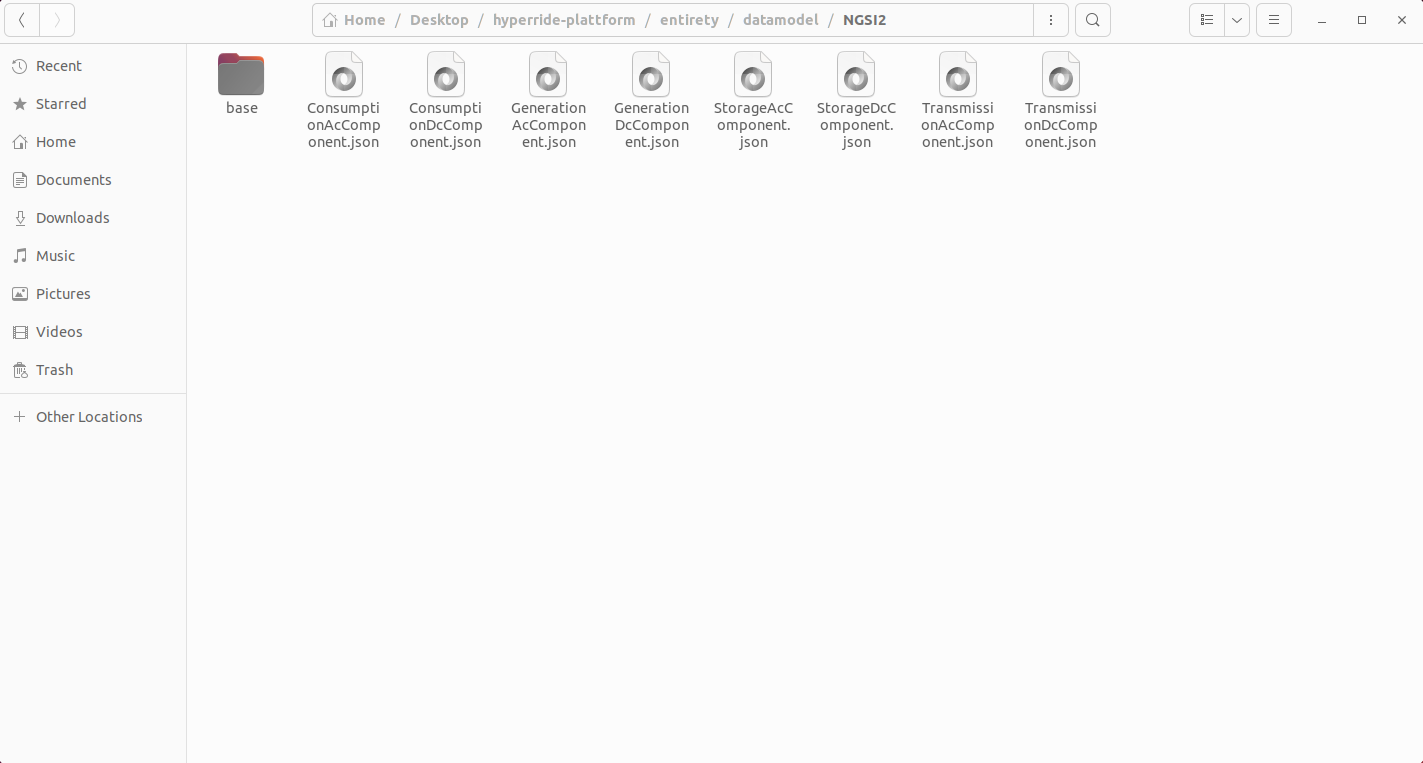
**New Ontology**

**Entirety Part**

1. **Add Components:**

According to the new ontology we had 8 components and they were add in the NGSI2 folder. The directory to the folder is “entirety --> datamodel --> NGSI2”. The components were added as .json files in the proper syntax.

