widget de tipo *Text*.
TITLE: Un titulo para este widget.
SIZE: El tamaño del widget.
VALUE: Aquí debemos damos clic en el signo ***(+)*** , elegimos el *DataSource panelsolarupv* y el atributo a graficar que en esta caso seria *voltajeSTC*

INCLUDE SPARKLINES: Yes
ANIMATE VALUE CHANGES: Yes
UNITS: Las unidades de este atributo. En este caso son *Voltios*

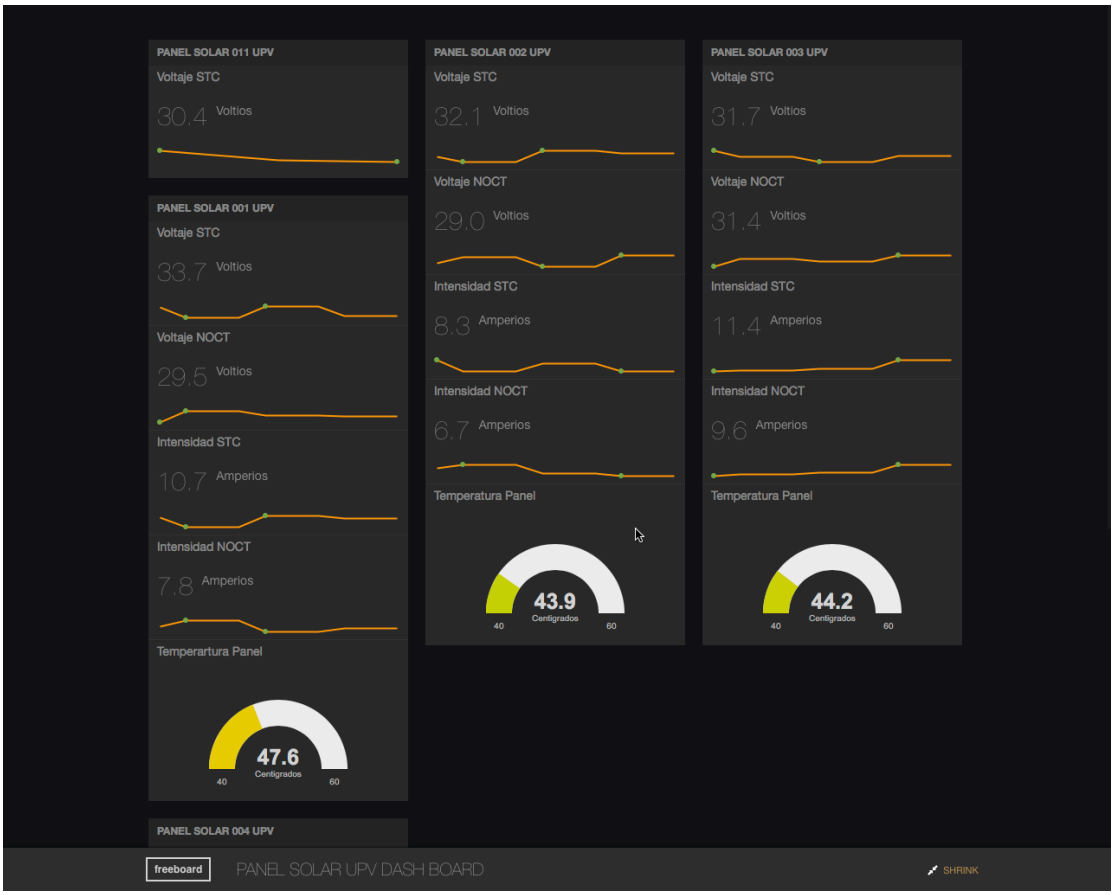
Al final debería quedar algo similar a esto.



Guardamos el Widget y deberíamos observar algo similar a esto.



Puedes probar con otro tipo de Widgets y seguir casi los mismos pasos anteriormente mencionados. Así mismo añadir más DataSources de diferentes entidades registradas en el Context Broker, éstas a su vez pueden ser creadas de la misma manera anteriormente explicada. A la final podrías tener un DashBoard similar a este



FreeBoard permite también generar un link publico donde cualquier usuario puede ver la información de tu DashBoard.

ANEXO 3: GUÍA DE IMPLEMENTACION EN WATSON IoT

En esta guía se detallará todos los pasos para simular y enviar datos de sensores desde una Raspberry Pi hasta la plataforma Watson IoT. La Raspberry Pi simulará cinco variables correspondientes a un panel solar:

- Voltaje STC
- Voltaje NOCT
- Intensidad STC
- Intensidad NOCT
- Temperatura Panel.

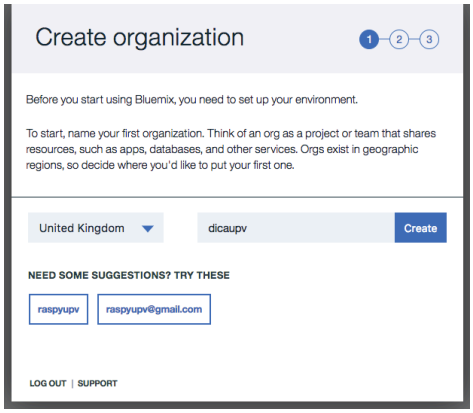
La Raspberry Pi usada tiene las siguientes características:

- Raspberry Pi 3
- Raspbian Jessie 8.0
- Linux 4.4.38-v7+
- Python 2.7

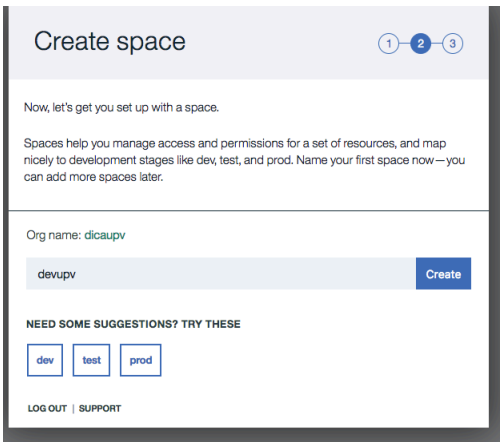
El primero que se debe hacer es crear una cuenta en la plataforma Watson IoT en el siguiente link:

https://console.bluemix.net/registration/?cm_mc_uid=16246976454914914061630&cm_mc_sid_50200000=1491856472&cm_mc_sid_52640000=1491856472

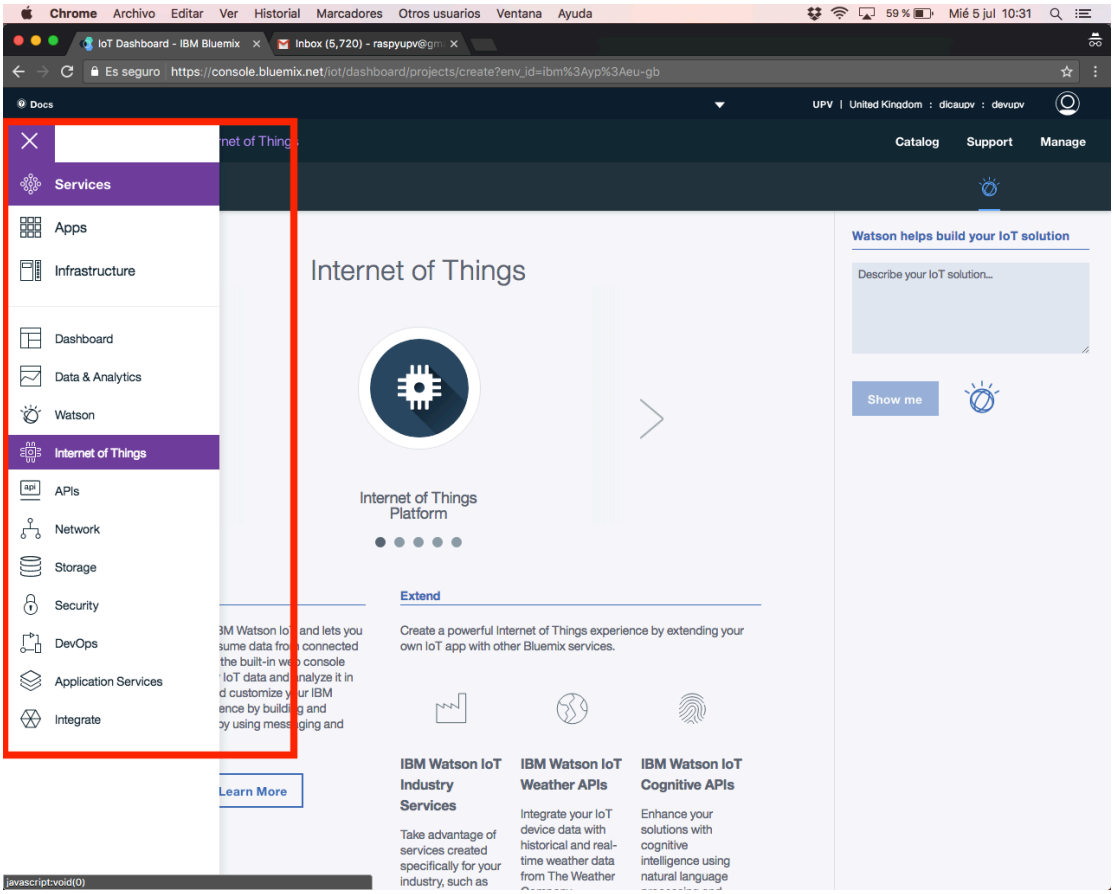
Una vez creada la cuenta ingresamos con nuestras credenciales a la plataforma. Estando allí debemos crear una organización la cual es el entorno donde las aplicaciones y servicios de IBM estarán contenidas.



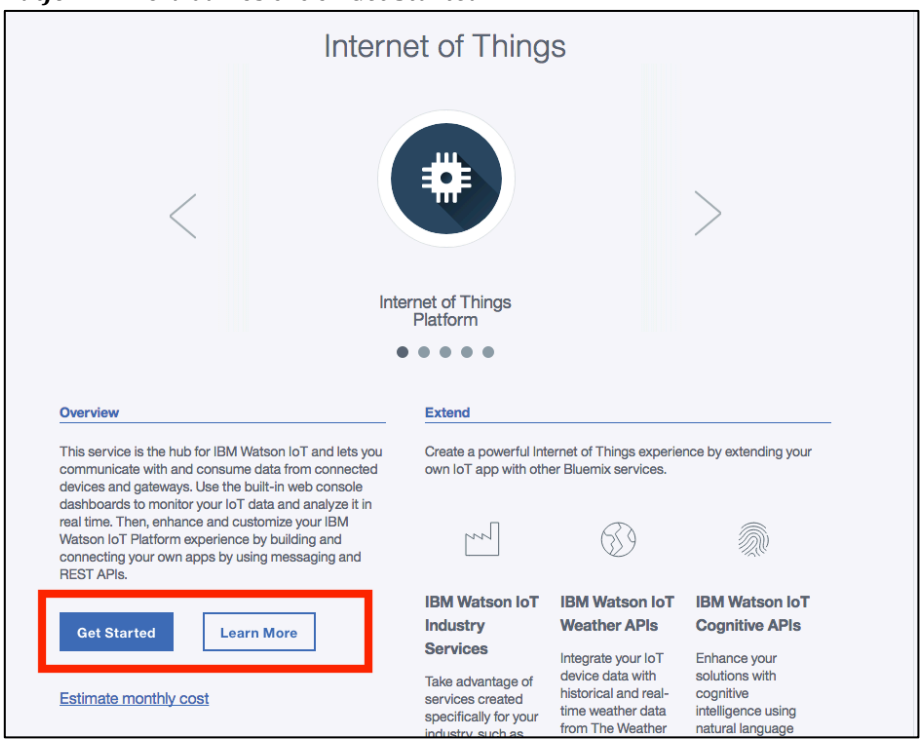
Ahora tendremos que crear el espacio del entorno de desarrollo.



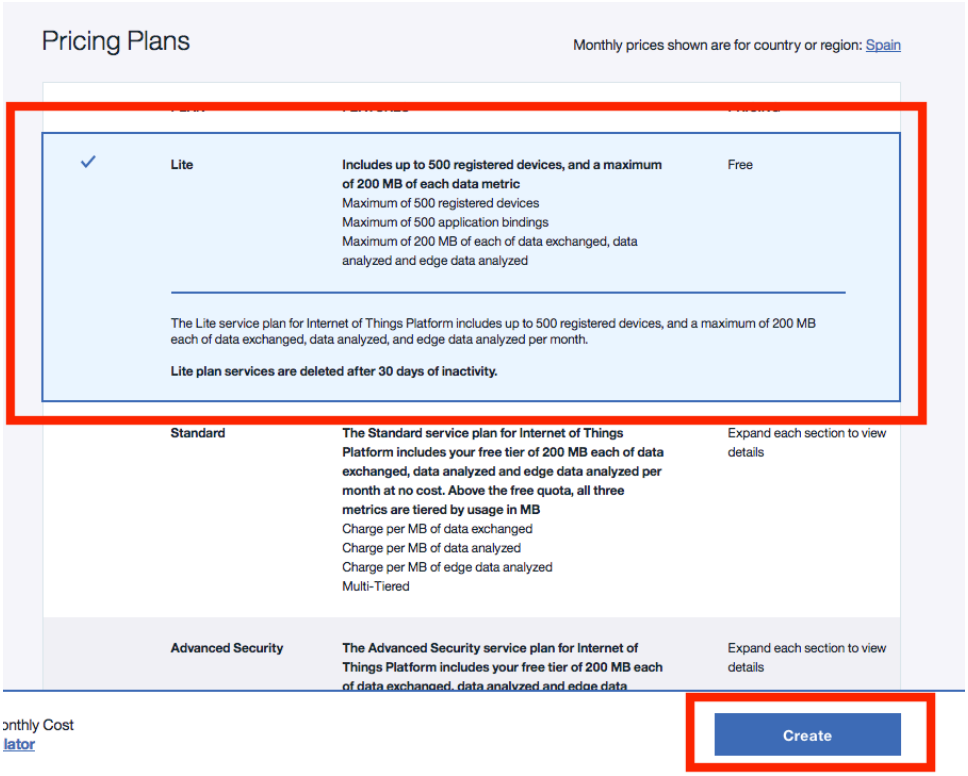
Hecho esto tendremos todo listo para seguir con los siguientes pasos. En la pagina principal de nuestro entorno de trabajo vamos a ***Menu >> Services*** y elegimos la opción de ***Internet of Things***.



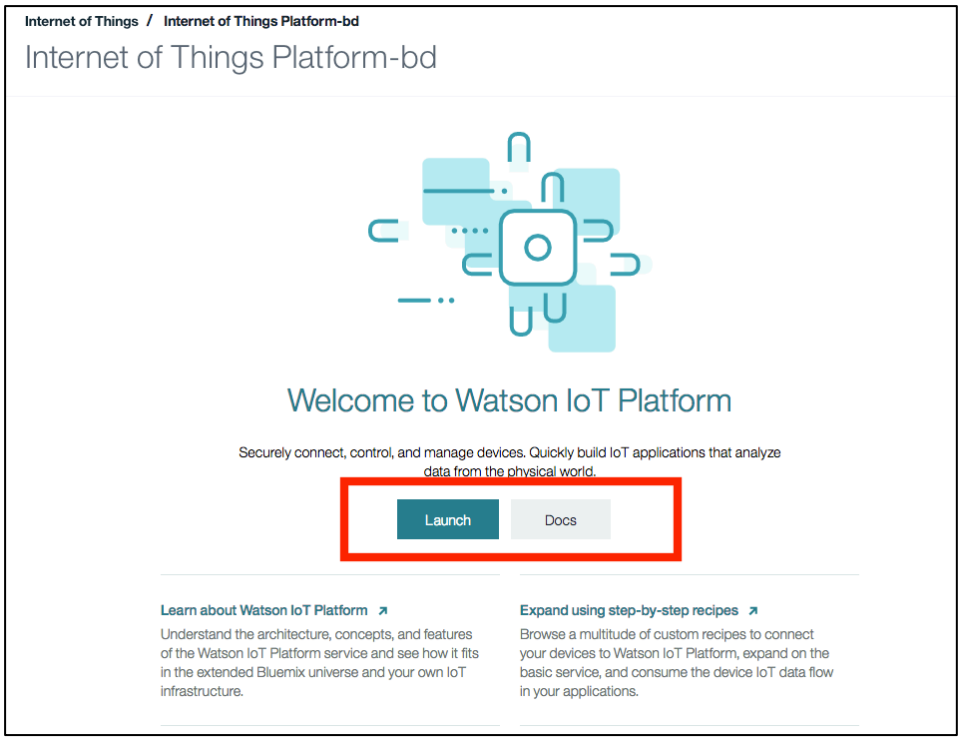
Aquí podemos elegir varios servicios relacionados con IoT pero nosotros elegiremos ***Internet of Things Platform***. Ahora damos clic en ***Get Started***



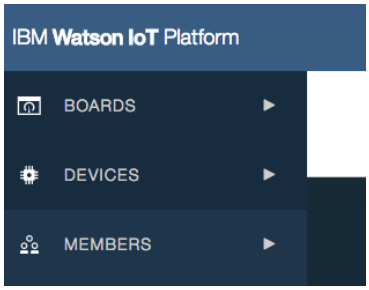
Esto nos llevará a una nueva página donde especificará en que consiste el servicio. Además de elegir el tipo de plan de suscripción. Como estamos en modelo de prueba escogemos la opción ***Lite*** que es gratuita. Después damos clic en ***Create***



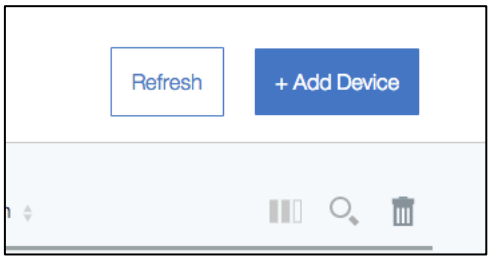
Se actualizara la pagina mostrando el servicio de IoT lista para ser usado. Damos clic en ***Launch***.



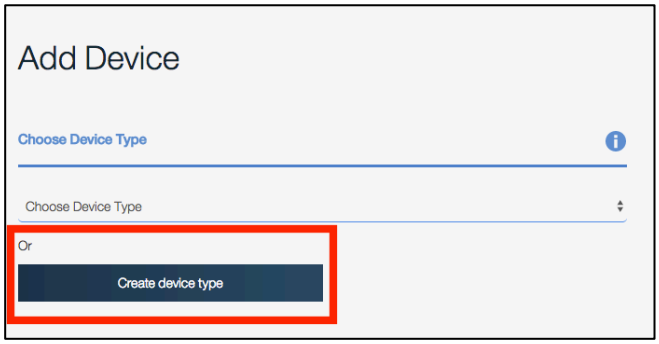
Con esto de abrirá una nueva pagina donde debemos empezar por registrar los dispositivos (que representan el panel solar) que se van a conectar a la plataforma Watson IoT. Para ello vamos a al menú de la parte izquierda y seleccionamos ***DEVICES***



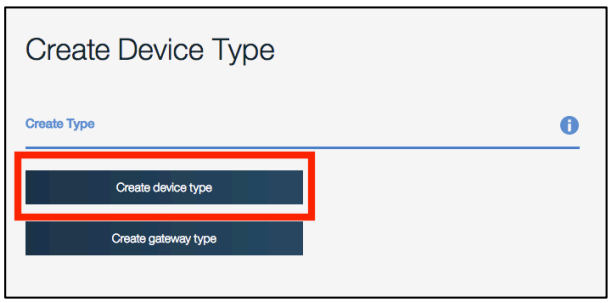
En esta nueva pagina elegimos la opción de ***+Add Device***



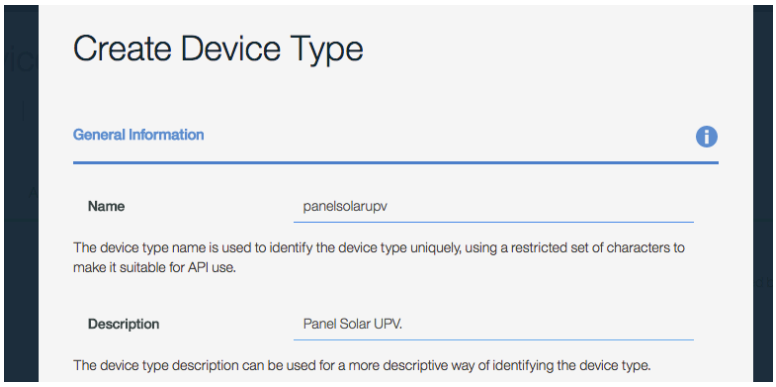
Se abrirá un cuadro de dialogo donde vamos a crear un nuevo tipo de dispositivo que será el modelo en que se basa el Panel Solar que contiene las cinco variables.



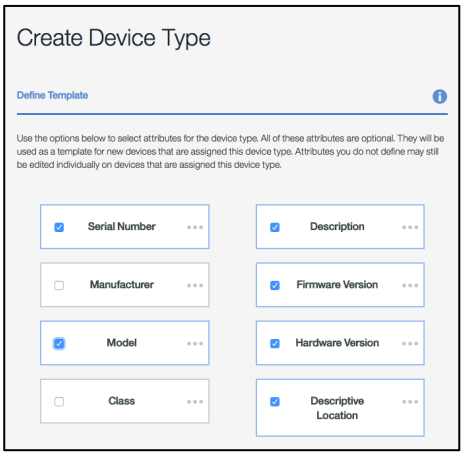
Ahora elegimos la opción de ***Create device type***



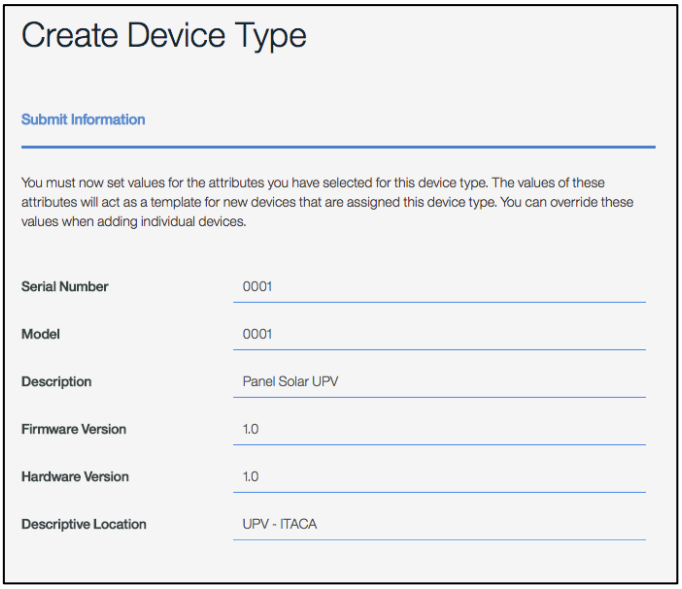
Introducimos un nombre y una descripción. Después le damos clic en ***Next***.



Después podemos crear un ***template*** de este nuevo tipo de dispositivo donde podremos elegir las características del mismo.

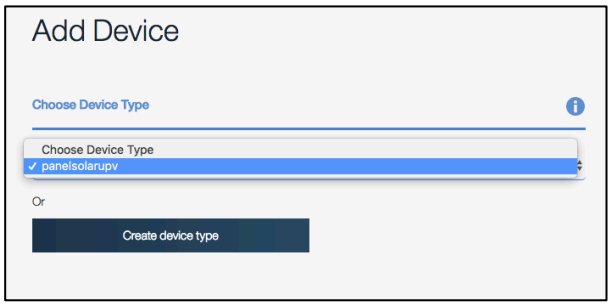


Ahora introducimos estas nuevas características con la información pertinente.

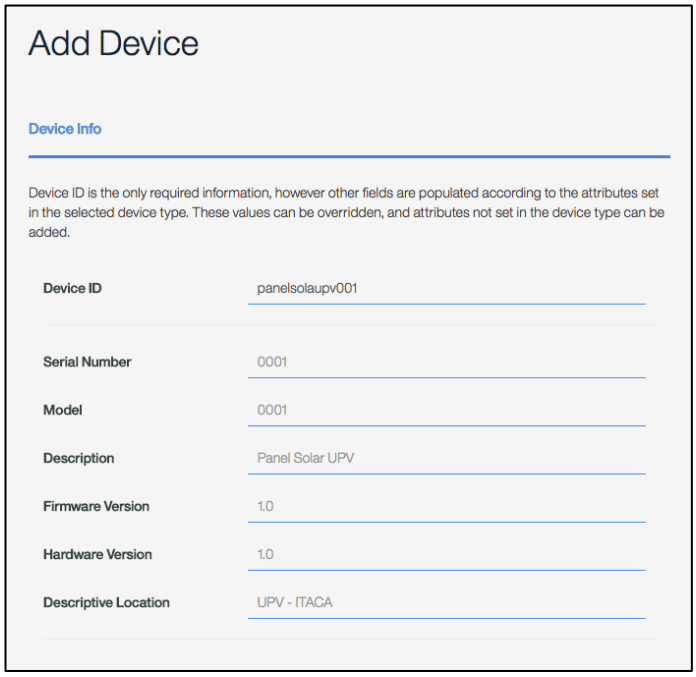


Después de ello nos pedirá si deseamos introducir ***METADA*** en formato JSON, como este no el caso damos clic en ***Create*** que creará el tipo de dispositivo y nos llevará a la ventana inicial

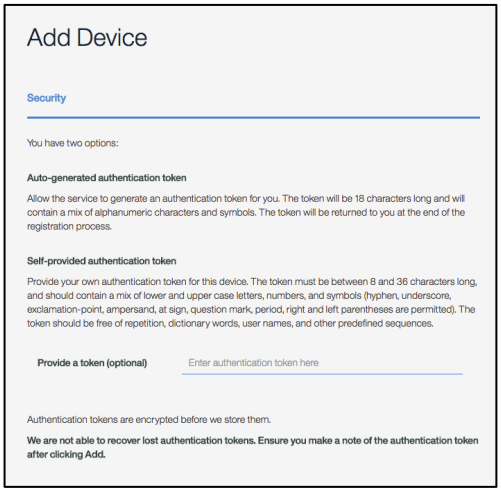
Ahora como ya tenemos creado el tipo de dispositivo podremos crear nuevos dispositivos basados en el modelo creado. Elegimos el tipo de dispositivo creado. Y damos clic en ***Next***



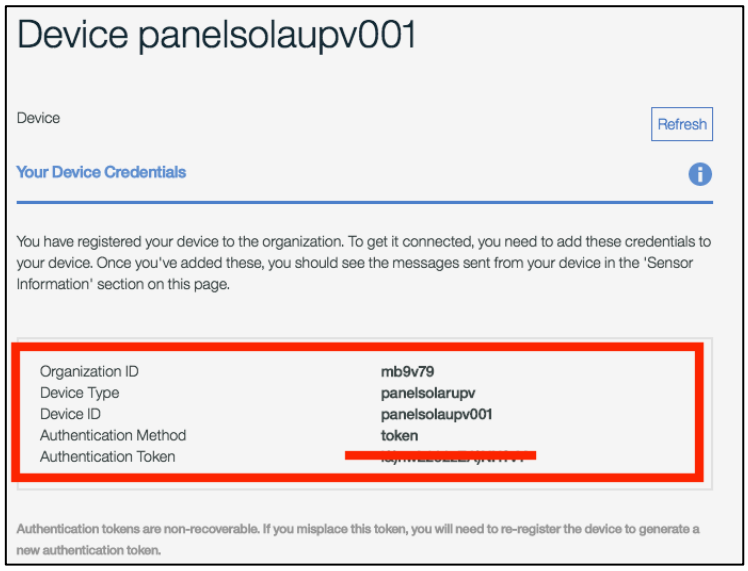
Ahora ingresamos la información necesaria para identificar nuestro dispositivo. En esta caso solo ***Device ID*** ya que lo demás información la toma del tipo de dispositivo tipo anteriormente creado.



Aquí también nos pide ingresar ***METADA*** en formato JSON lo cual en este ejemplo no se usa. Ahora tendremos la opción de crear un token nosotros mismos o que se autogenere uno. Dando clic en Next el sistema autogenerar un token que es la opción mas segura.



Ahora nos mostrará un sumario con toda la información y opciones elegidas con anterioridad. Damos clic en ***ADD*** para finalizar el proceso. Después de ello se abrirá un cuadro de dialogo mostrando la información del dispositivo como es el ***Device ID*** y el token. Esta información debe ser anotada por el administrador para ser usado en un futuro ya que el servicio no lo vuelve a mostrar mas.



Ahora creado el dispositivo en la plataforma Watson IoT es necesario desde la Raspberry enviar la información de los sensores (variables). Para ello en nuestra Raspberry instalaremos un cliente Python para la comunicación con Watson IoT.

$ sudo pip install ibmiotf

En el repositorio https://github.com/edisonupv/iot-servicesexamples esta disponible un script en python para Watson IoT llamado ***panelsolarIBM.py*** el cual contiene todo lo necesario para generar las variables aleatoriamente y el formato de mensaje en JSON que será enviada a la plataforma. En este script solo es necesario modificar los parámetros del ID de la organización, tipo de dispositivo, el ID del dispositivo y el token. Toda esta información es la que anteriormente se mencione que había que tomar nota.

```
organization = "****"                #Your organization ID
deviceType = "**********"            #Your device type
deviceId = "**********"              #Your device ID
authMethod = "token"
authToken = "**********)"            #Your Token device
```

Ahora ejecutamos el script

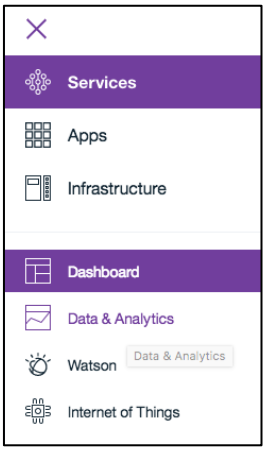
$ python panelsolarIBM.py

Si todo lo anterior esta configurado debería aparecer una respuesta similar a esta.

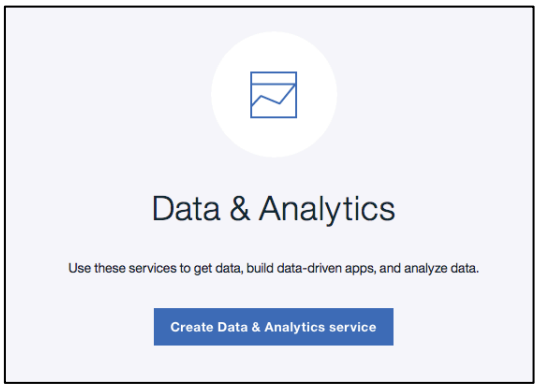
```
pi@raspyedison:~ $ python panelsolar002IBM.py
2017-07-05 11:35:47,233   ibmiotf.device.Client      INFO    Connected successfully: d:cg6c6c:panelsolarupv:panelsolarupv0
Confirmed event %s received by IoTF

2017-07-05 11:35:48,237   ibmiotf.device.Client      INFO    Disconnected from the IBM Watson IoT Platform
2017-07-05 11:35:48,240   ibmiotf.device.Client      INFO    Closed connection to the IBM Watson IoT Platform
```

Se puede usar la herramienta ***crontab*** de Linux para que estos reportes los envía la Raspberry Pi automáticamente por ejemplo cada cinco o diez minutos.

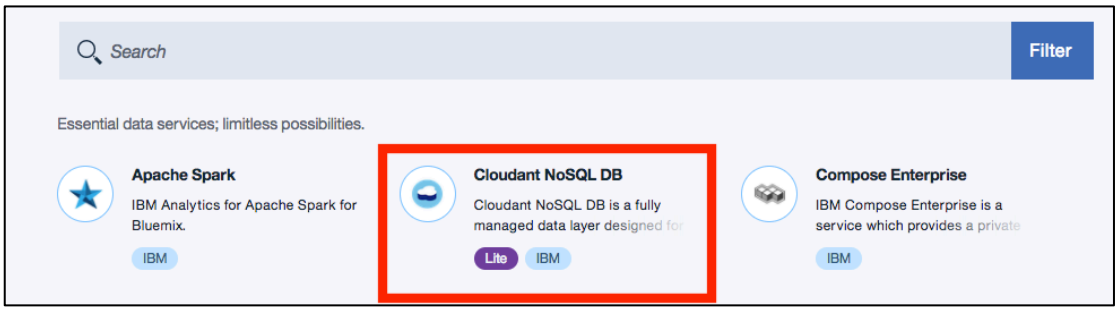
Ahora que estamos seguros que la información de la Raspberry se envía correctamente a la plataforma Watson IoT procedemos a implementar la base de datos donde la información de los sensores será almacenada. Para ellos vamos a al menú principal de IBM BLUEMIX y vamos a ***Menu*** >> ***Services*** >> ***Data & Analytics***



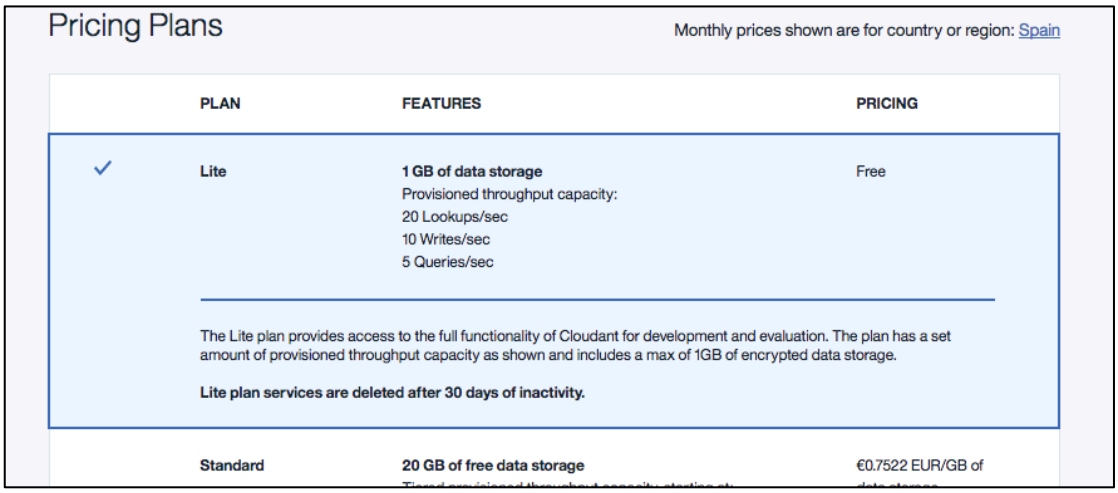
Elegimos ***Create Data & Analytics Service***



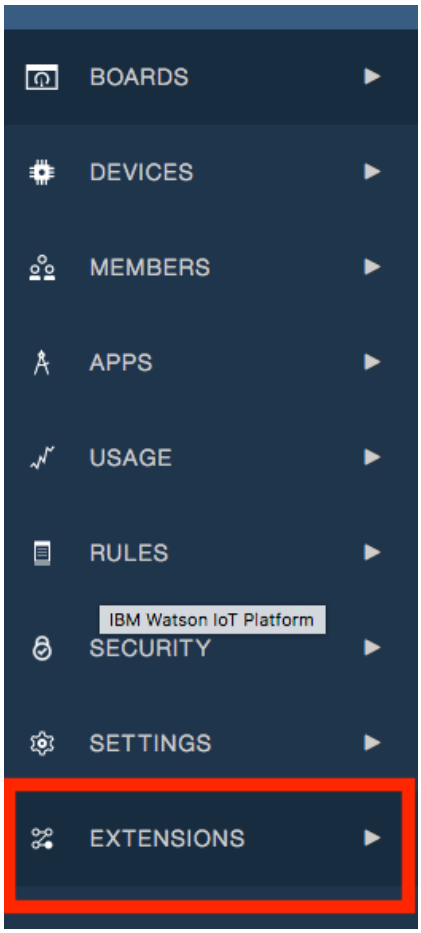
Se abrirá un buscador de servicios. Elegimos el servicio ***Cloudant NoSQL DB***



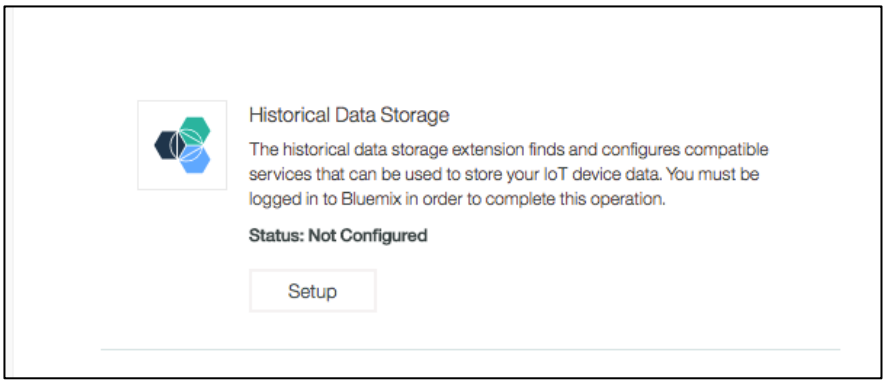
Elegimos el plan ***Lite*** que es gratuito y damos clic en ***Create***



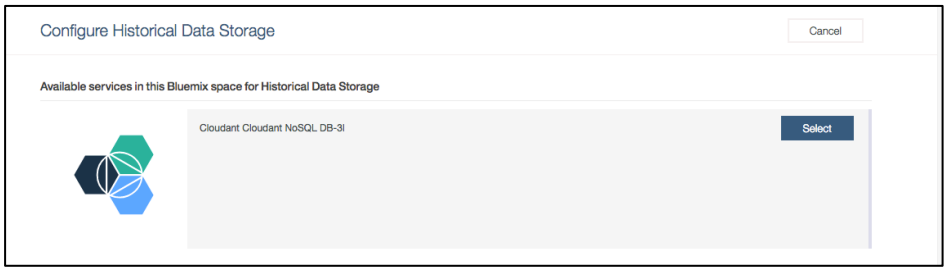
Ahora debemos conectar el servicio de IoT con la base de datos. Para ello dentro del servicio de ***Internet of Things Platform*** vamos a ***Extensions***



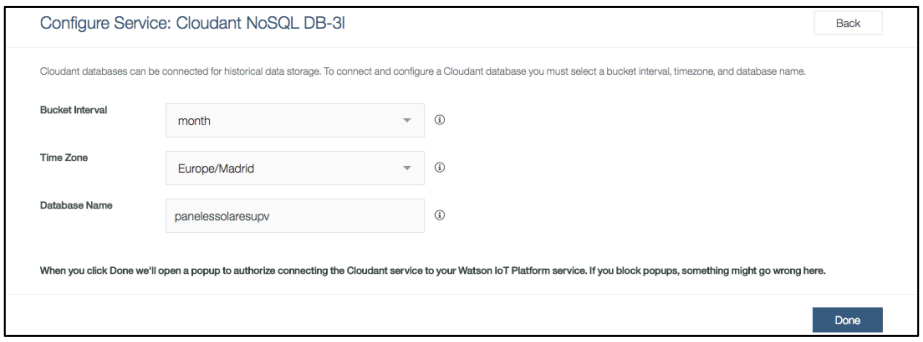
En la opción de ***Historical Data Storage*** damos clic en ***Setup***



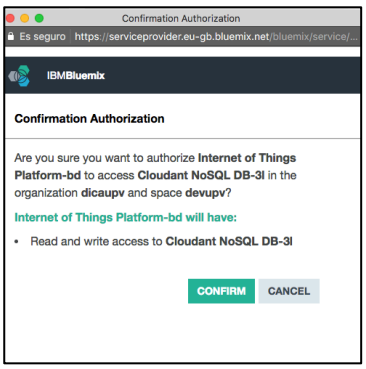
Seleccionamos la base de datos creada anteriormente.



En ***Bucket Interval*** elegimos ***month***, en ***Time Zone*** elegimos la de ***Madrid*** y ***Database Name*** un nombre a la base de datos que contendrá la información de los sensores (varibles) del dispositivo (panel solar).

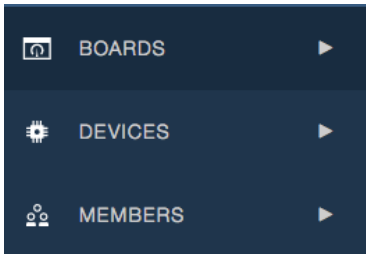


Antes de dar clic en Done hay que asegurase que los ventanas emergentes están habilitadas en el navegador ya que se abrirá un pagina emergente para validar la confirmación de la conexión entre estos dos servicios.

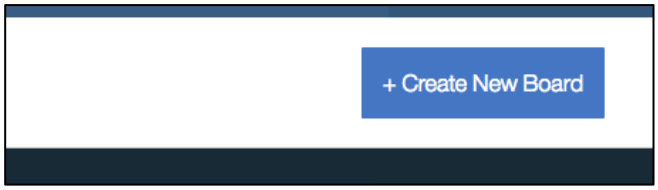


Si todo lo anterior ha sido correctamente configurado debería aparecer el ***Status*** como ***Configured***.

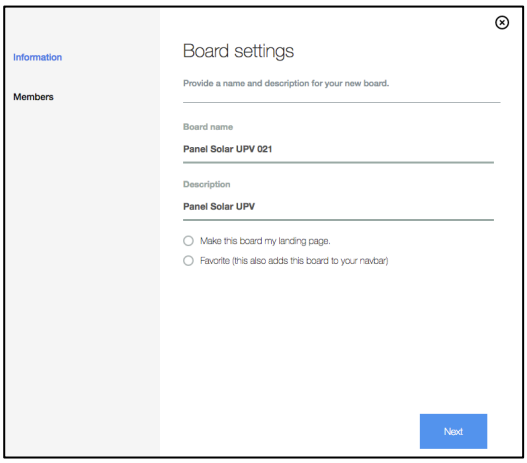
Ahora ya es posible construir las gráficas mostrando la información histórica de lo sensores. Para ello nos vamos a ***Boards***.



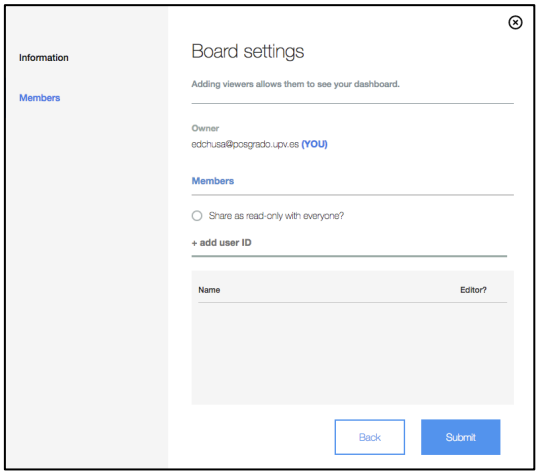
Damos clic en crear ***+ Create New Board***



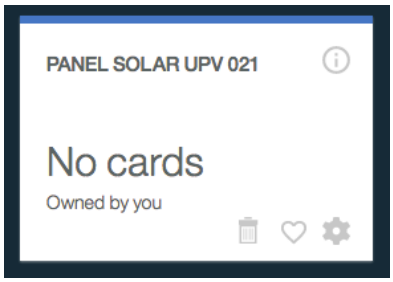
Ingresamos un nombre y una descripción de este nuevo Board.