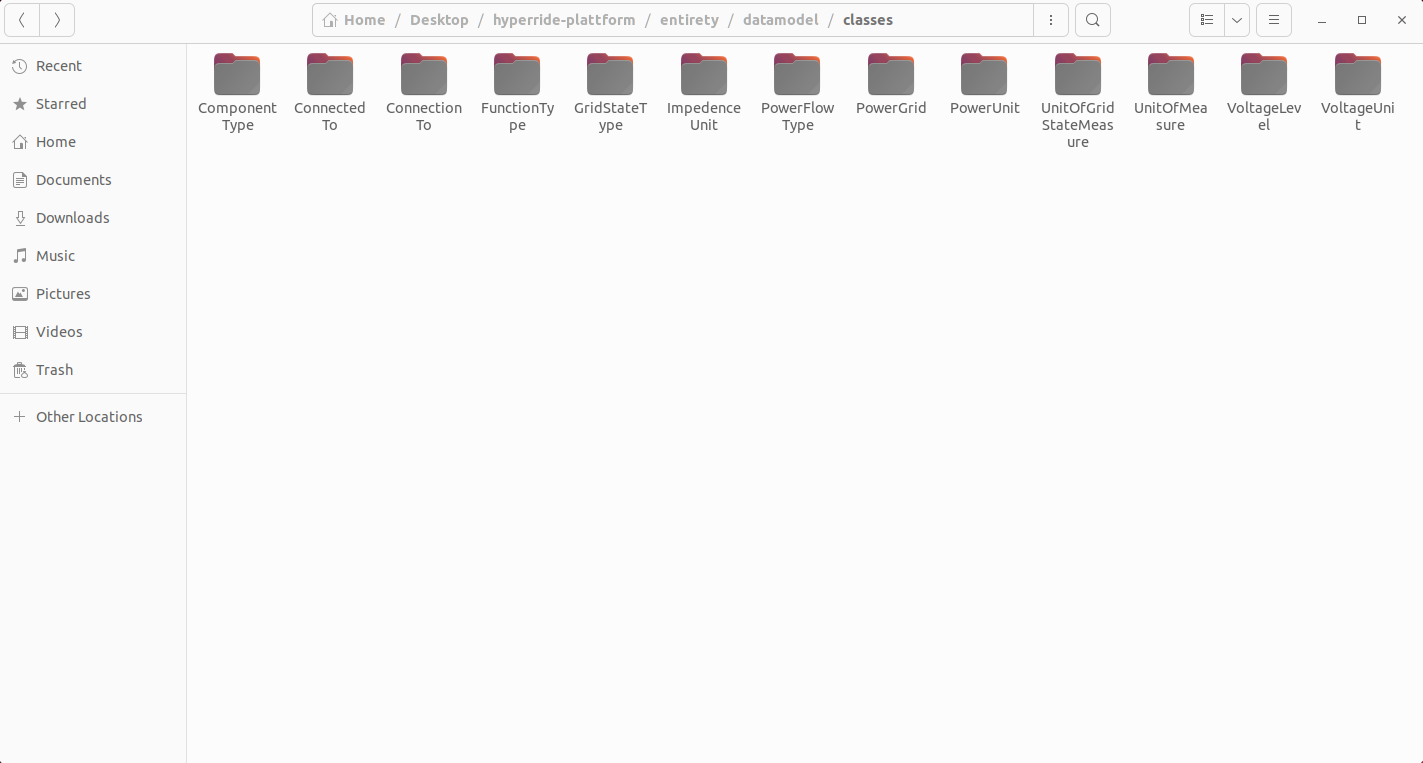
The image shows the component .json files:



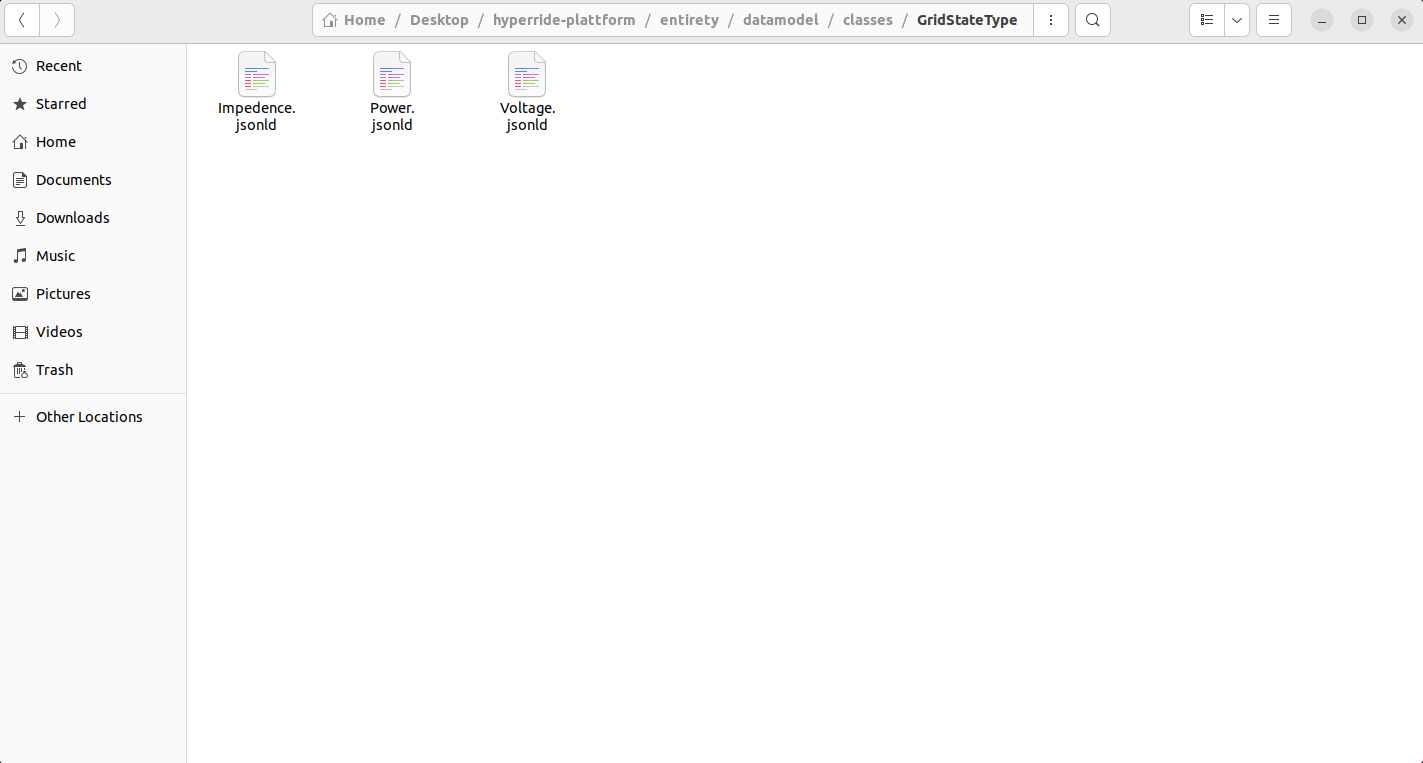
1. **Add Classes:**

The classes were added as folders in the classes folder. Each class folder represents an entity. The directory to the folder is “entirety --> datamodel --> classes”.

The image shows the class folders:

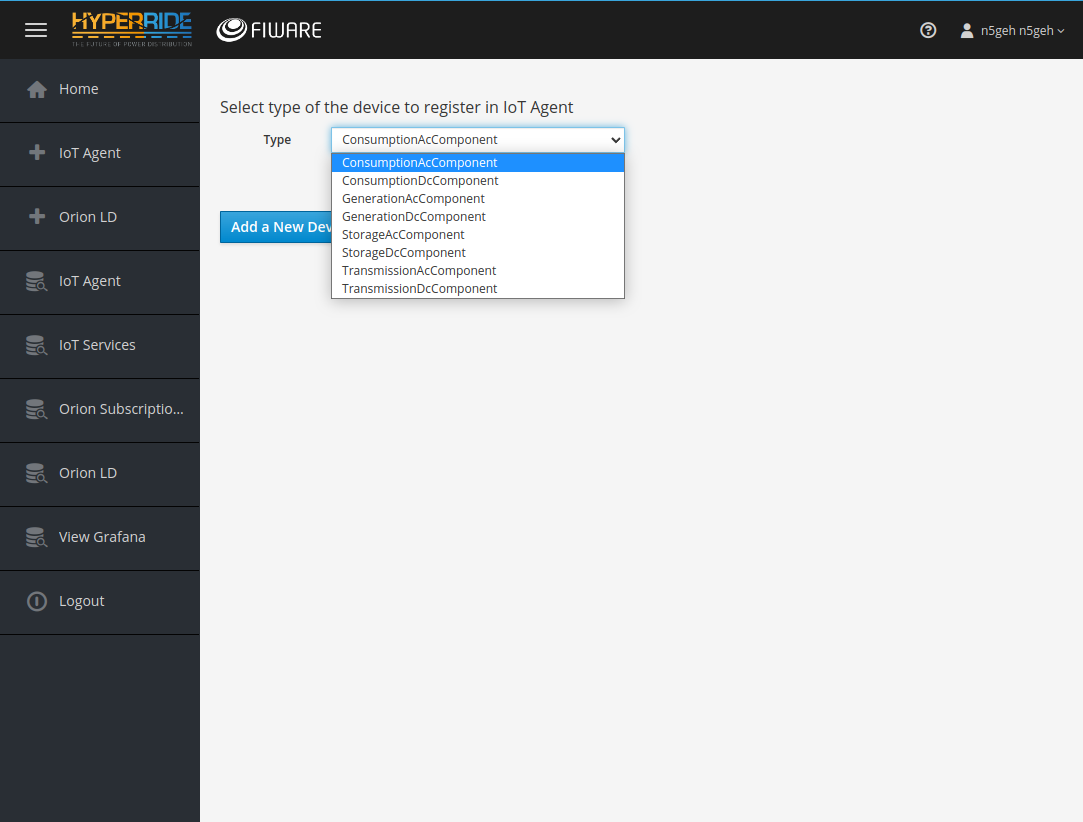


Inside the folders are entity types, if they have any.  
The image shows one example of the entity “GridStateType”:

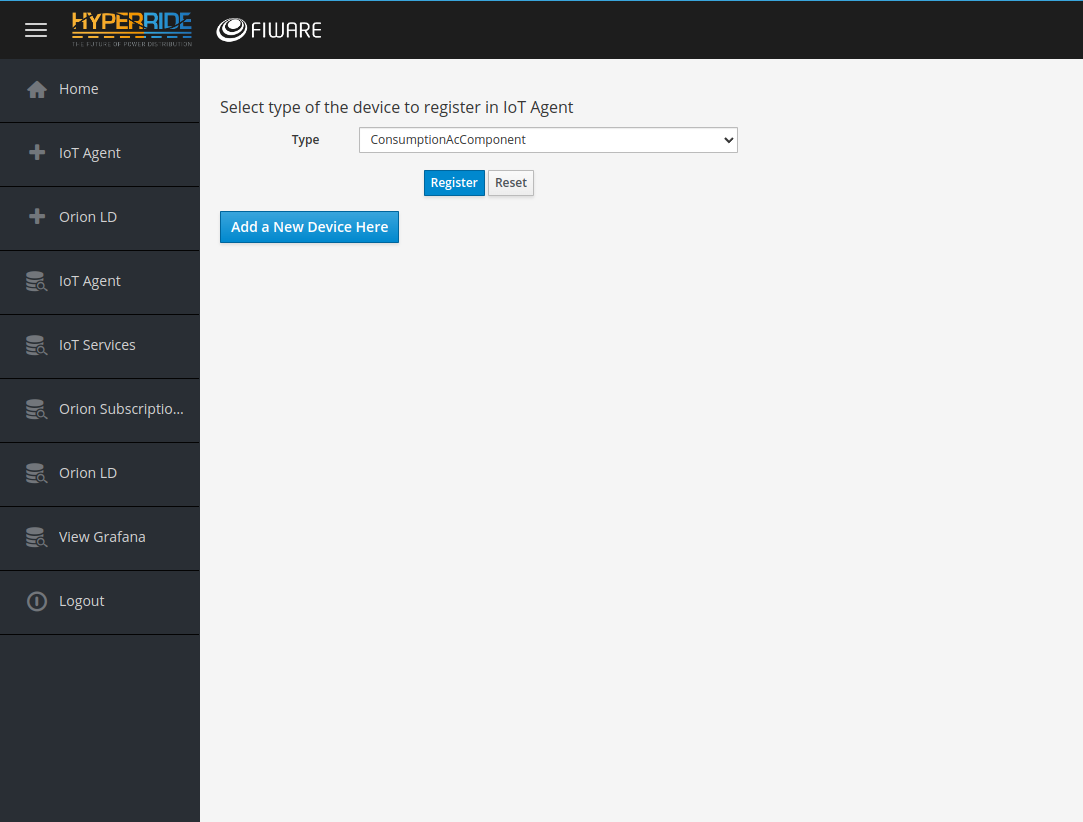


1. **Run command “Sudo ./services secrets\_update” in terminal, from where project is run.**
2. **Add and new component (IoT Agent) and try:**