También nos pedirá la opción de añadir a otros miembros. En este caso omitimos este paso.

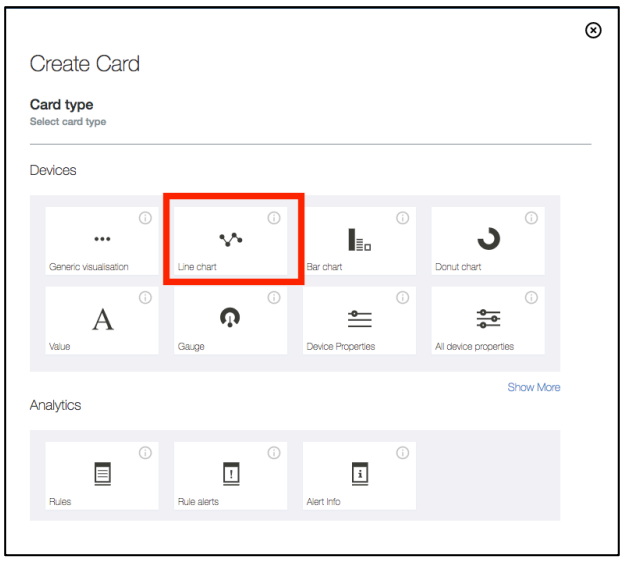


Una vez creado el dashboard debería mostrarse como vacío.

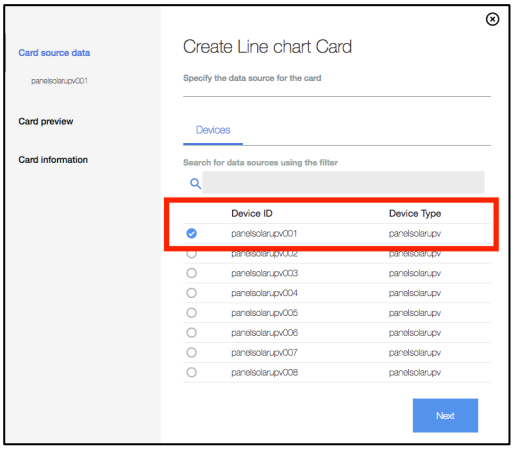


Hacemos clic sobre el y elegimos la opción de ***+ Add New Card***.

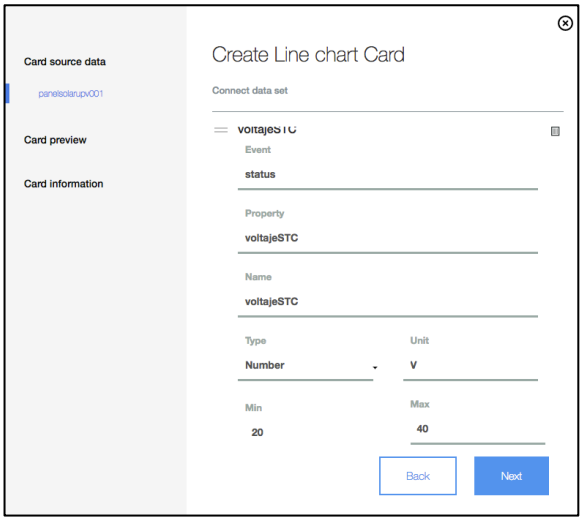
Se abrirá un cuadro de dialogo donde podremos elegir el tipo de gráfico. En este caso ***Line Chart***.



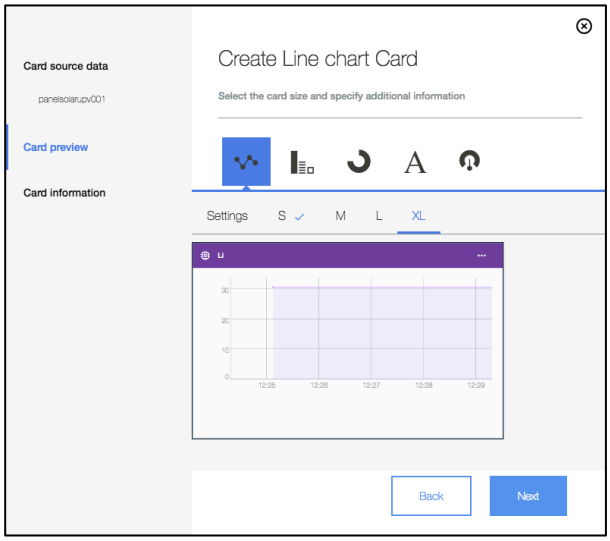
Después elegimos el dispositivo que proveerá de los datos a graficar.



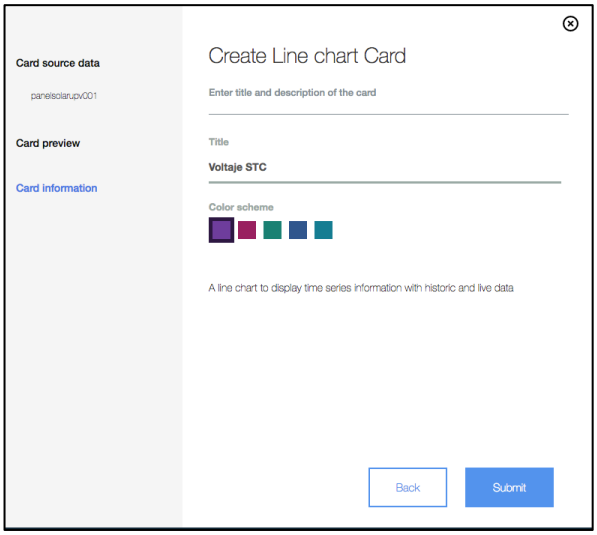
Ahora nos pedirá qué datos serán los graficados. En este caso sería la variable ***Voltaje STC*** registrado en el evento ***Status*** (el cual esta ya configurado en el script Python)



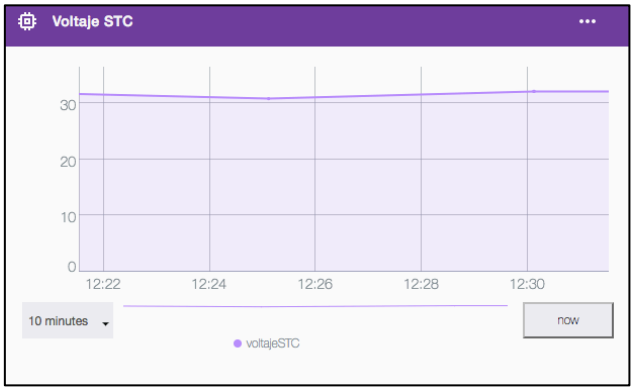
Ahora elegimos el tamaño de la gráfica en este caso ***XL*** para poder una mejor visualización.



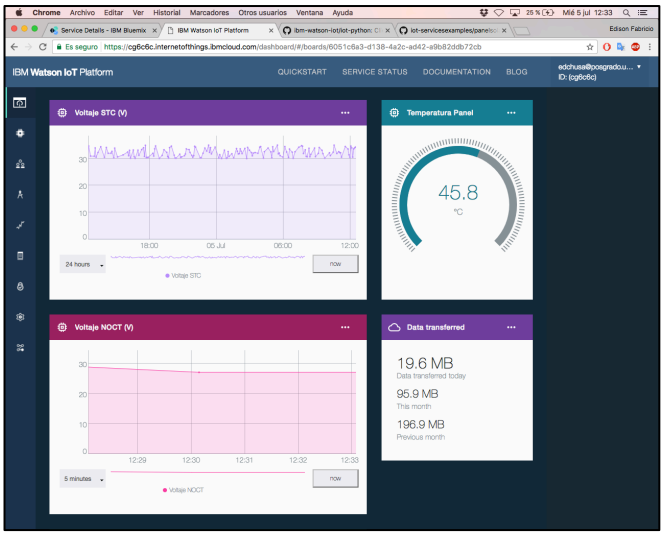
Elegimos un nombre de la gráfica y un color para mejor personalización.



Con ello debería quedar una gráfica similar a esta



Estos mismos pasos se pueden repetir para cada variable restante además de poder elegir otro tipo de gráficas. Por ejemplo se podría al final tener algo similar a esto.



Ahora podemos repetir el mismo procedimiento para añadir otros dispositivos y otras variables. Y para cada dispositivo se puede crear un Board con sus respectivas gráficas de las variables que le pertenecen.

En este repositorio se encuentran muchos mas ejemplos que se pueden implementar en una Raspberry Pi https://github.com/ibm-watson-iot/iot-python

Aquí se podrá encontrar información y guías de desarrollo relacionado con la plataforma Watson IoT: https://developer.ibm.com/recipes/tutorials/category/internet-of-things-iot/

ANEXO 4: GUÍA DE IMPLEMENTACIÓN EN SAMSUNG ARTIK CLOUD

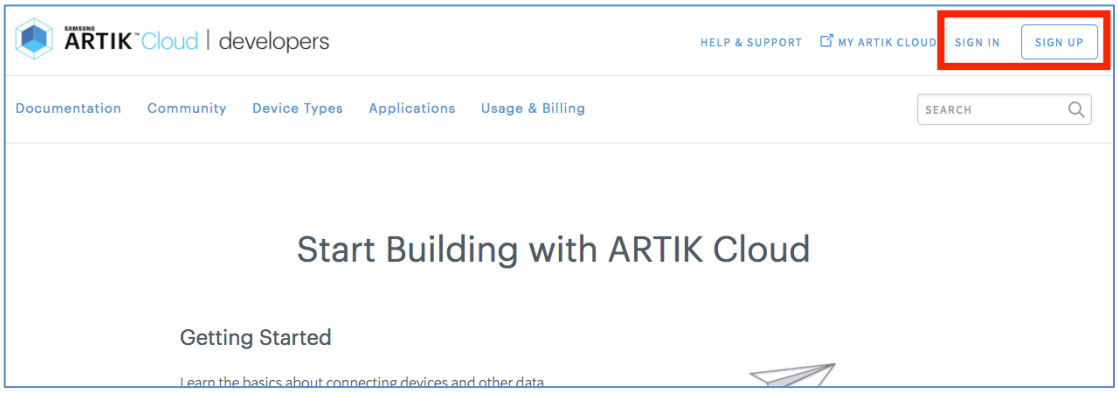
En esta guía se detallará todos los pasos para simular y enviar datos de sensores desde una Raspberry Pi hasta la plataforma Samsung Artik Cloud. La Raspberry Pi simulará cinco variables correspondientes a un panel solar:

- Voltaje STC
- Voltaje NOCT
- Intensidad STC
- Intensidad NOCT
- Temperatura Panel.

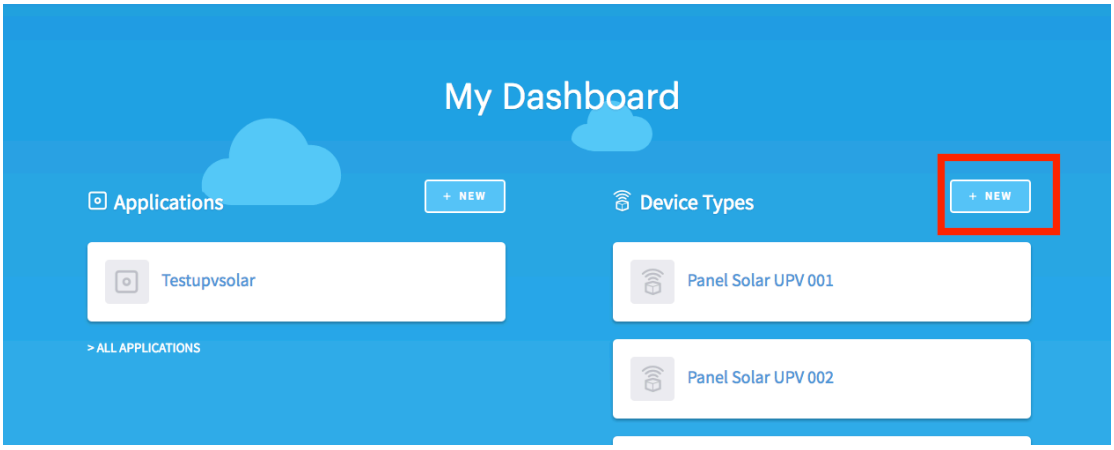
La Raspberry Pi usada tiene las siguientes características:

- Raspberry Pi 3
- Raspbian Jessie 8.0
- Linux 4.4.38-v7+
- Python 2.7

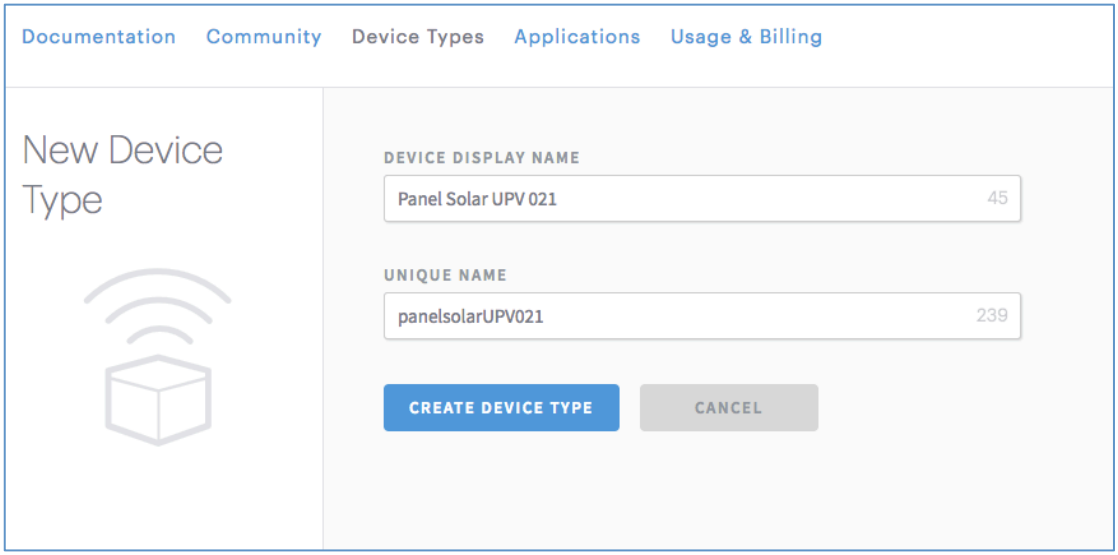
El primero que se debe hacer es crear una cuenta en la plataforma Artik Cloud en el siguiente link: https://developer.artik.cloud/. Una vez creada la cuenta ingresamos a la plataforma.



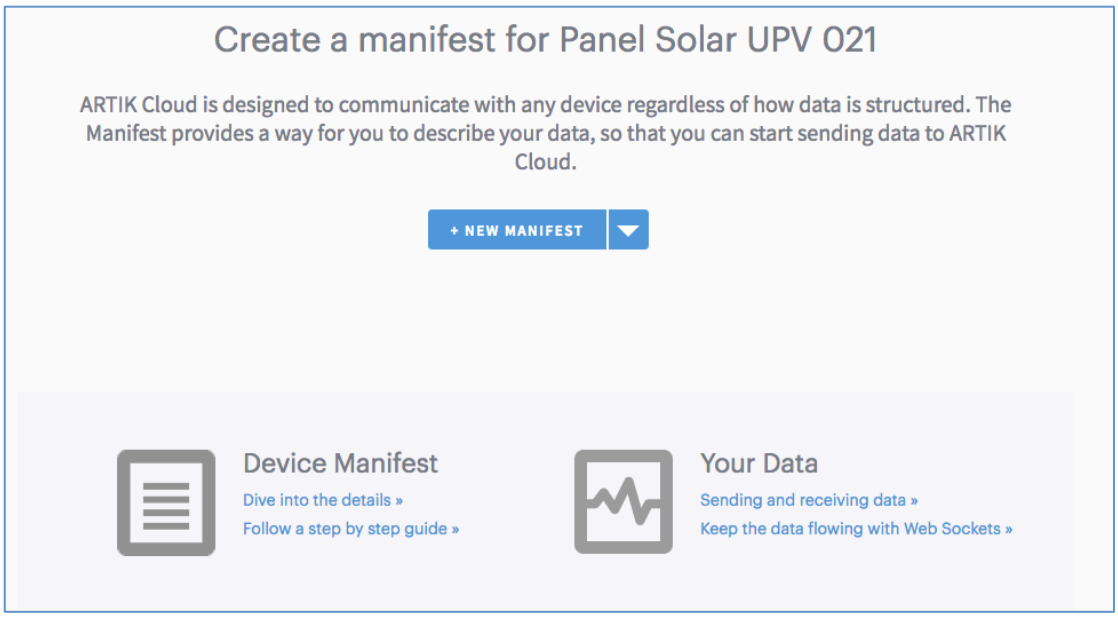
Un vez ingresamos a la cuenta nos situamos en la parte correspondiente a ***My Dashboard*** y agregamos un nuevo dispositivo.



Ingresamos el nombre que se mostrara del dispositivo y un identificador único del dispositivo. Después damos clic en ***CREATE DEVICE TYPE***.



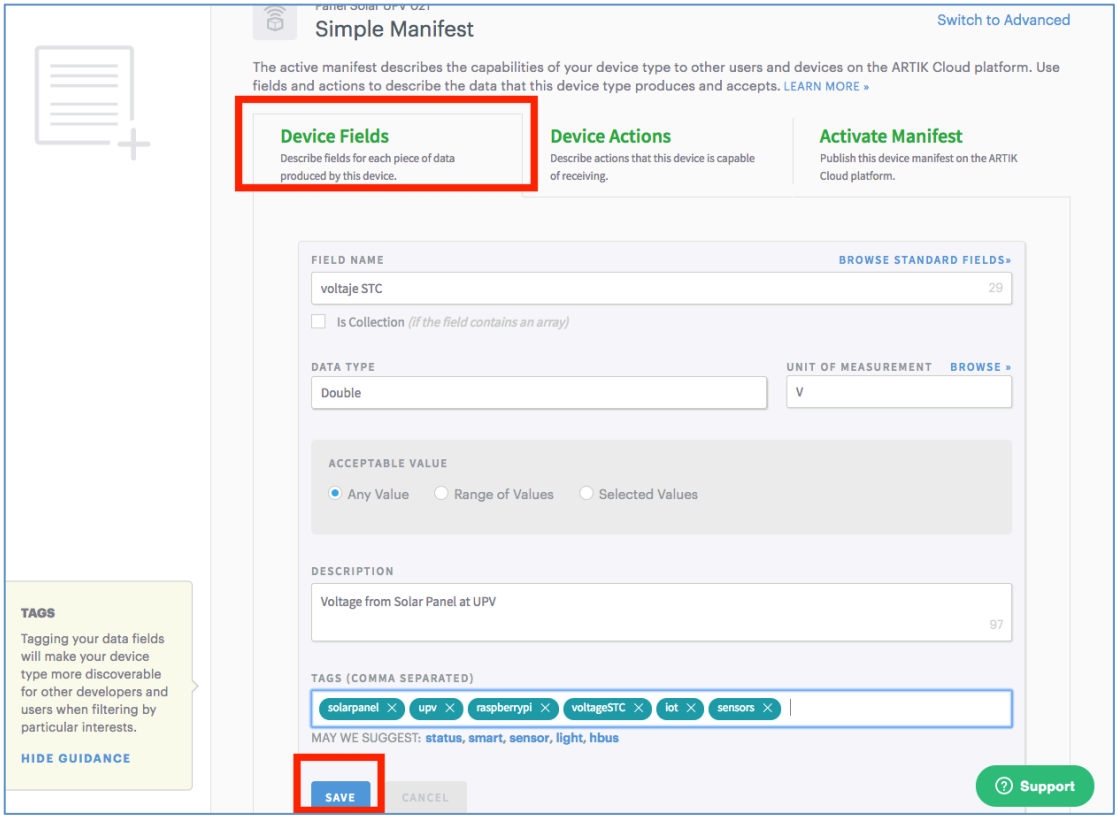
Ahora tenemos dos opciones: crear un nuevo ***Manifest*** para nuestros dispositivo o subir una previamente ya configurado. En este caso crearemos un nuevo desde cero. Damos clic en ***+ NEW MANIFEST***.



En la pestaña de ***Device Fields*** ingresamos cada uno de los atributos que caracterizaran a nuestro dispositivo, es decir la variables del panel solar. Por ejemplo para Voltaje STC ingresamos de la siguiente manera

FIELD NAME: voltajeSTC
DATA TYPE: Double
UNIT OF MEASUREMENTE: V
ACCEPTABLE VALUE: Any Value
DESCRIPTION: Voltaje from Solar Panel at UPV
TAGS: solarpanel, upv, raspberrypi, voltajeSTC, iot,sensors.

Damos clic en ***SAVE***



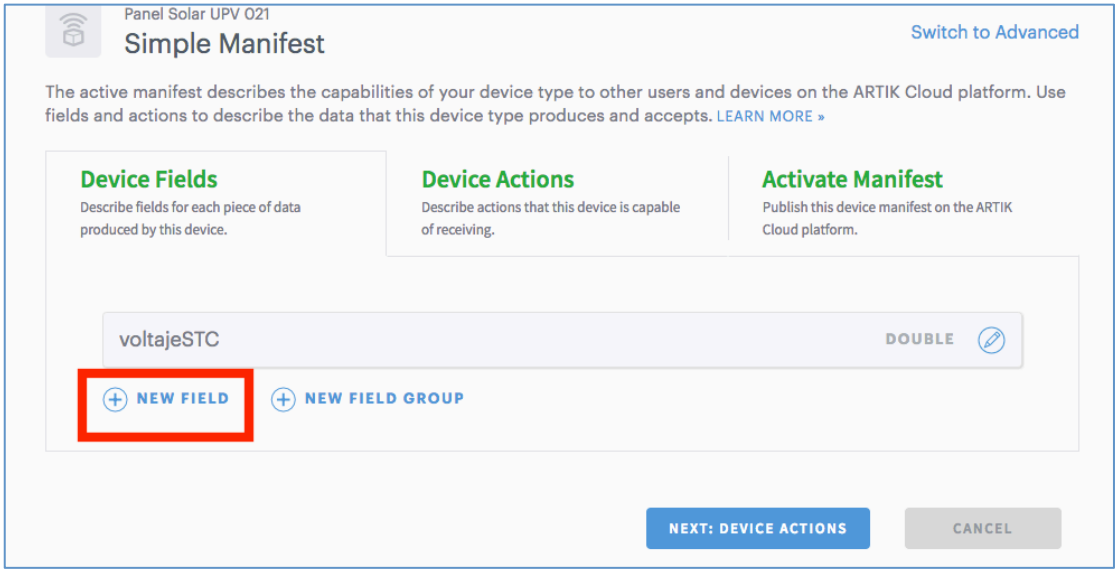
Ahora falta añadir las demás variables. Damos clic en el botón ***+ NEW FIELD*** y añadimos cada variable como se muestra a continuación:

FIELD NAME: voltajeNOCT
DATA TYPE: Double
UNIT OF MEASUREMENTE: V
ACCEPTABLE VALUE: Any Value
DESCRIPTION: Voltaje from Solar Panel at UPV
TAGS: solarpanel, upv, raspberrypi, voltajeNOCT, iot,sensors.

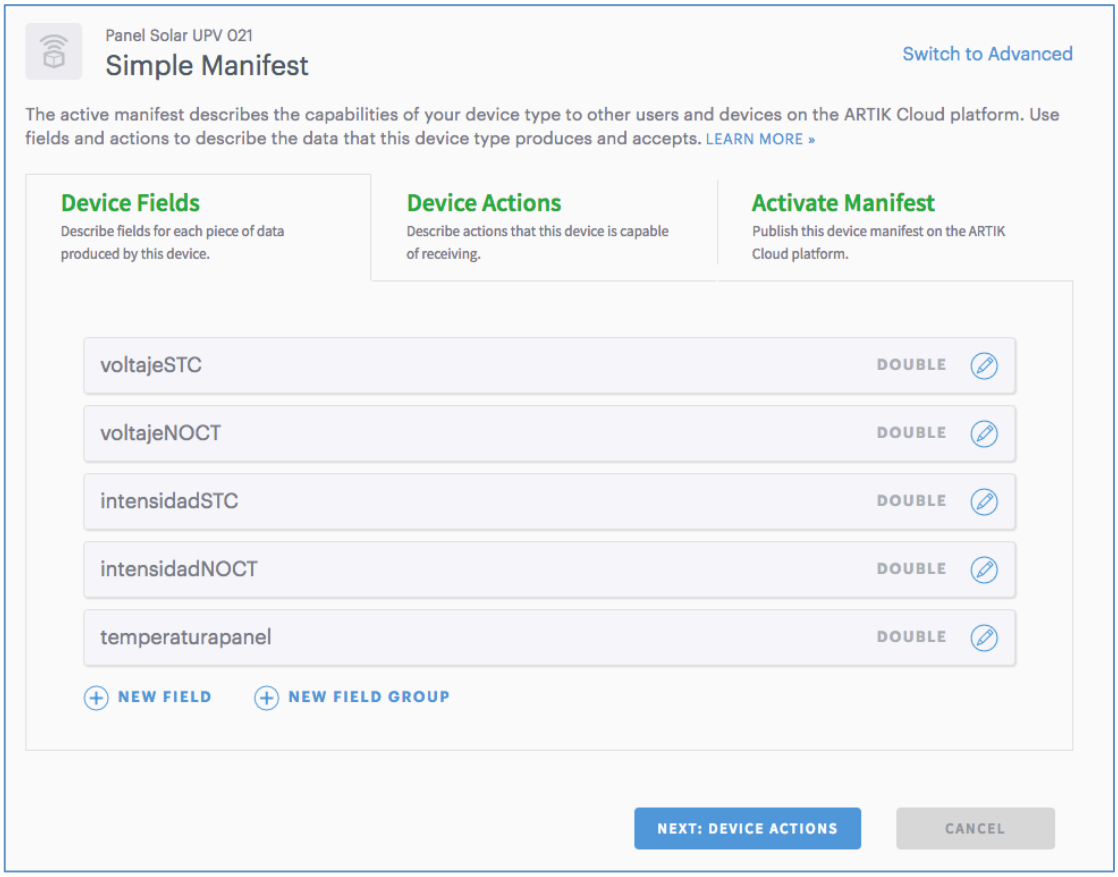
FIELD NAME: intensidadSTC
DATA TYPE: Double
UNIT OF MEASUREMENTE: A
ACCEPTABLE VALUE: Any Value
DESCRIPTION: Current from Solar Panel at UPV
TAGS: solarpanel, upv, raspberrypi, corrienteSTC, iot,sensors.

FIELD NAME: intensidadNOCT
DATA TYPE: Double
UNIT OF MEASUREMENTE: A
ACCEPTABLE VALUE: Any Value
DESCRIPTION: Current from Solar Panel at UPV
TAGS: solarpanel, upv, raspberrypi, currentNOCT, iot,sensors.

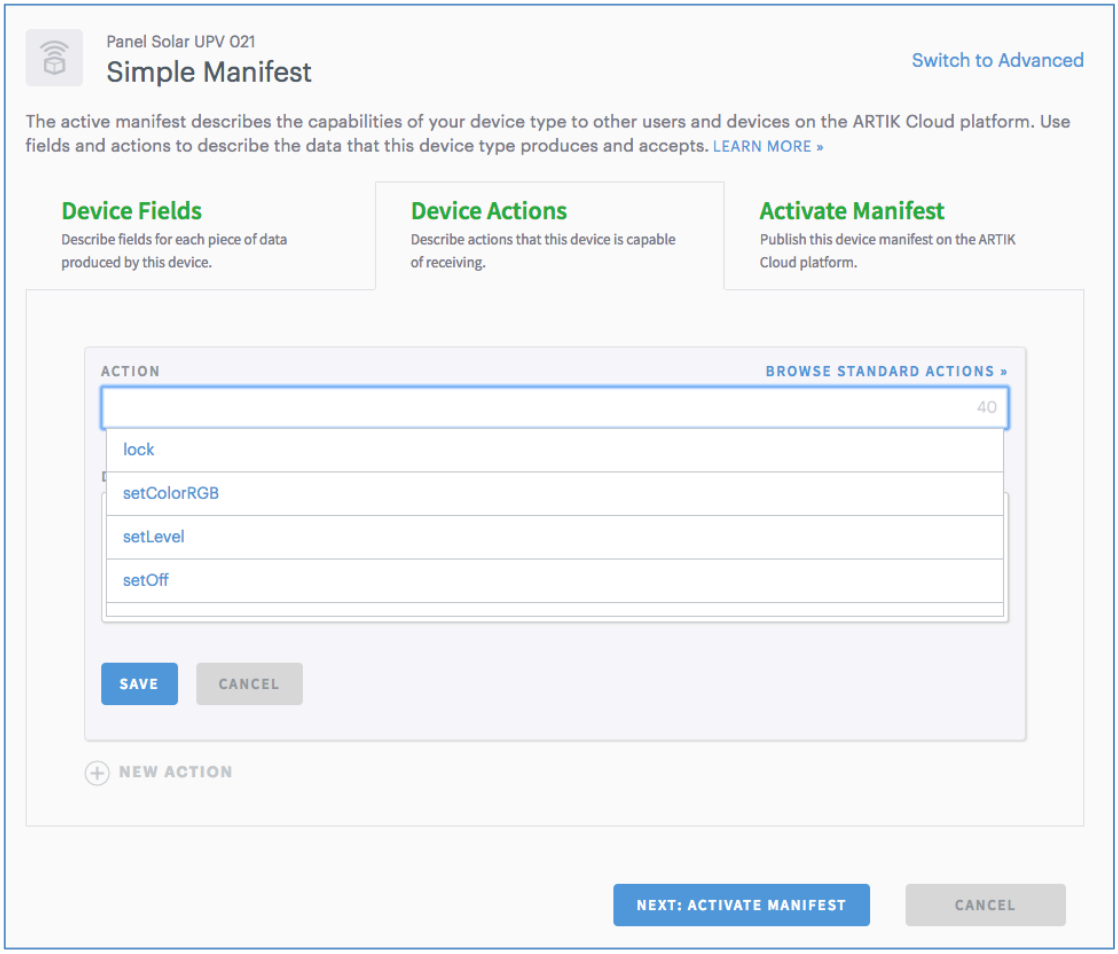
FIELD NAME: temperaturapanel
DATA TYPE: Double
UNIT OF MEASUREMENTE: C
ACCEPTABLE VALUE: Any Value
DESCRIPTION: Temperature from Solar Panel at UPV
TAGS: solarpanel, upv, raspberrypi, temperature, iot, sensors.



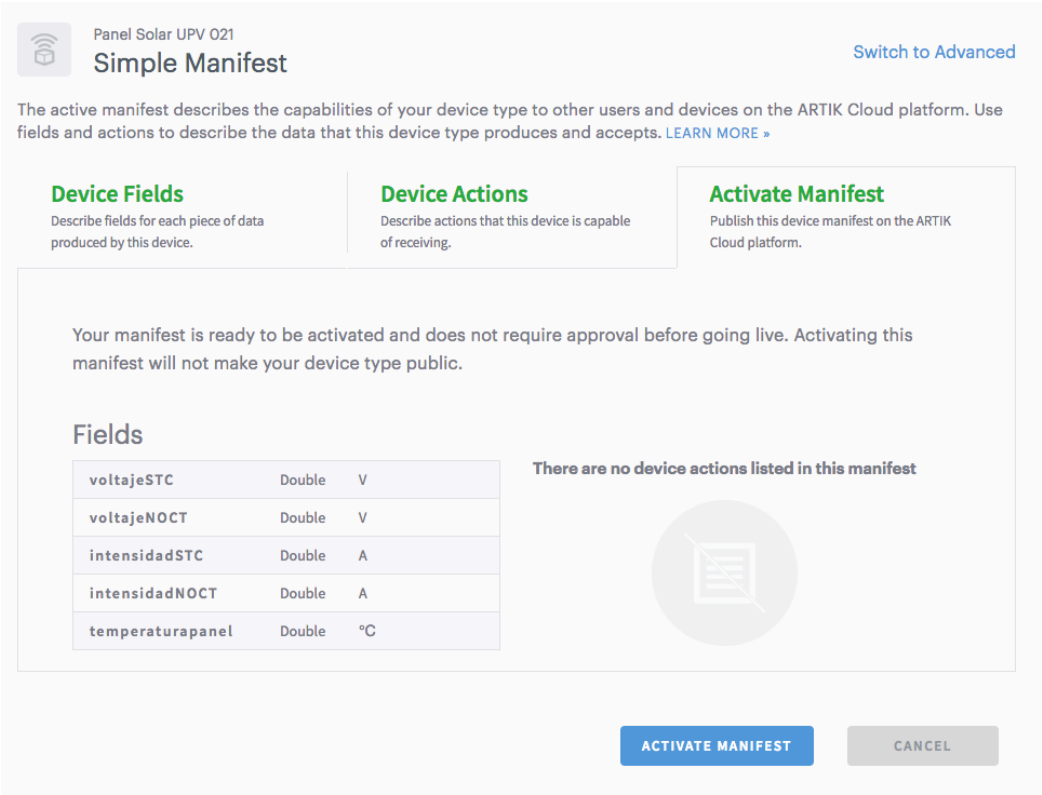
Al final debería quedar algo similar a esto:



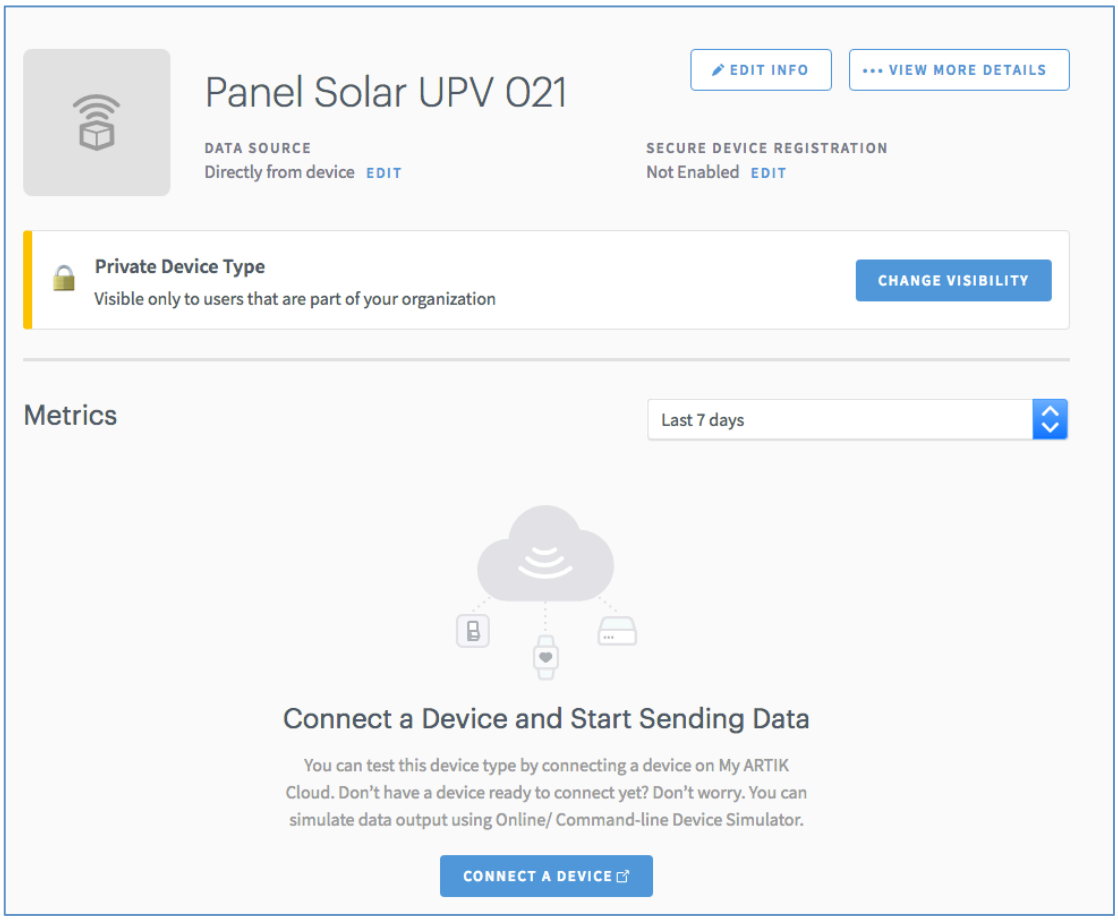
Ahora damos clic en ***NEXT: DEVICE ACTIONS***. Esta característica por ahora no será usada y quedara en blanco. Damos clic en ***NEXT: ACTIVATE MANIFEST***



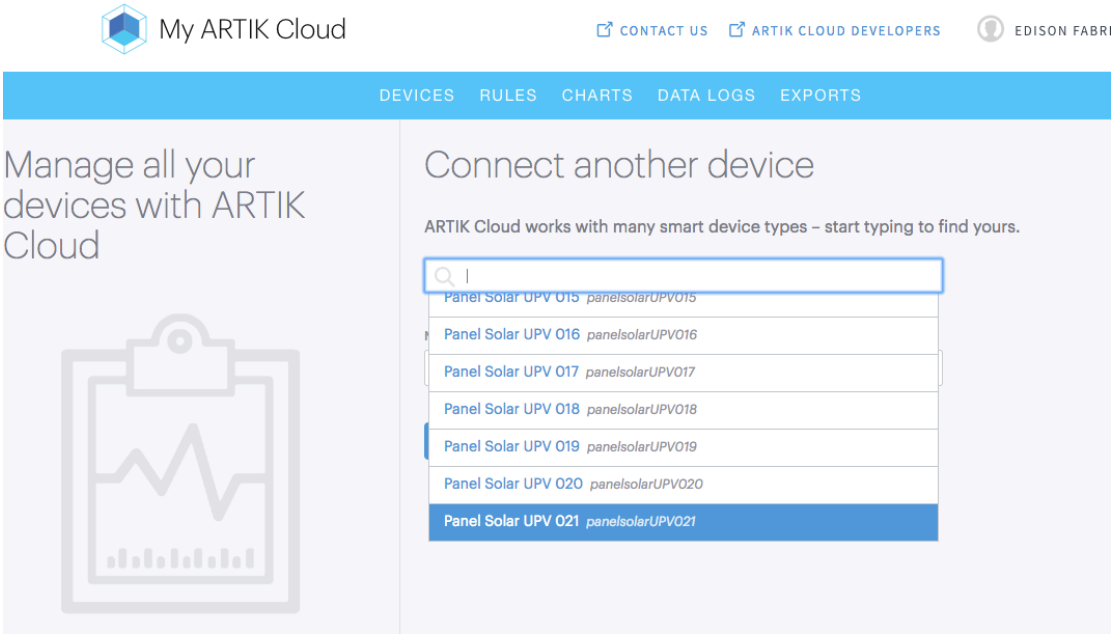
Ahora damos clic en ***ACTIVATE MANIFEST***

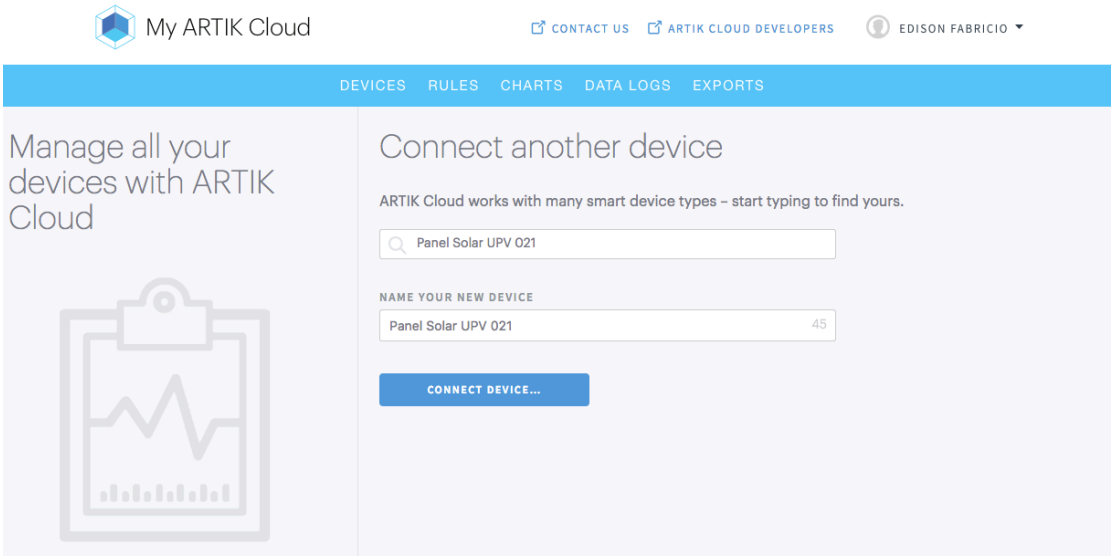


Con esto último nuestro dispositivo esta lista para poder empezar a enviar información y que Artik la almacene en la nube.