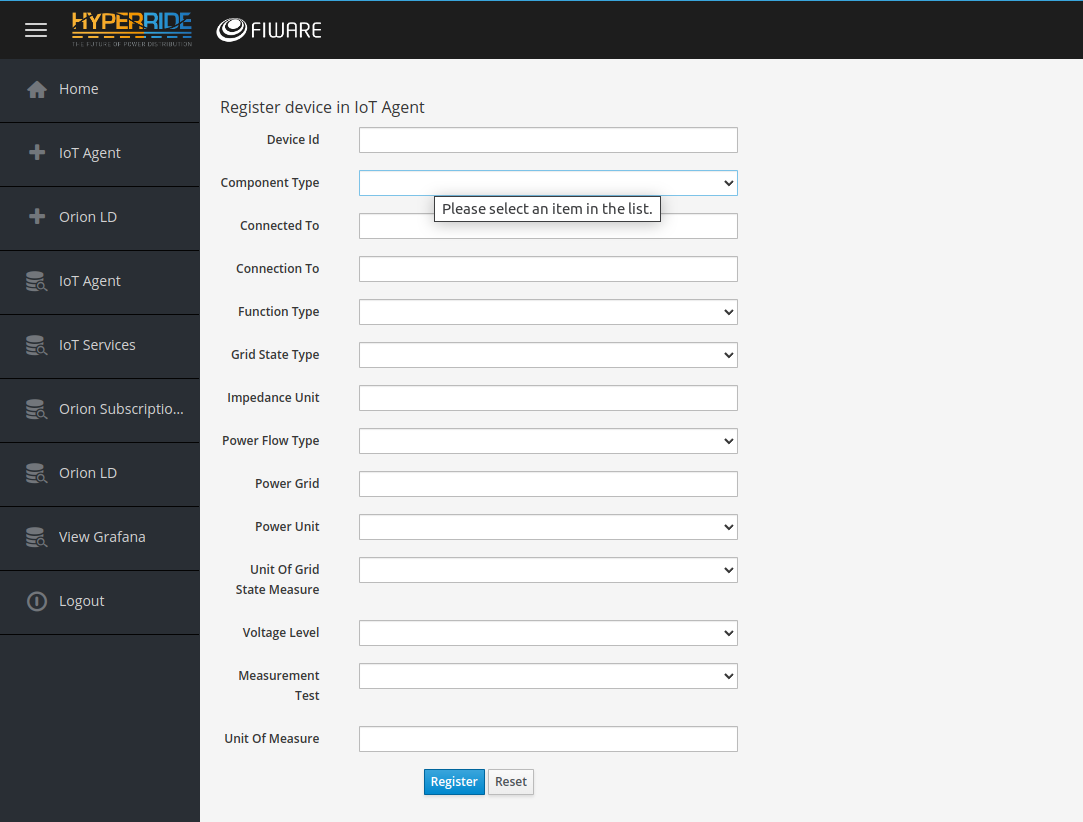
First Go to entirety (Localhost:8090). Select IoT Agent, click dropdown and the new component (IoT Agents) will appear as follows:



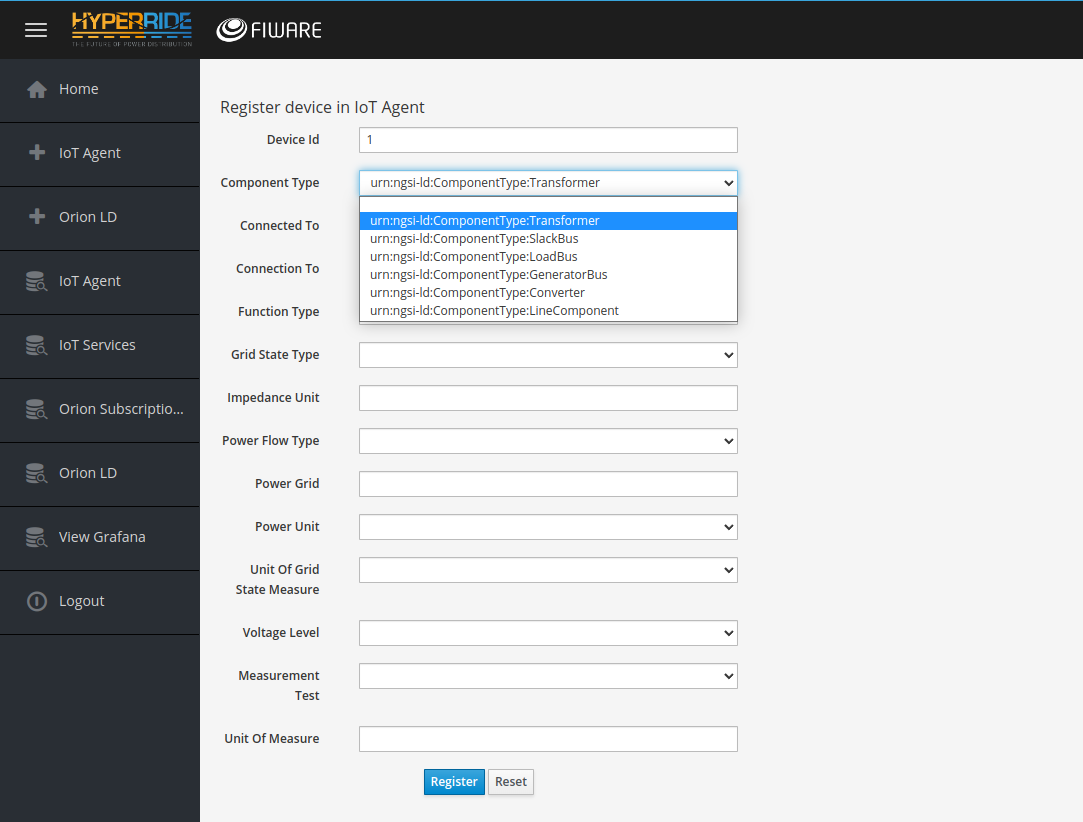
Select a Component. We have taken “ConsumptionAcComponent” as an example for illustration.