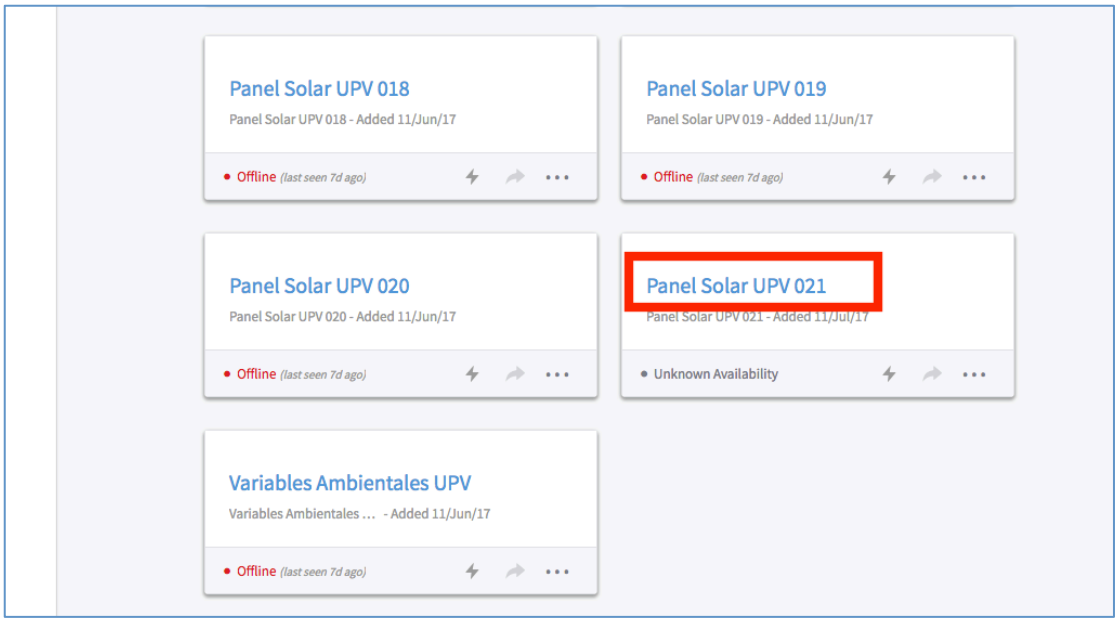


Damos clic en ***CONNECT A DEVICE.*** Se abrirá una nueva pagina que es la plataforma Cloud de Artik. Aquí añadimos el dispositivo creado y damos clic en ***CONNECT DEVICE***

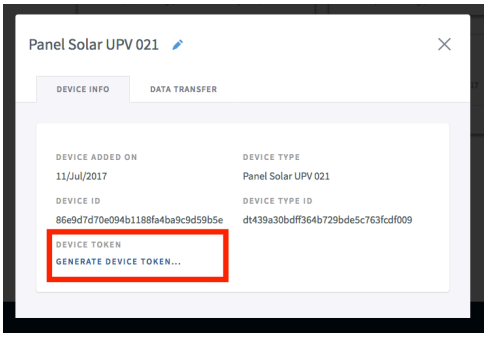




Después de ellos estaremos en otra pagina donde podremos navegar entre los diferentes dispositivos nuestros. Encontramos el anteriormente creado y damos clic sobre su nombre.



Se abrirá una pagina flotante que mostrara información útil como el ***Device ID*** y el ***Device Type ID***. Damos clic en ***Generate Device Token*** para obtener el token necesario para realizar la autenticación cuando enviemos las variables desde la Raspberry Pi hasta el servicio en la nube de Artik.



Con esta información ahora ya podemos proceder a configurar la Raspberry Pi para que envíe las variables. En el repositorio https://github.com/edisonupv/iot-servicesexamples existe un archivo llamado ***panelsolarARTIK.js*** escrito en lenguaje ***JSON*** que esta ya configurado para enviar las variables desde la Raspberry Pi. Descargar este fichero en la Raspberry Pi.

En la Rasberry Pi ingresar las siguiente ordenes

$sudo apt-get update
$ sudo apt-get upgrade
$npm install ws

Las anteriores ordenes actualizaran el sistema, repositorios e instalaran la librerías Web Socket para conectar la Raspberry Pi con Artik Cloud. Del fichero ***panelsolarARTIK.js*** modificar solamente los siguiente

```
var device_id = "*****************************";      //Your DEVIDE ID
var device_token = "*****************************";   //Your DEVICE TOKEN
```

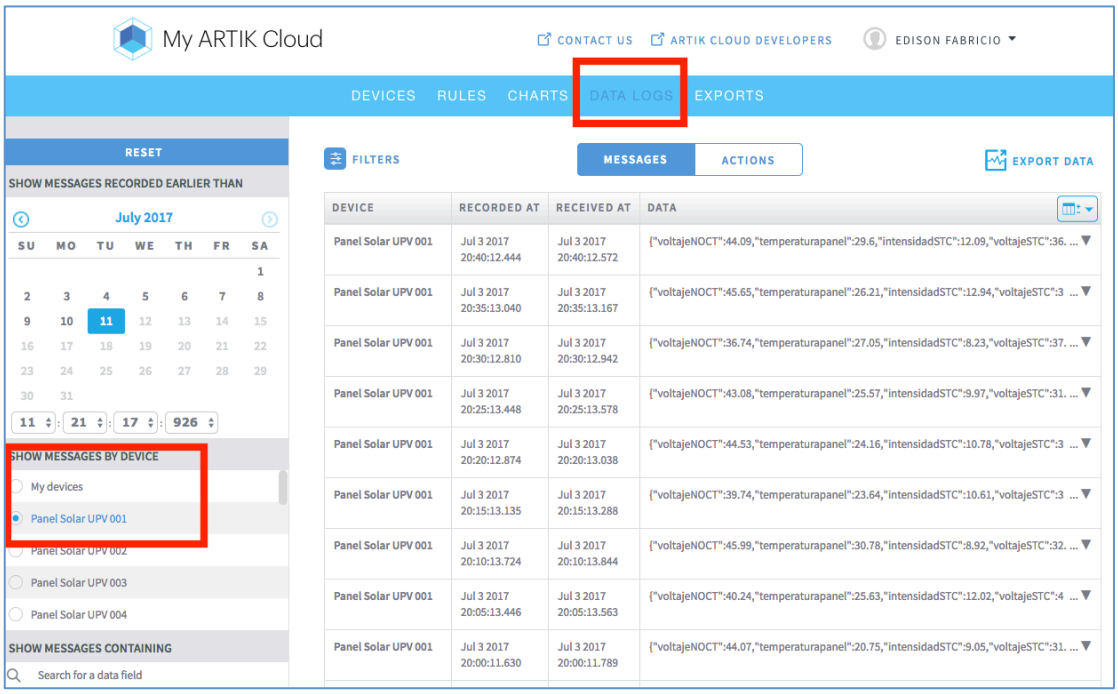
Que corresponde a los identificadores y token del dispositivo anteriormente obtenidos en Artik Cloud. Ahora ejecutamos el script:

$ node panelsolarARTIK.js

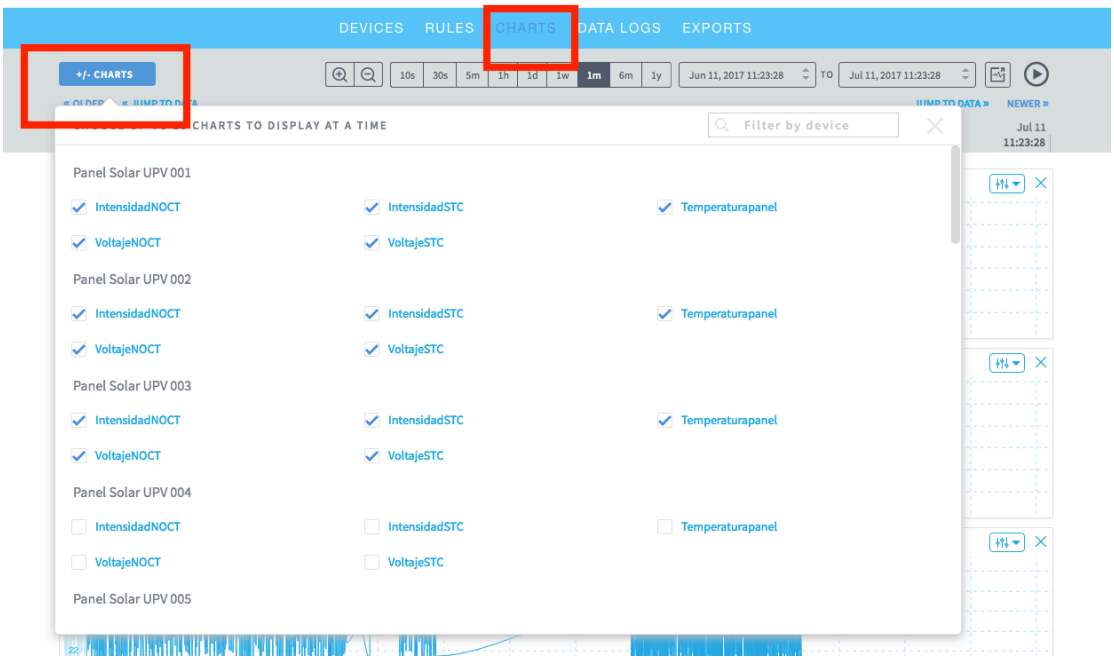
Si todo esta correcto debería aparecer un mensaje similar a este

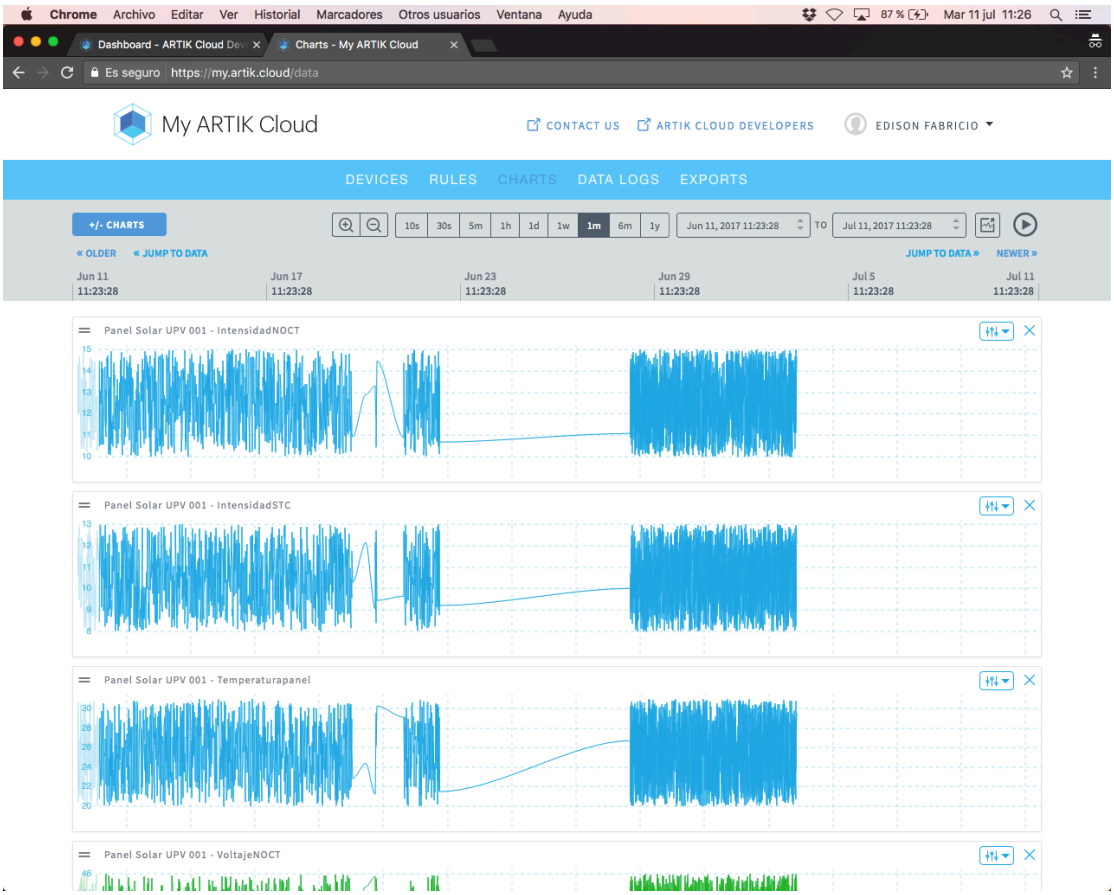
Podemos usar la herramienta ***crontab*** de Linux para programar el script para que se ejecute cada cierto tiempo, por ejemplo cada 15 minutos.

Para visualizar la información tanto de manera gráfica como en desde su base de datos nos vamos a la opción de ***DATA LOGS*** y elegimos el dispositivo del que queremos consultar sus datos.



Es posible observar las graficas en tiempo real de la información de los sensores de los dispositivos. Para ello vamos a ***CHARTS*** y elegimos el tiempo histórico de los datos a visualizar y los dispositivos.





Ahora es posible añadir nuevos dispositivos, nuevas variables y modificar el script para adaptarlo a nuestras necesidad y empezar a enviar información hacia Artik Cloud. En los siguiente link podrán encontrar más información relacionado con el desarrollo de aplicaciones para Artik Cloud

https://developer.artik.cloud/documentation/tutorials/how-to/

ANEXO 5: GUÍA DE IMPLEMENTACIÓN EN MICROSOFT AZURE IoT SUITE

En esta guía se detallará todos los pasos para simular y enviar datos de sensores desde una Raspberry Pi hasta la plataforma Microsoft Azure IoT. La Raspberry Pi simulará cinco variables correspondientes a un panel solar:

- Voltaje STC
- Voltaje NOCT
- Intensidad STC
- Intensidad NOCT
- Temperatura Panel.

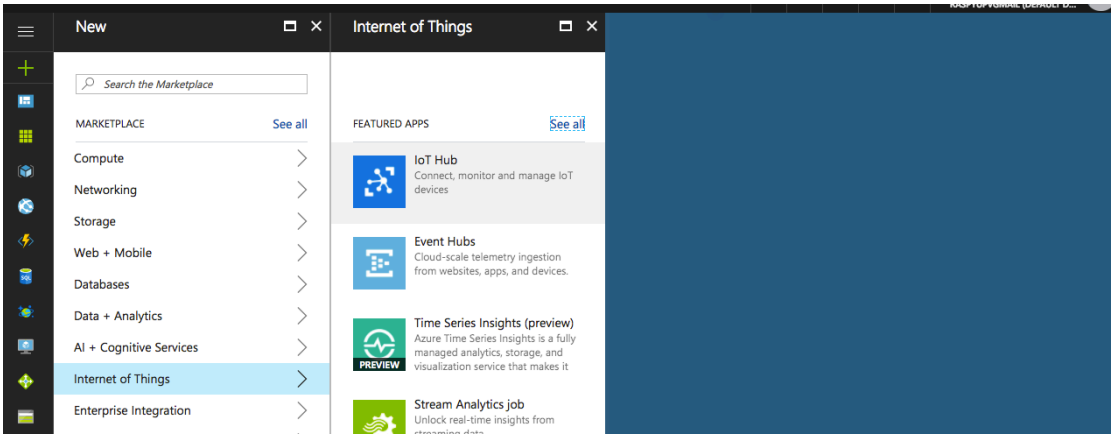
La Raspberry Pi usada tiene las siguientes características:

- Raspberry Pi 3
- Raspbian Jessie 8.0
- Linux 4.4.38-v7+
- Python 2.7

El primero que se debe hacer es crear una cuenta Microsoft en el siguiente link https://www.microsoft.com/es-es/account.

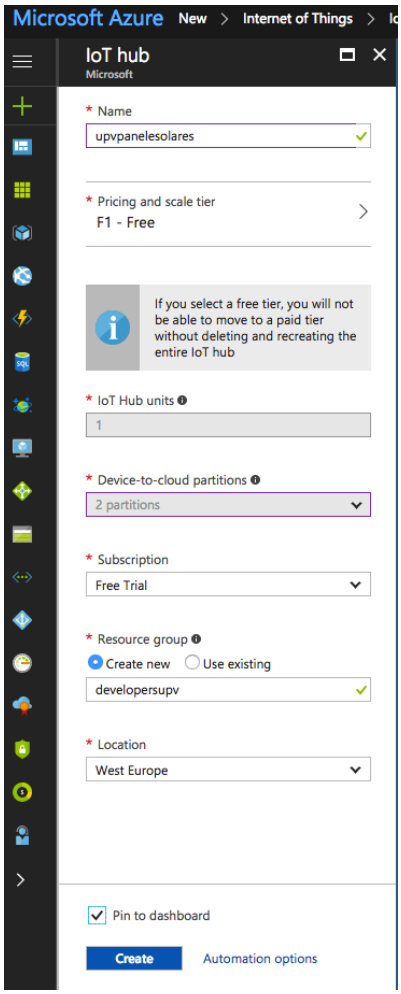
Una vez creada la cuenta accedemos a los servicios en la nube de Microsoft Azure en el siguiente link Azure Portal.

Una vez en el portal web debemos crear el servicio ***IoT Hub***. En la parte superior derecha vamos a ***New >> Internet of Things >> IoT Hub***

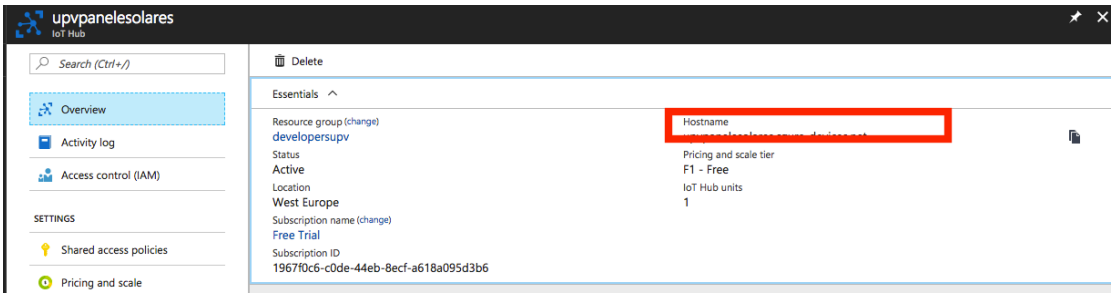


Se abrirá un cuadro de dialogo e ingresamos la siguiente información:

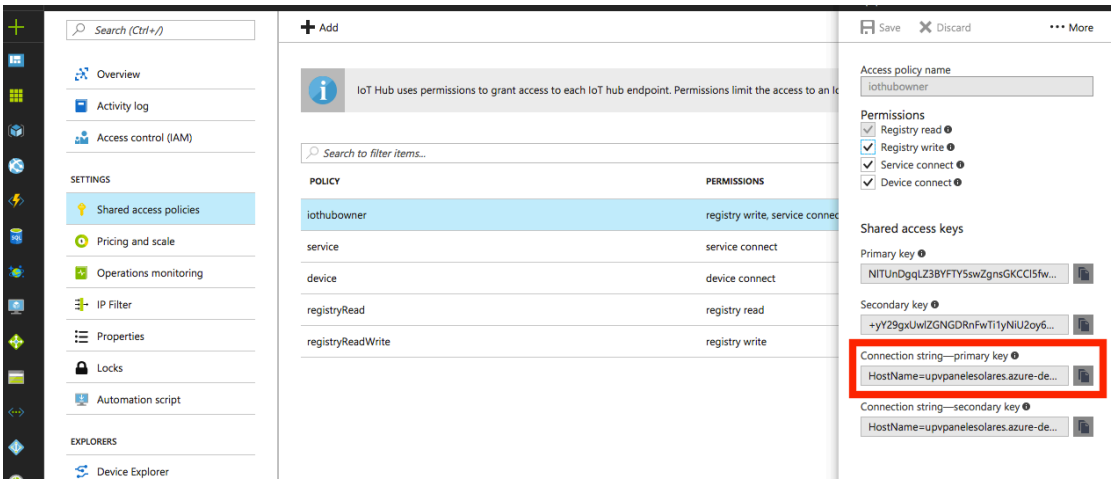
Name: Un nombre cualquiera con el que se identificará al servicio.
Pricing and Scale tier: Seleccionar el modelo F1 - Free
Resource Group: Crear un grupo nuevo. Este grupo debe ser común para todos los servicios en los se que se desee compartir información.
Location: Elegir el lugar físico disponible donde estarán presentes los servicios. Se recomienda una lugar cercano.
Pin To Dashboard: Anclar el servicio en la pagina principal para rápido acceso.



Damos clic en ***Create*** y esperemos a que el servicio se instale y este listo para ser usado. Ahora es necesario gestionar los permisos del ***IoT Hub*** para que los dispositivos se pueden autenticar con el mismo. Para ello en el ***Dashboard*** damos clic en el ***IoT Hub*** anteriormente creado. Un vez lanzado este servicio damos clic en ***Overview*** para consultar el ***hostname*** asignado. Copiar este nombre para futuras referencias.



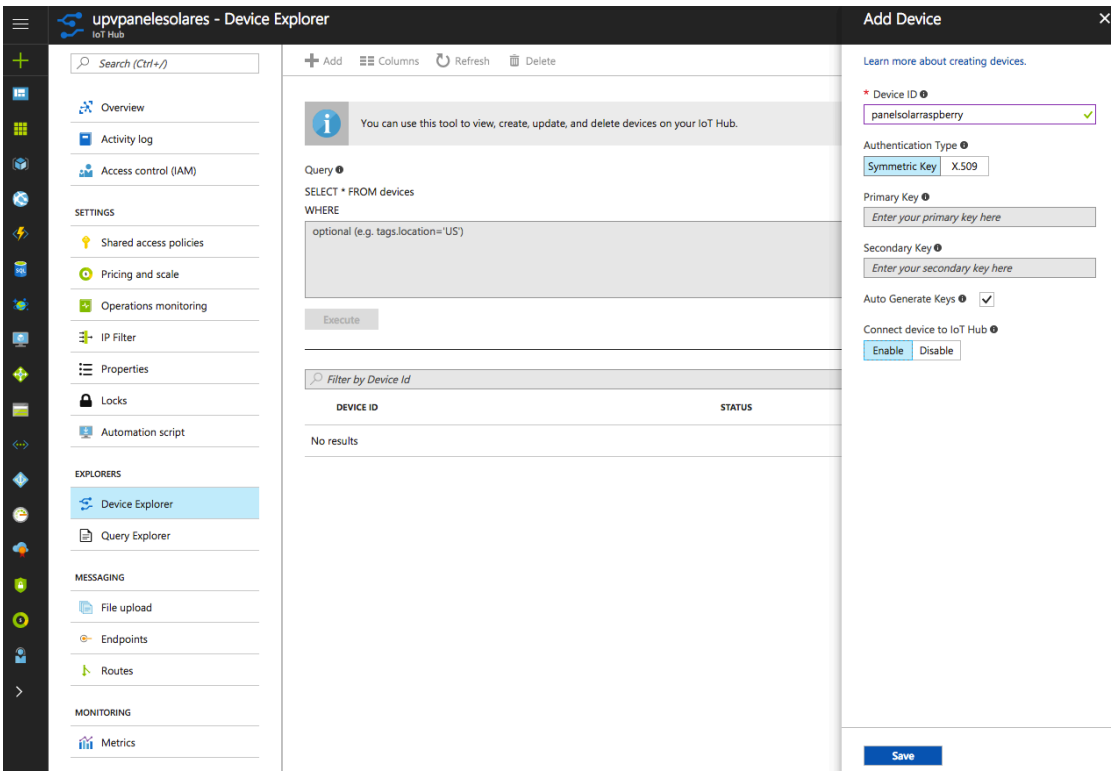
Ahora vamos a ***Shared Access Policies >> Policy >> iothubowner >> Connection string – primary key***. De este ultimo hacemos una copia para futuras referencias.



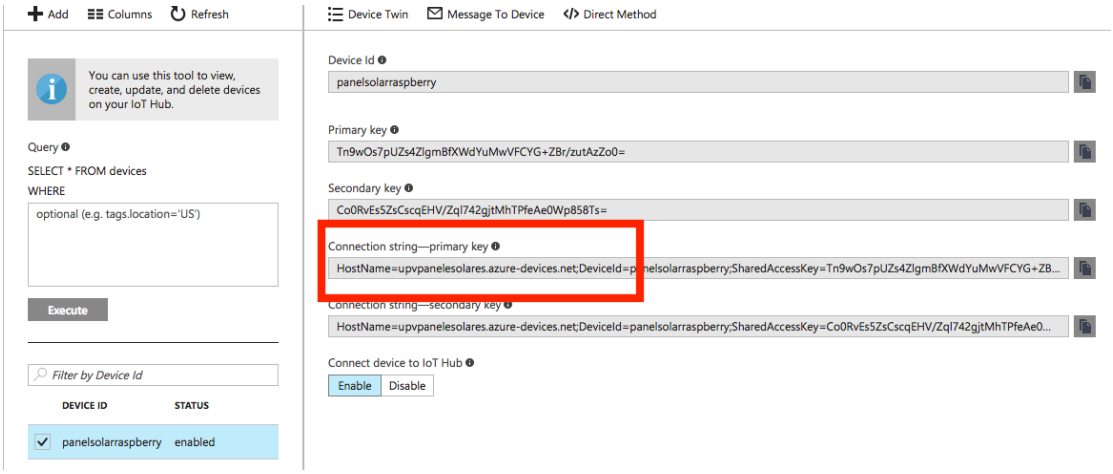
Ahora procedemos a registrar el dispositivo que se conectará al ***IoT Hub***. Vamos a ***Device Explorer*** y damos clic en ***Add*** e ingresamos la siguiente información

Device ID: Un identificador del dispositivo.
Authentication Type: Seleccionamos Symmetric Key
Auto Generate Keys: Habilitamos esta opción
Connect device to IoT Hub: Seleccionamos Enable.

Damos clic en ***Save***.



Después damos clic nuevamente en ***Device Explorer*** y seleccionamos el dispositivo anteriormente creado. Copiamos el ***Connection string-primary key*** para futuras referencias.



Con esta información ahora podemos configurar la Raspberry para que envié información hacia Azure. En la Raspberry Pi ingresamos las siguientes ordenes:

$ cd
$ sudo apt-get update
$ sudo apt-get install g++ make cmake git libcurl4-openssl-dev libssl-dev uuid-dev
$ git clone --recursive https://github.com/Azure-Samples/iot-remote-monitoring-c-raspberrypi-getstartedkit.git

En el repositorio https://github.com/edisonupv/iot-servicesexamples existe un archivo llamado ***panelsolarAZURE.c*** el cual esta listo para enviar la información del panel solar hacia Azure. Solo debemos cambiar las siguientes líneas:

```
static const char* deviceId = "********************";    //Your device ID
static const char* connectionString = "**********";    //Connection String of your device
```

Esta información es la que anteriormente se pidió realizar copias. Una vez este archivo esté modificado lo movemos al directorio donde se descargó las librerías de Azure IoT para Raspberry Pi.

$ mv panelsolarAZURE.c ~/iot-remote-monitoring-c-raspberrypi-getstartedkit/simulator/remote_monitoring/remote_monitoring.c

Después ingresos esta orden

$ chmod +x ~/iot-remote-monitoring-c-raspberrypi-getstartedkit/simulator/build.sh

Compilamos el archivo

$ ~/iot-remote-monitoring-c-raspberrypi-getstartedkit/simulator/build.sh

Si la compilación no tuvo inconvenientes debería aparecer algo similar a esta imagen:

```
[ 98%] Built target websockets_sample
[ 98%] Building C object azure-iot-sdk-c/iothub_service_client/CMakeFiles/iothub_service_client.dir/src/iothub_devicetwin.c.o
[ 98%] Building C object azure-iot-sdk-c/iothub_service_client/CMakeFiles/iothub_service_client.dir/src/iothub_devicemethod.c.o
[ 99%] Building C object azure-iot-sdk-c/iothub_service_client/CMakeFiles/iothub_service_client.dir/src/iothub_service_client_auth.c.o
[ 99%] Building C object azure-iot-sdk-c/iothub_service_client/CMakeFiles/iothub_service_client.dir/src/iothub_sc_version.c.o
[100%] Building C object azure-iot-sdk-c/iothub_service_client/CMakeFiles/iothub_service_client.dir/__/iothub_client/src/iothub_message.c.o
[100%] Linking C static library libiothub_service_client.a
[100%] Built target iothub_service_client
UpdateCTestConfiguration  from :/home/pi/cmake/DartConfiguration.tcl
UpdateCTestConfiguration  from :/home/pi/cmake/DartConfiguration.tcl
Test project /home/pi/cmake
Constructing a list of tests
Checking test dependency graph...
Checking test dependency graph end
No tests were found!!!
```

Ahora ejecutamos el archivo ejecutable resultado de la compilación

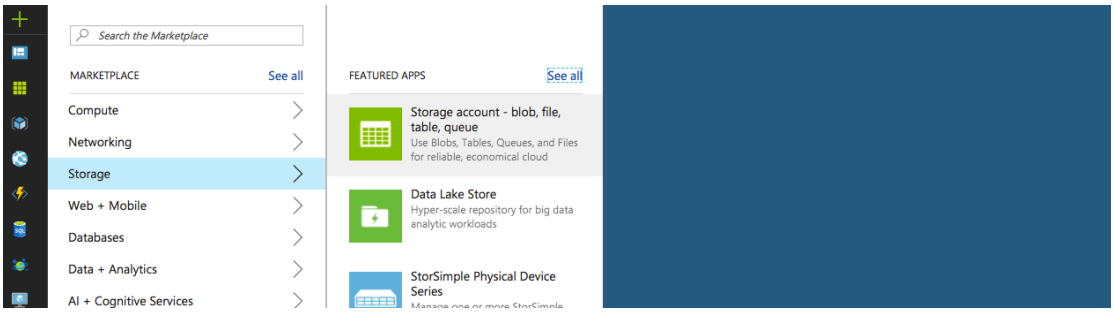
$ sudo ~/cmake/remote_monitoring/remote_monitoring

Debería aparecer una imagen similar a esta:

```
pi@raspyITACA:~ $ sudo ~/cmake/remote_monitoring/remote_monitoring
Info: IoT Hub SDK for C, version 1.1.13
Sending sensor value: {"DeviceID": "panelsolarraspberry","voltajeSTC01" : 26.67,"voltajeNOCT01" : 18.27,"intensidadSTC01" : 28.59,"intensidadNOCT01" : 16.76,"temppanel01" : 28.92 }
 159
IoTHubClient accepted the message for delivery
IoTHub: reported properties delivered with status_code = 204
pi@raspyITACA:~ $
```

Ahora se puede usar la herramienta ***crontab*** de Linux para programar el ejecución de este script y que la Raspberry envié información de manera automática.

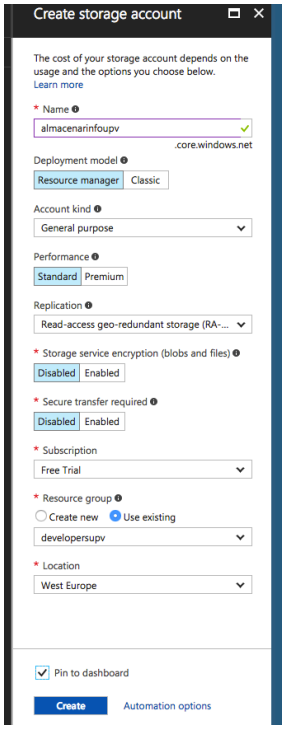
Una vez que la Raspberry Pi esta enviando información de manera constante a Azure IoT es necesario almacenar la información del dispositivo en una base de datos. Para ello vamos a ***New >> Storage >> Storage Account***



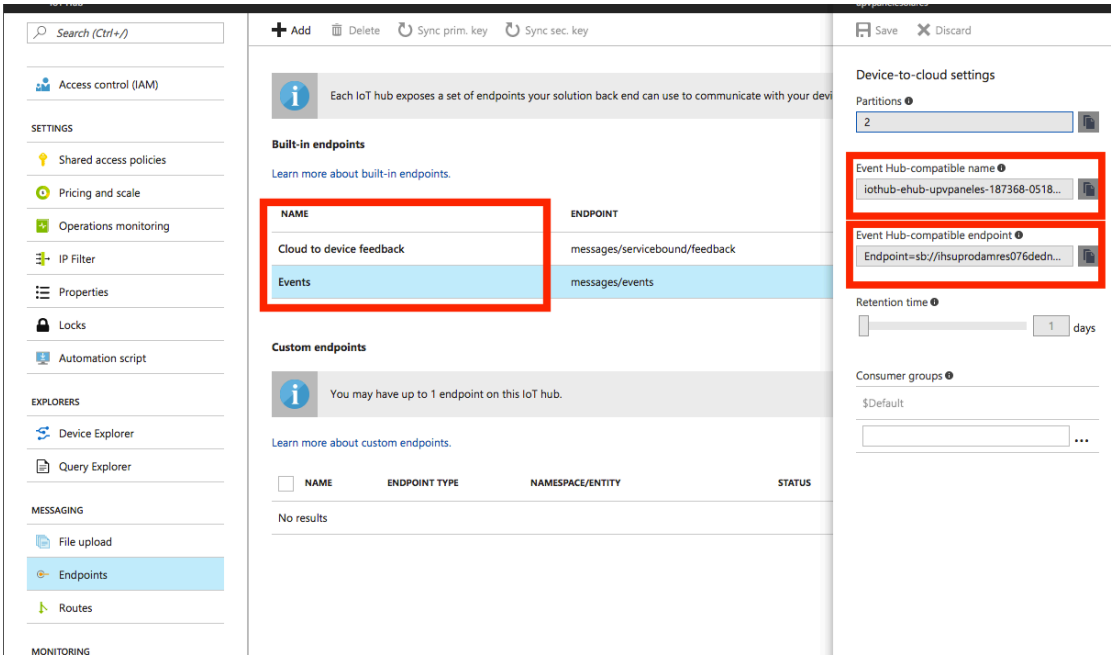
En el cuadro de dialogo que se presenta ingresar la siguiente información:

Name: Nombre para el servicio de almacenamiento en la nube.
Resource Group: Debe ser el mismo ya creado en el servicio IoT Hub
Pin to DashBoard: Habilitar esta opción.

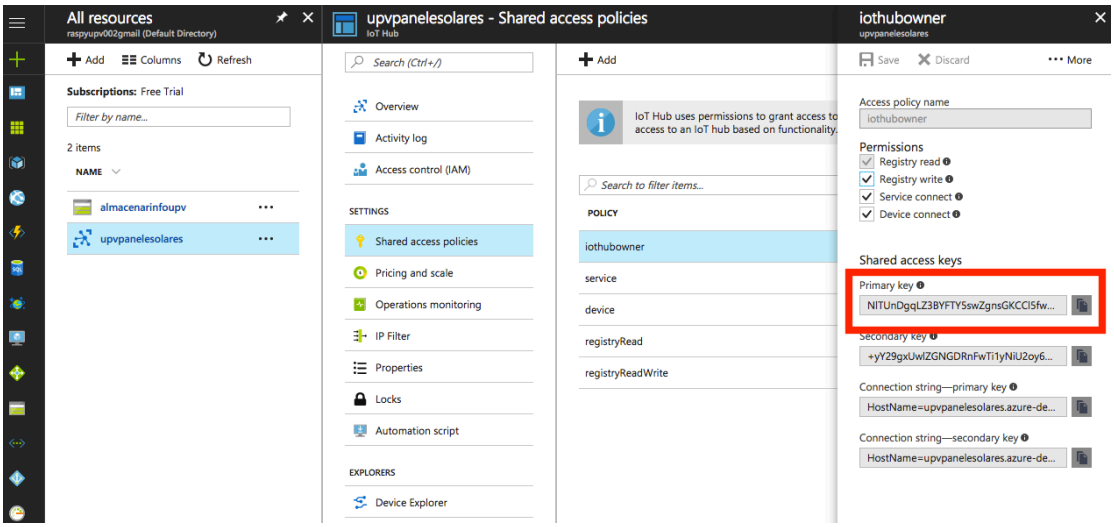
Damos clic en ***Create*** y esperamos a que el servicio termine de instalarse y configurarse.



Ahora debemos "conectar" los eventos que se producen cuando un mensaje de algún dispositivo llega al ***IoT Hub*** con el servicio de almacenamiento de Azure. Para ello vamos a ***IoT Hub >> Messaging >> EndPoints >> Built-in endposints >> Events***. Copiamos los valores de ***Event hub-compatible endpoint*** y ***Event hub-compatible name*** para futuras referencias.



Ahora vamos a ***IoT Hub >> Setting >> Shared Access Policies >> iothubowner***. Y copiamos el valor de ***Primary key*** para futuras referencias.

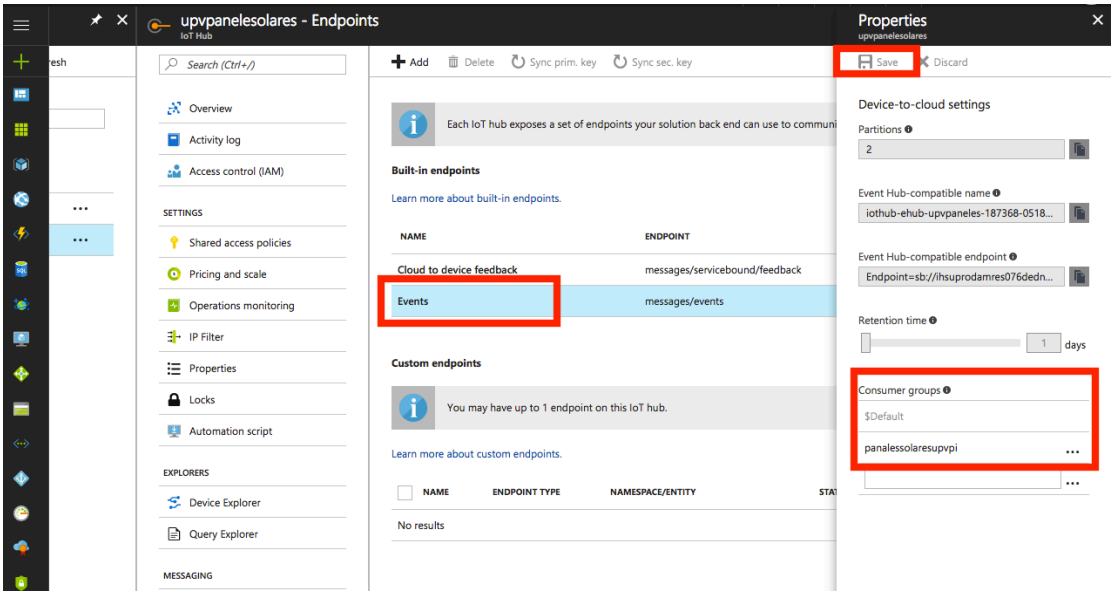


Debemos crear un cadena de texto que representa el ***IoT Hub endpoints*** con el siguiente formato:

Endpoint=<Event Hub-compatible endpoint>;SharedAccessKeyName=iothubowner;SharedAccessKey=<Primary key>

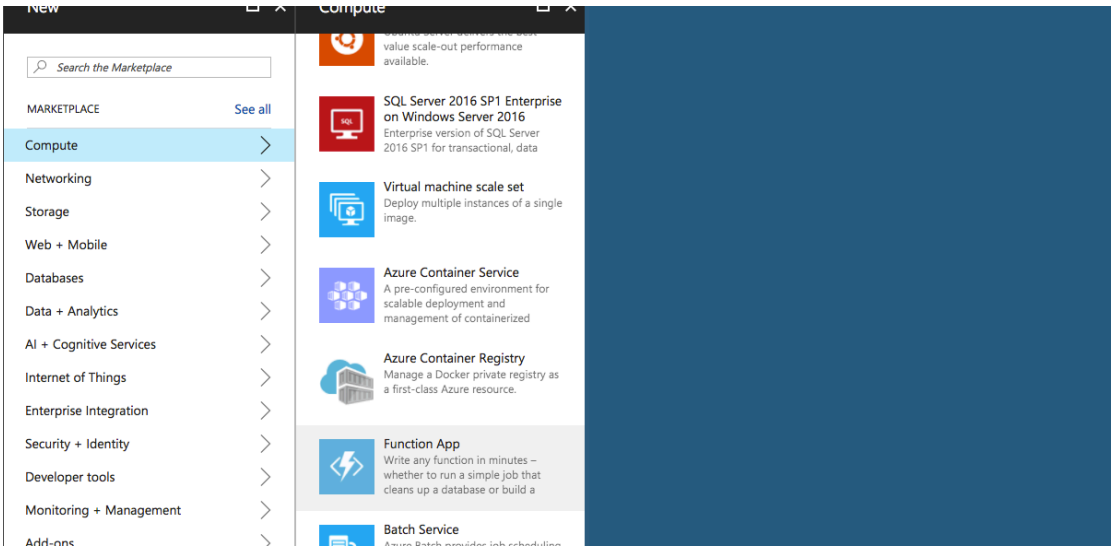
Donde el ***<Event Hub-compatible endpoint>*** y el ***<Primary key>*** es la información que anteriormente se ha pedido copiar.

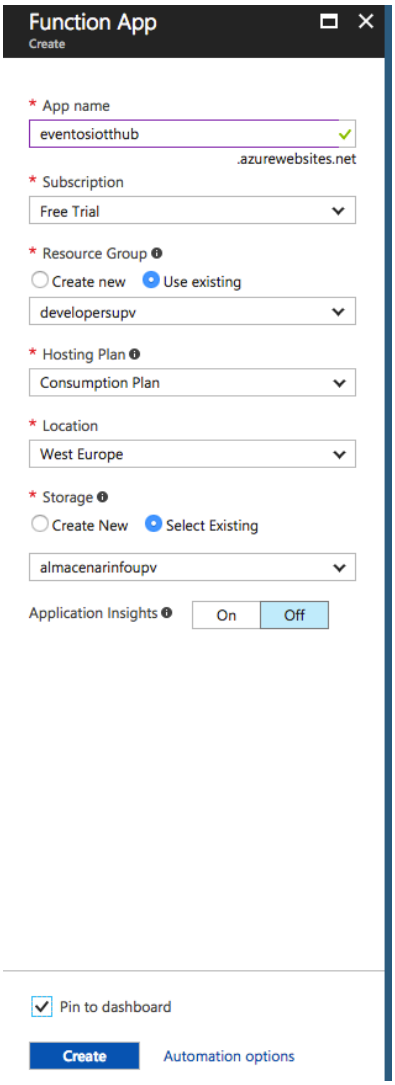
Ahora vamos a ***IoT Hub >> Messaging >> Endpoints >> Built-in endposints >> Events***. Aquí vamos al panel de ***Properties*** y creamos un ***Consumer Group***. Copiamos el nombre de este grupo para futuras referencias. Damos clic en ***Save***



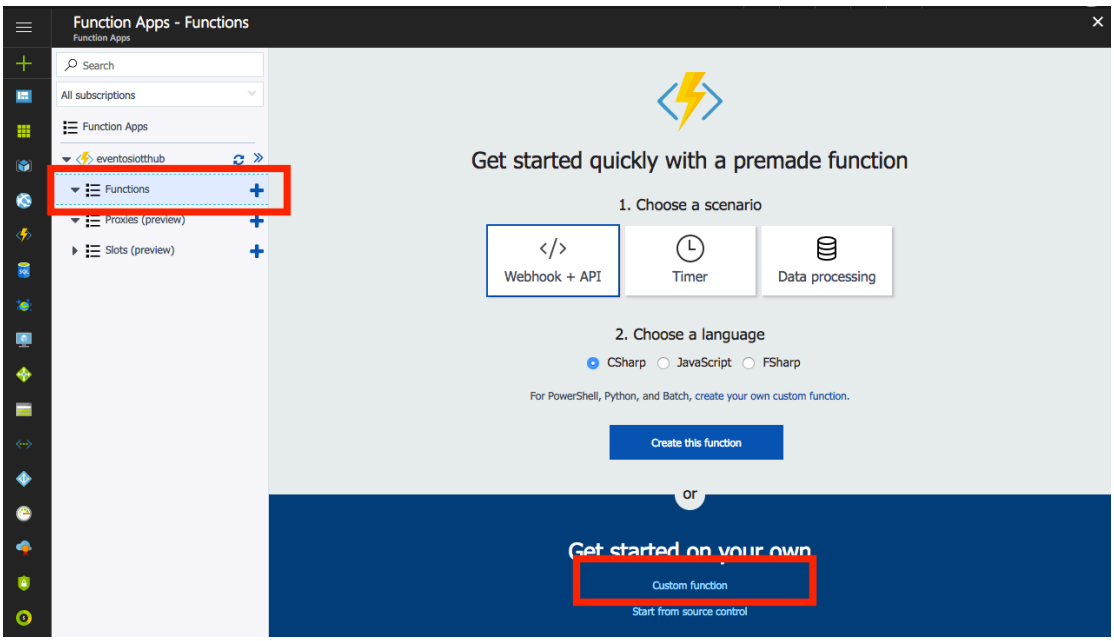
Ahora es necesario hacer uso del servicio ***Fuction App*** para que cada vez que ocurra un evento de recepción de mensajes de algún dispositivos en el ***IoT Hub***, este mensaje se almacene en el servicio de ***Azure Storage***. Para ello vamos ***New >> Compute >> Fuction App*** e ingresamos la siguiente información:

App Name: El nombre para el servicio
Resource Group: El mismo al que pertenecen IoT Hub y el Storage Azure
Storage Account: La creada anteriormente
Pin to Dashboard: Habilitar esta opción.

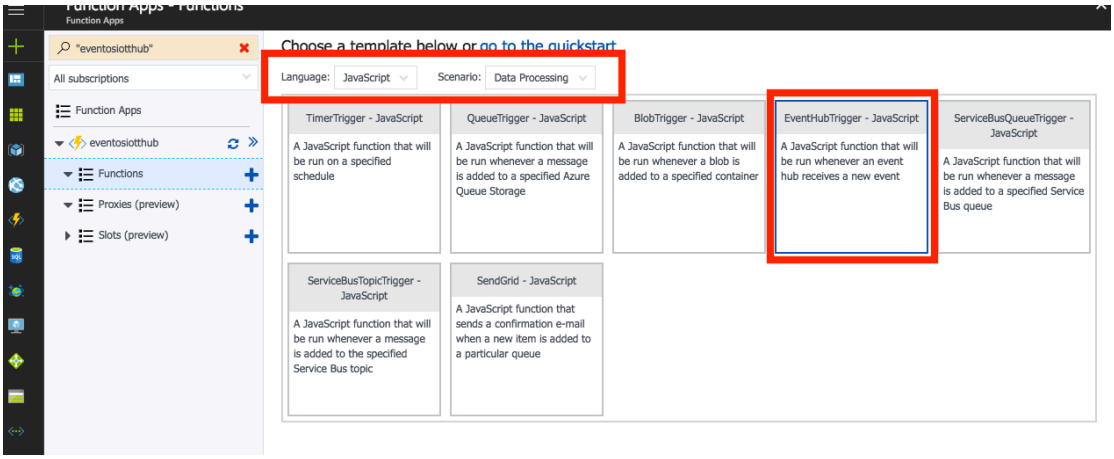




Damos clic en ***Create*** y esperamos a que el servicio se instale y se configure. Una vez configurado el servicio lo abrimos y nos vamos a ***Fuctions >> New >> Custom Fuction***

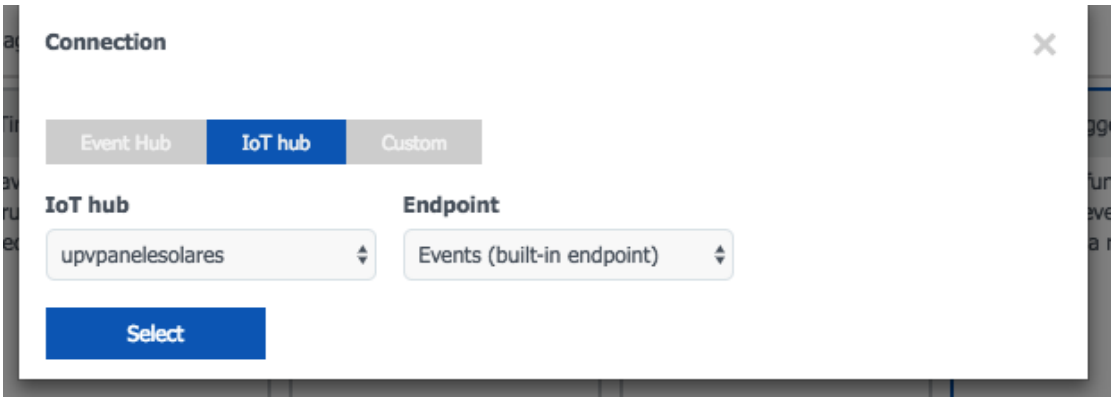


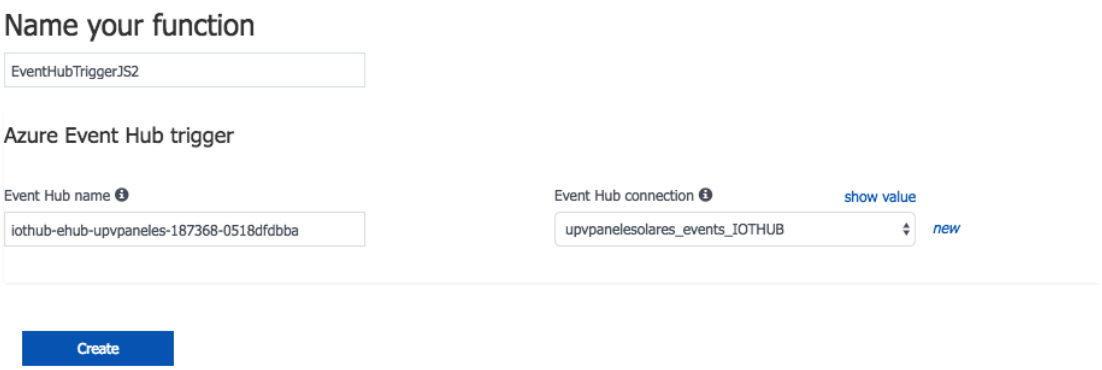
En la nueva pagina que se abre elegimos ***Language : Java Script >> Scenario: Data Processing >> EventHubTrigger-JavaScript***



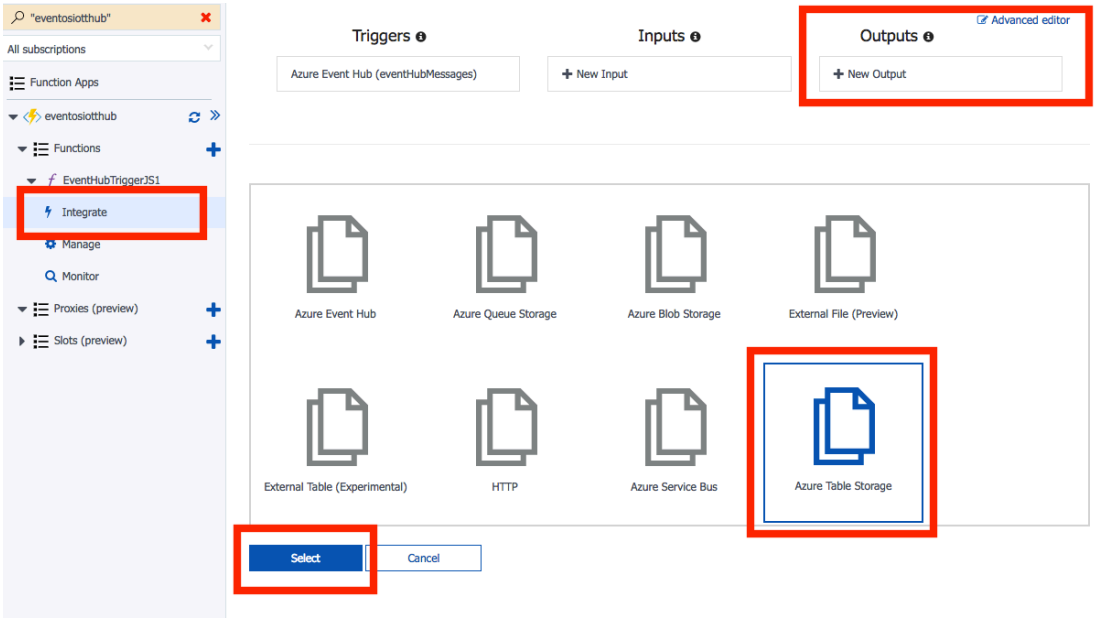
Ahora ingresamos la siguiente información:

Name your Fuction: Un nombre para la función.
Event Hub name: El *event hub-compatible name* anteriormente copidado
Event Hub Connection: Aquí hacemos clic en ***New >> IoT Hub*** y lo configuramos con el *IoT Hub* creado anteriormente y como *Endpoint* seleecionamos *<<Events: (built-in endpoints)>>*. Después clic en *Select* y *Create*



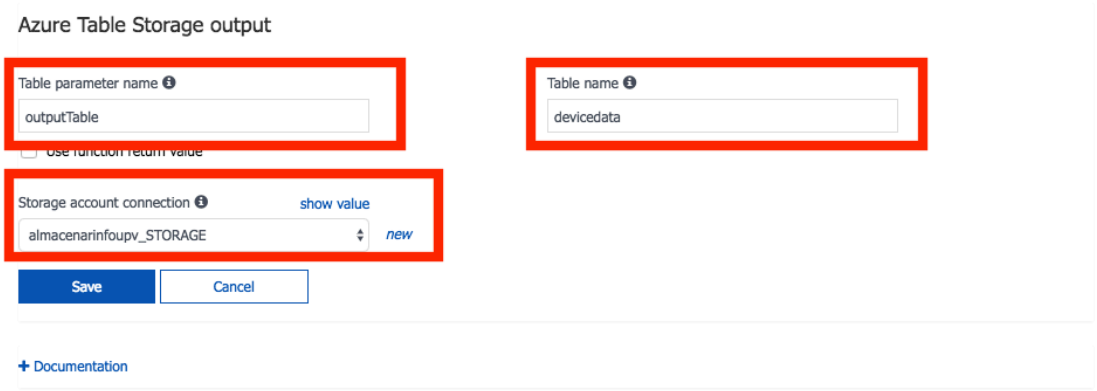


Ahora vamos el sub menú ***Integrate >> New Output >> Azure Table Storage >> Select***

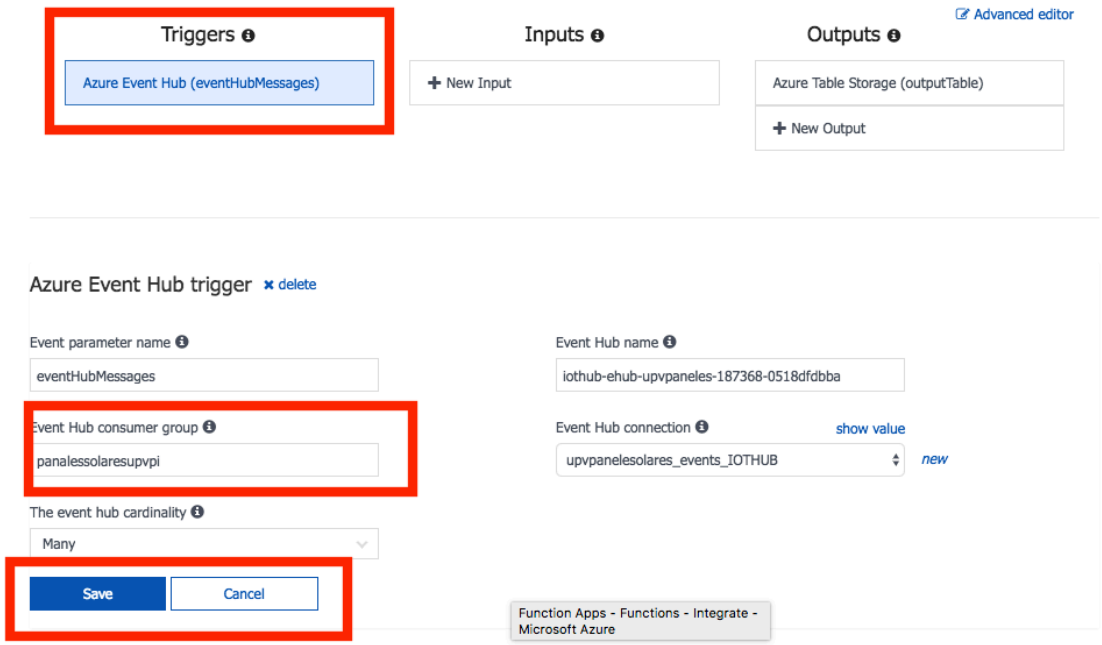


Ingresamos la siguiente información

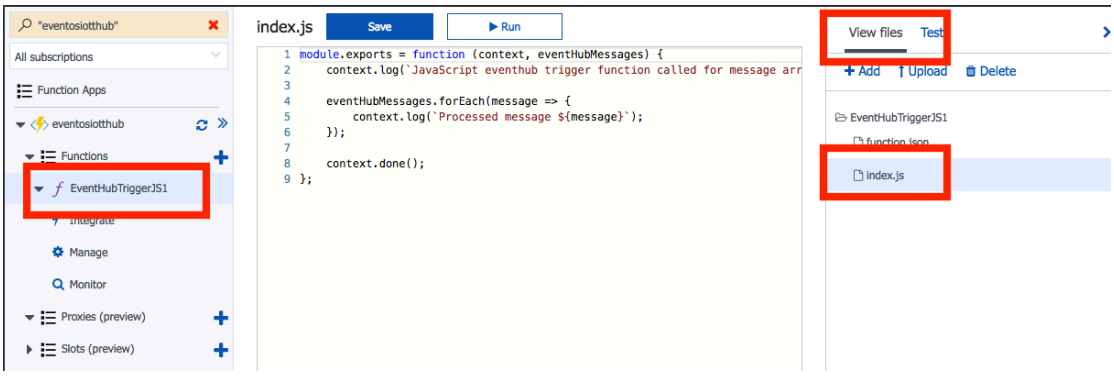
Table parameter name: Nombrar como *outputTable*, el cual será el nombre usado en la función Java Script
Table name: Nombrar como *devicedata*
Storage account connection: En *New* elegir el almacenamiento anteriormente creado.



Damos clic en ***Save***. Ahora vamos a ***Triggers >> Azure Event Hub (eventHUbMessages) >> Event Hub consumer group*** donde ingresamos el nombre del *consumer group* anteriormente creado. Después damos clic en Save



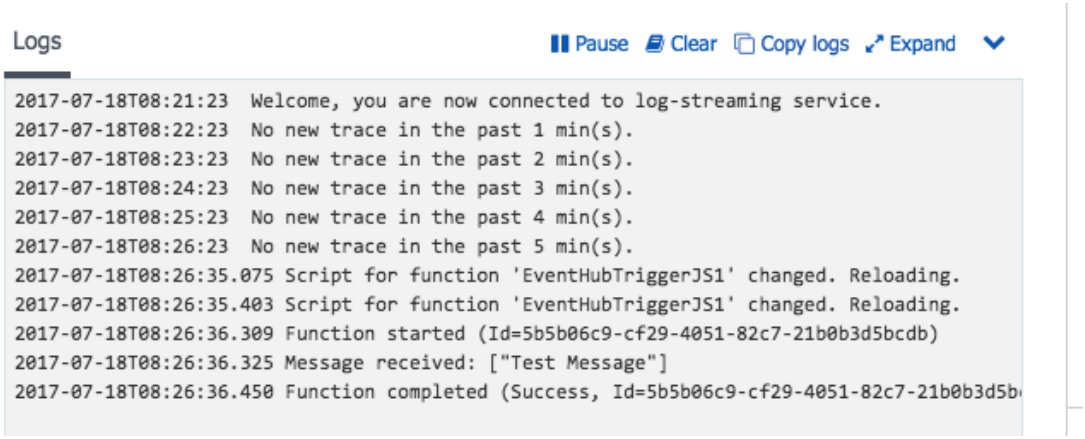
Después vamos al menú izquierdo donde esta la función que se ha creado y cambiamos a ***View files >> index.js***



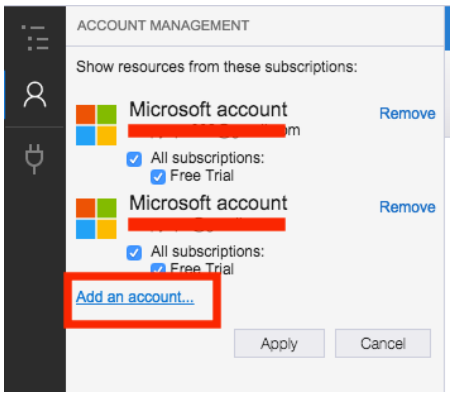
El código de dicho archivo lo reemplazamos con la siguiente función:

```
'use strict';
// This function is triggered each time a message is received in the IoT hub. // The
message payload is persisted in an Azure storage table
module.exports = function (context, iotHubMessage) { context.log('Message received: ' +
JSON.stringify(iotHubMessage)); var date = Date.now();
var partitionKey = Math.floor(date / (24 * 60 * 60 * 1000)) + ''; var rowKey = date +
'';
context.bindings.outputTable = {
"partitionKey": partitionKey,
"rowKey": rowKey,
"message": JSON.stringify(iotHubMessage)
};
context.done();
};
```

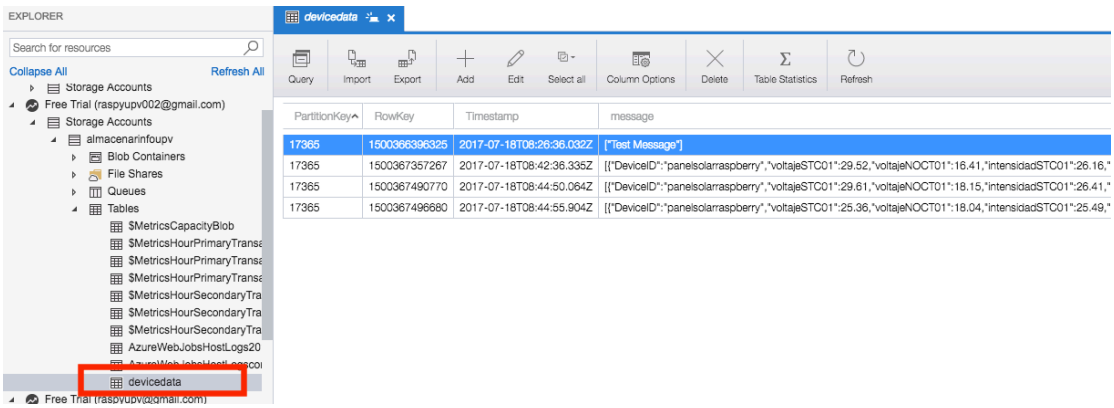
Damos clic en ***Save and run***. En el terminal de Logs debería aparecer el siguiente mensaje.



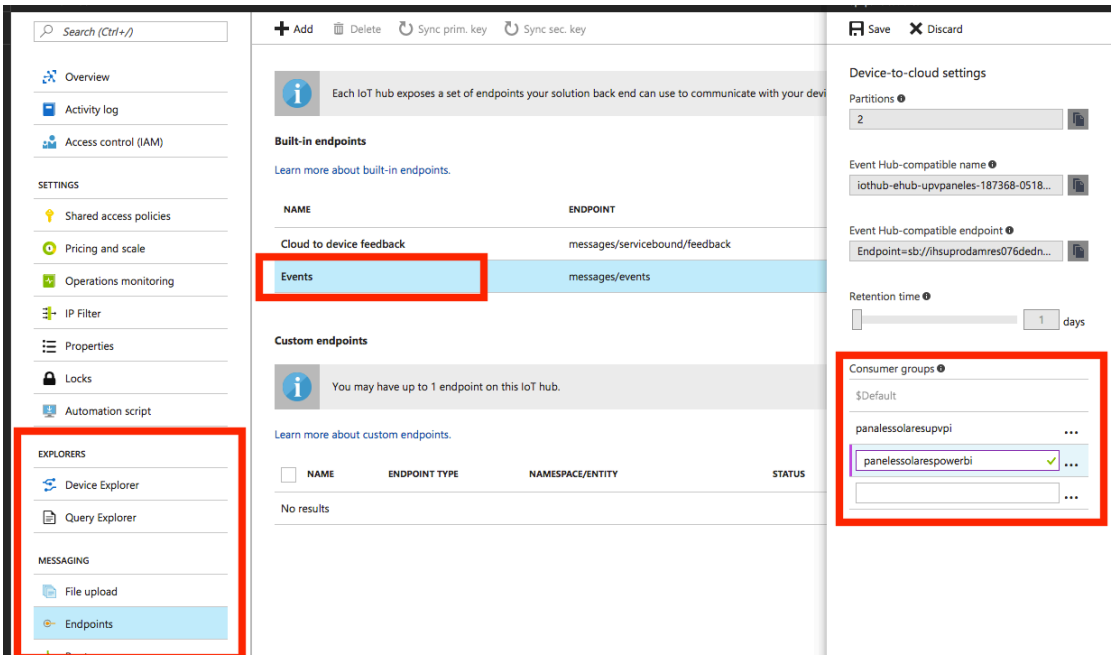
Ahora ya podemos ver la información de los sensores en el base de datos de Azure. Para ello descargamos la aplicación [Azure Storage Explorer](). Dentro de la aplicación añadimos nuestra cuenta de Microsoft Azure.



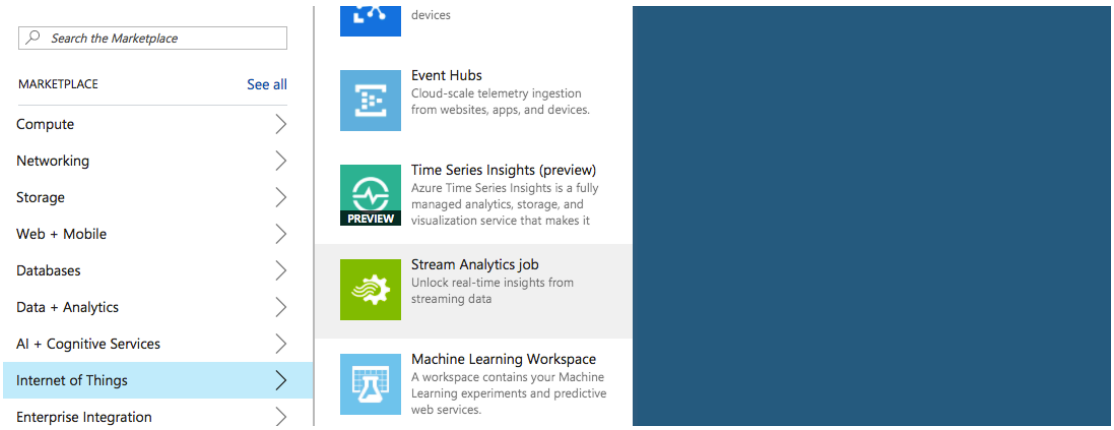
Después vamos a ***Explorer >> Nuestra cuenta de Azure >> Storage Account >> Tables >> devicedata*** donde estará la tabla con la información de nuestros sensores.



Una vez almacenada la información en la base de datos de Azure Storage es posible exportar esta información hacia una aplicación externa propia de Microsoft para la visualización de los datos en tiempo real. Para ello añadimos un nuevo ***consumer group*** en ***IoT Hub >> Explorers >> Endpoints >> Events >> Consumer groups***. Una vez realizado esto damos clic en ***Save***.



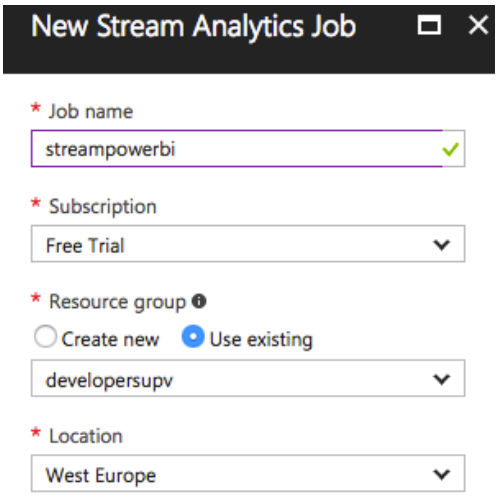
Ahora debemos implementar el servicio de *Stream Analytics Job* en ***New >> Internet of Things >> Stream Analytics Job***



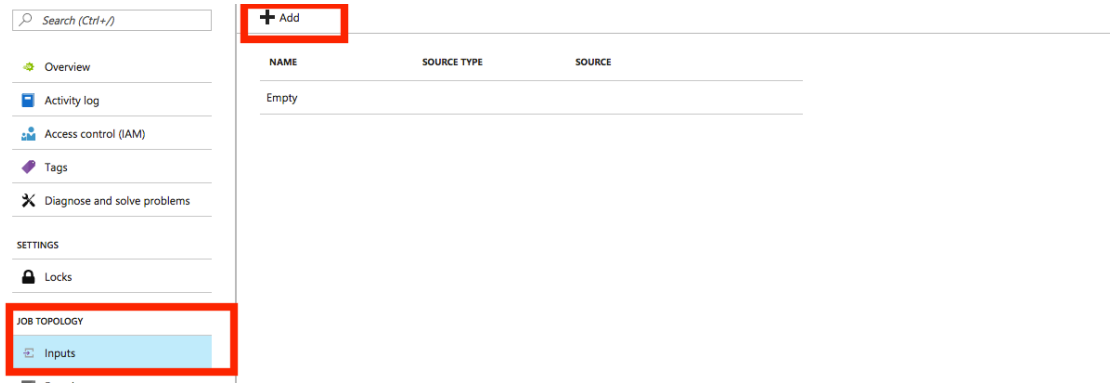
Ingresamos la siguiente información:

Job name: Un nombre para el servicio
Resource Group: El mismo grupo común de los demás servicios
Location: La misma locación de los demás servicios.
Pin to dashboard: Habilitar esta opción.

Damos clic en ***Create***.



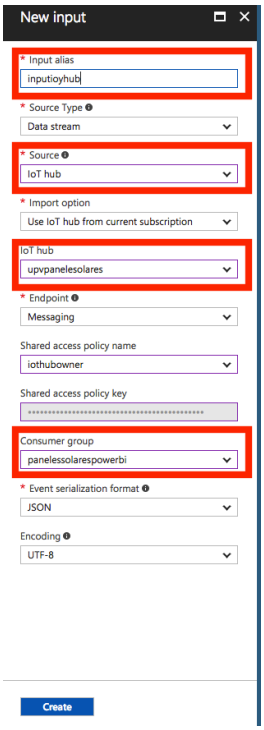
Desde el *Dashboard* lanzamos el servicio *Stream Analytics* anteriormente creado. Ahora vamos a ***Job Topology >> Inputs*** damos clic en ***+ Add***



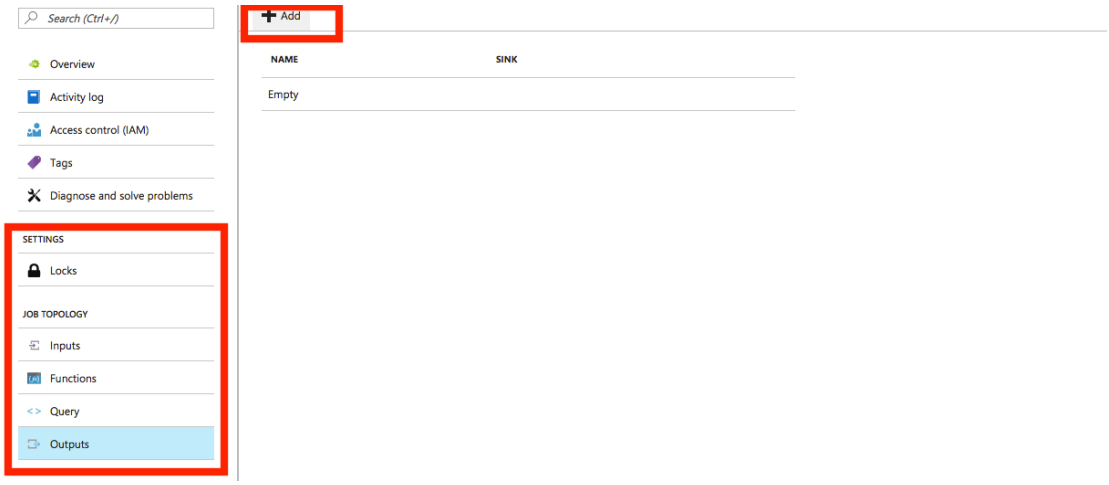
Ingresamos la siguiente información

Input Alias: Un nombre para esta entrada
Source: Seleccionamos IoT Hub
Consumer group: El consumer group anteriormente creado.

Damos clic en ***Create***



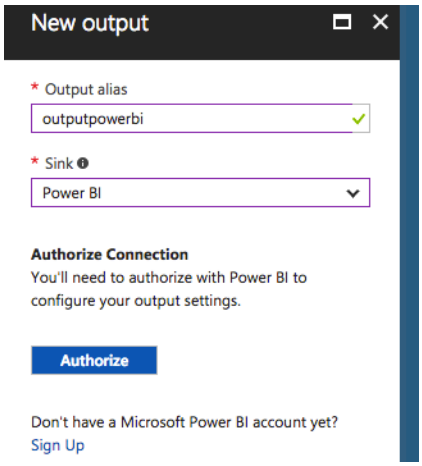
Ahora debemos añadir un output que se conectara a la aplicación *PowerBI*. En el siguiente link podemos crear una cuenta gratuita en Power BI. Una vez lista la cuenta vamos a ***Job Topology >> Outputs >> + Add***



Ingresamos la siguiente información

OutputAlias: Un nombre para esta salida.
Sink: Seleccionamos Power Bi

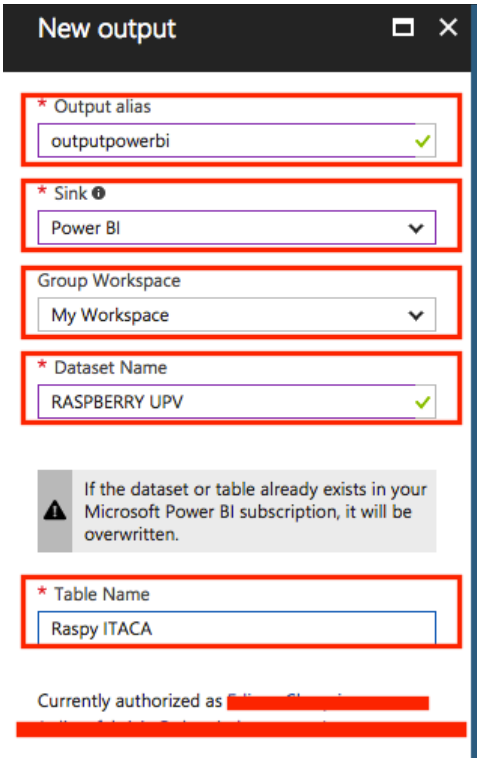
Damos clic en ***Authorize***



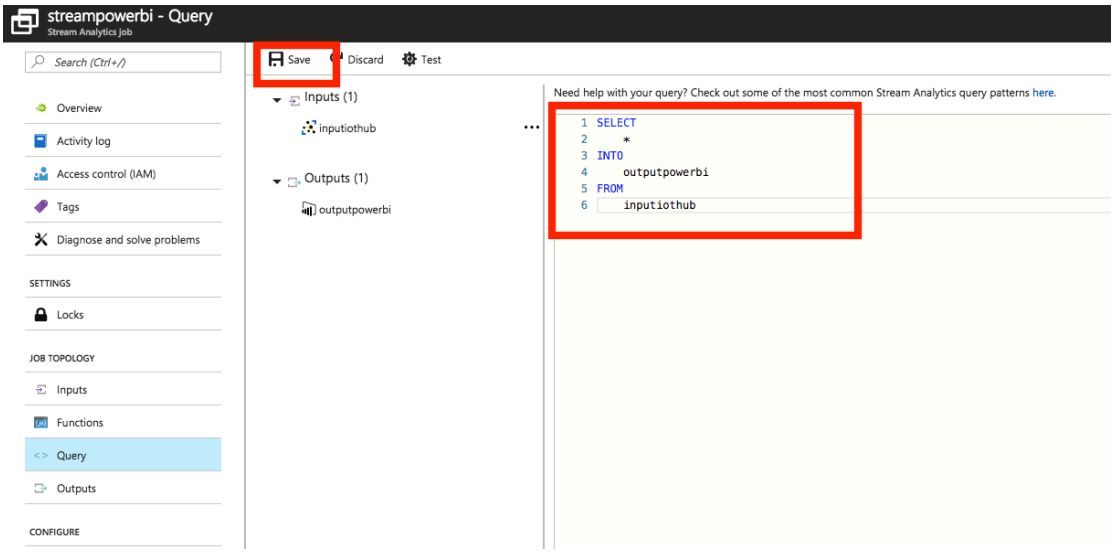
Se abrirá una página en la que debemos autenticarnos con nuestra cuenta de *Power BI*. Después Ingresamos la siguiente información:

Group Workspace: Seleccionamos un *Workspace*
Dataset Name: Un nombre para el *dataset*
Table Name: Un nombre para el tabla.

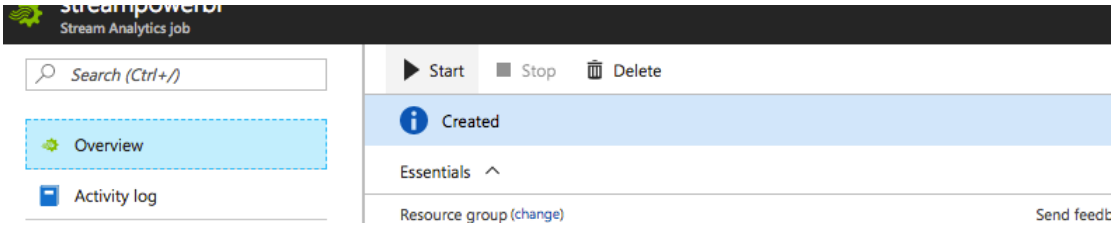
Damos clic en ***Create***



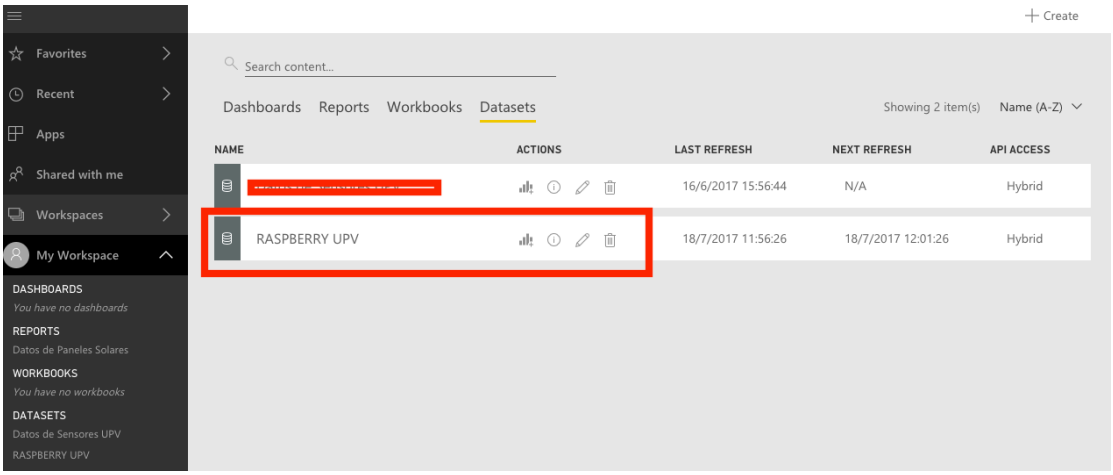
Ahora debemos configurar la manera en que se manejaran las consultas al *IoT Hub*. Para ello vamos a ***Job Tolology >> Query*** y remplazamos *[YourInputAlias]* con el anteriormente creado. Aplicar lo mismo para *[YourOutputAlias].* Damos clic en ***Save***



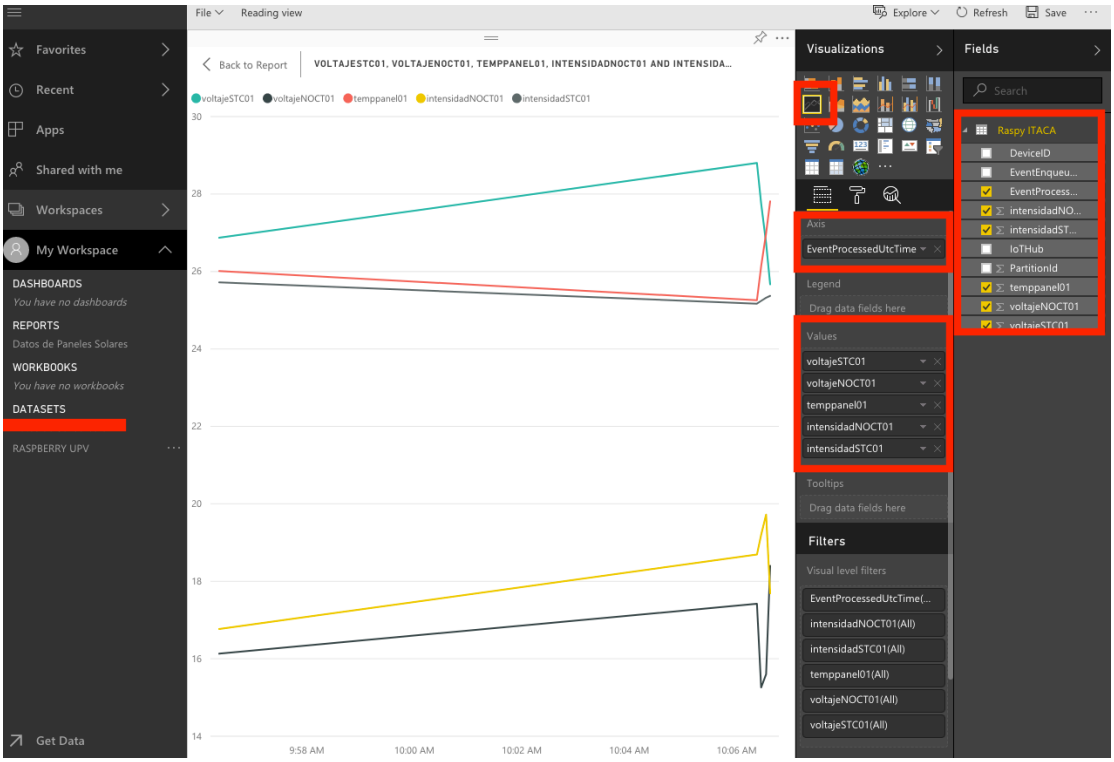
Ahora vamos a ***Overview >> Start >> Now >> Start***. El servicio debería pasar de *Start* a *Running*.



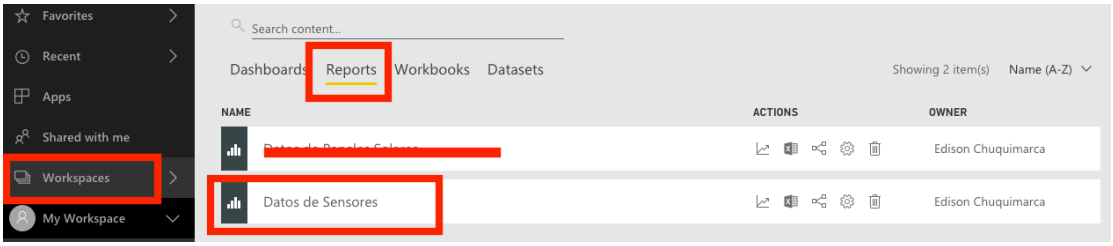
Asegurarse que el script dentro de la Raspberry Pi continúa enviando información a Azure IoT. Ahora vamos a la aplicación de ***Power BI >> Workspace >> Dataset*** donde se debería mostrar el dataset anteriormente creado

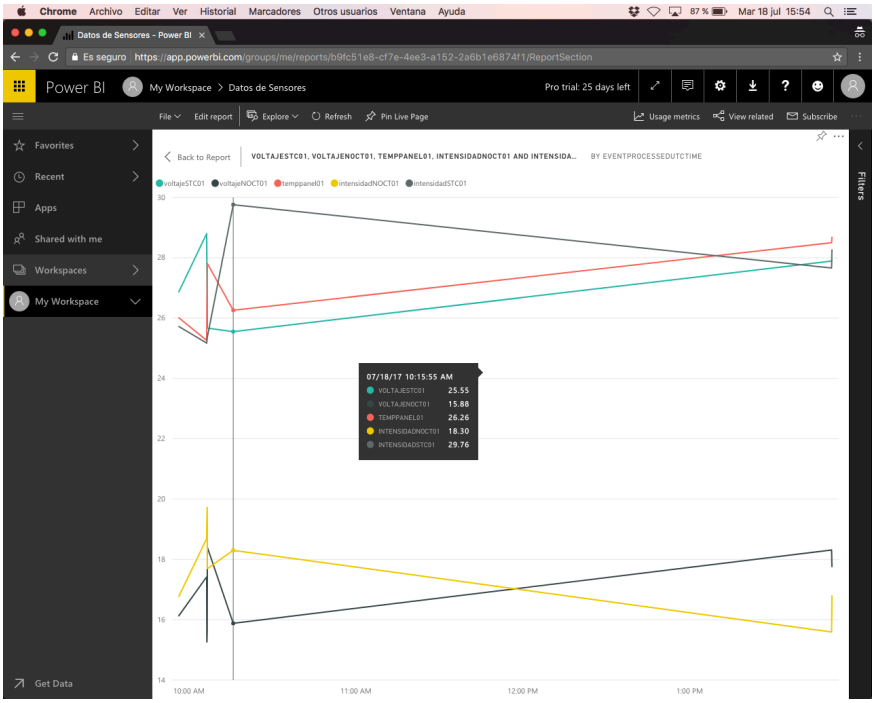


Damos clic en el primer icono debajo de *Actions* denominado ***Create a Report***. En la nueva página que se abre, en la parte derecha superior estará una tabla con cada uno de los valores de los sensores que esta enviando la Raspberry Pi. En la parte de ***Visulizations*** elegimos la tabla ***Line Chart*** y en *Values* arrastramos cada uno de los campos correspondientes a las variables. En Axis arrastramos el campo de ***EventProcessedUtcTim***e. Al final debería quedar algo similar a esto.



Damos clic en ***Save*** y lo guardamos como Reporte. En la sección de ***Workspaces >> Reports*** estará ubicado el reporte para ser consultado.





En este reporte estarán los datos de los sensores en tiempo real. Ahora con estas conocimientos previos es posible añadir nuevos y más dispositivos y almacenar la información en Azure y que sea posible visualizar en Power BI.

En los siguientes enlaces se puede encontrar información útil de cómo desarrollar soluciones IoT para Azure usando la Raspberry Pi.

https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-devguide-sdks
https://docs.microsoft.com/en-us/azure/iot-hub/iot-hub-raspberry-pi-kit-c-get-started