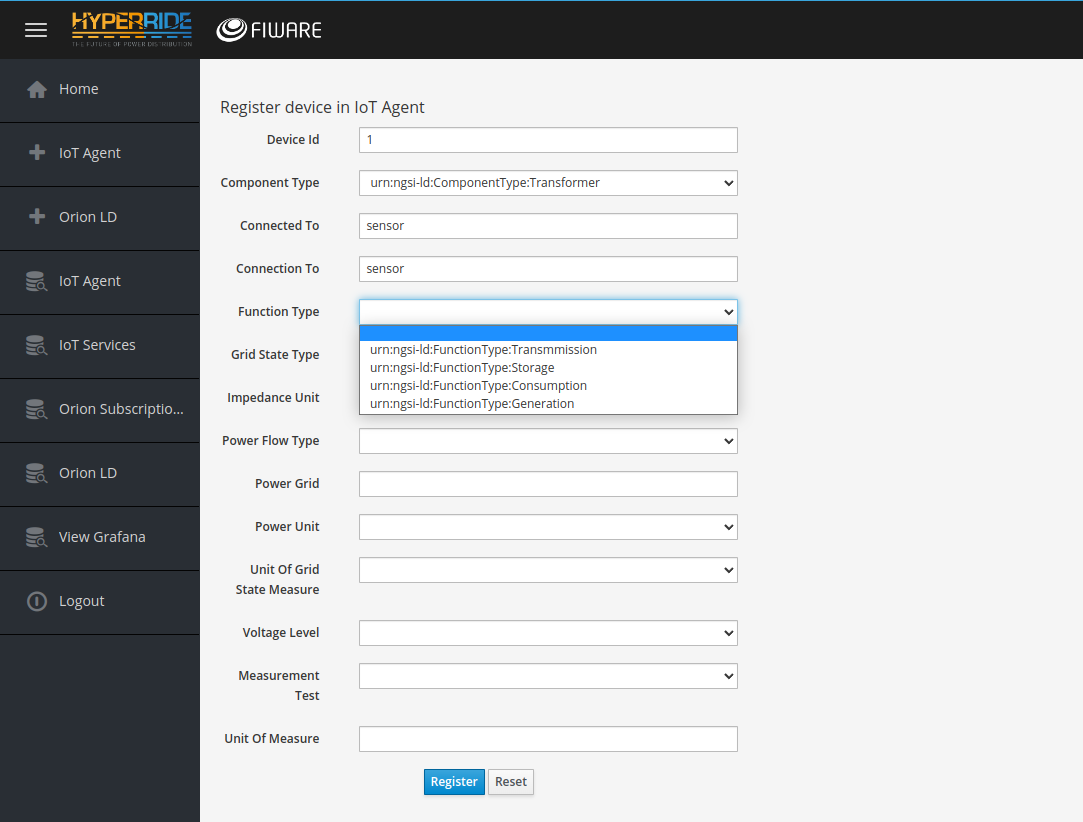


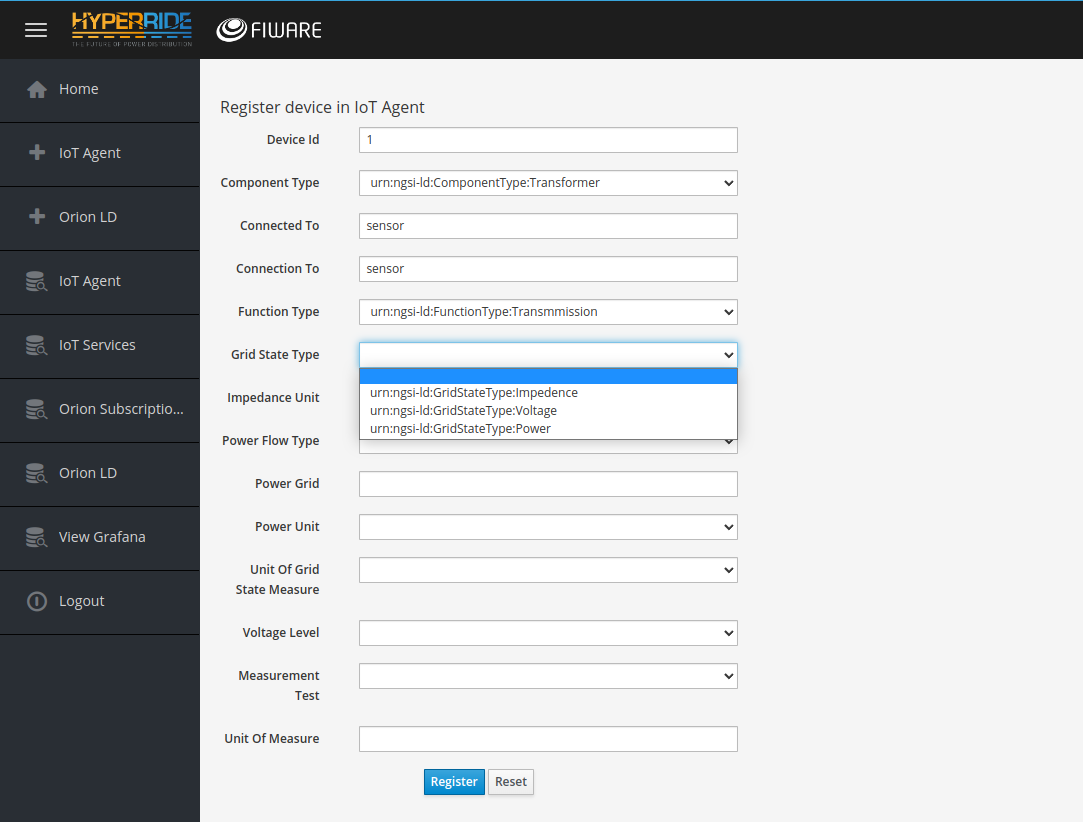
The next images show the form that comes after choosing a component:

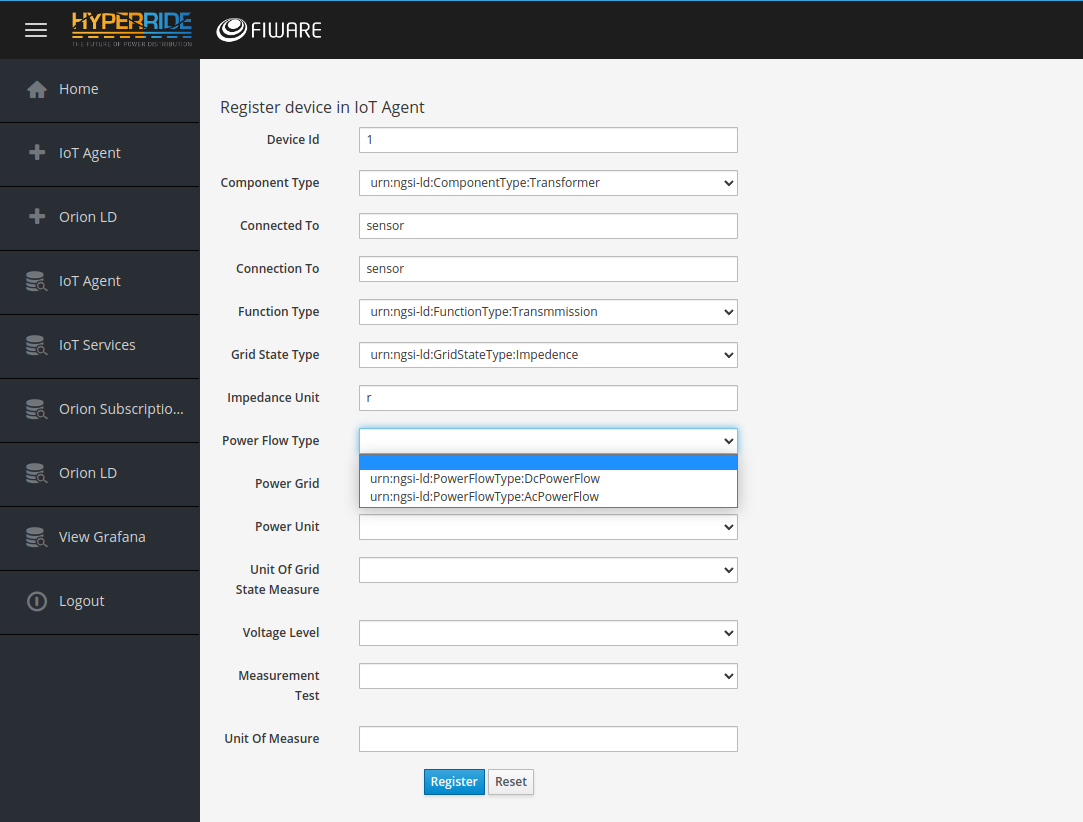


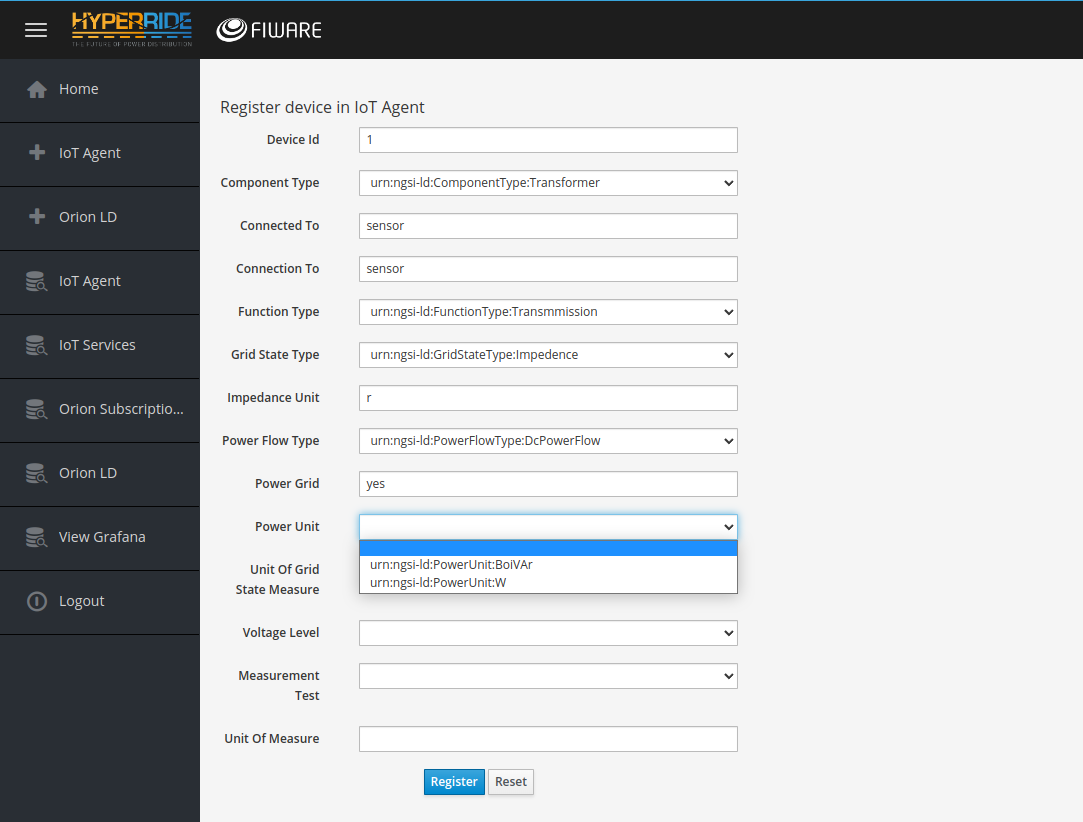
All the upcoming form just gives and example of form filling:

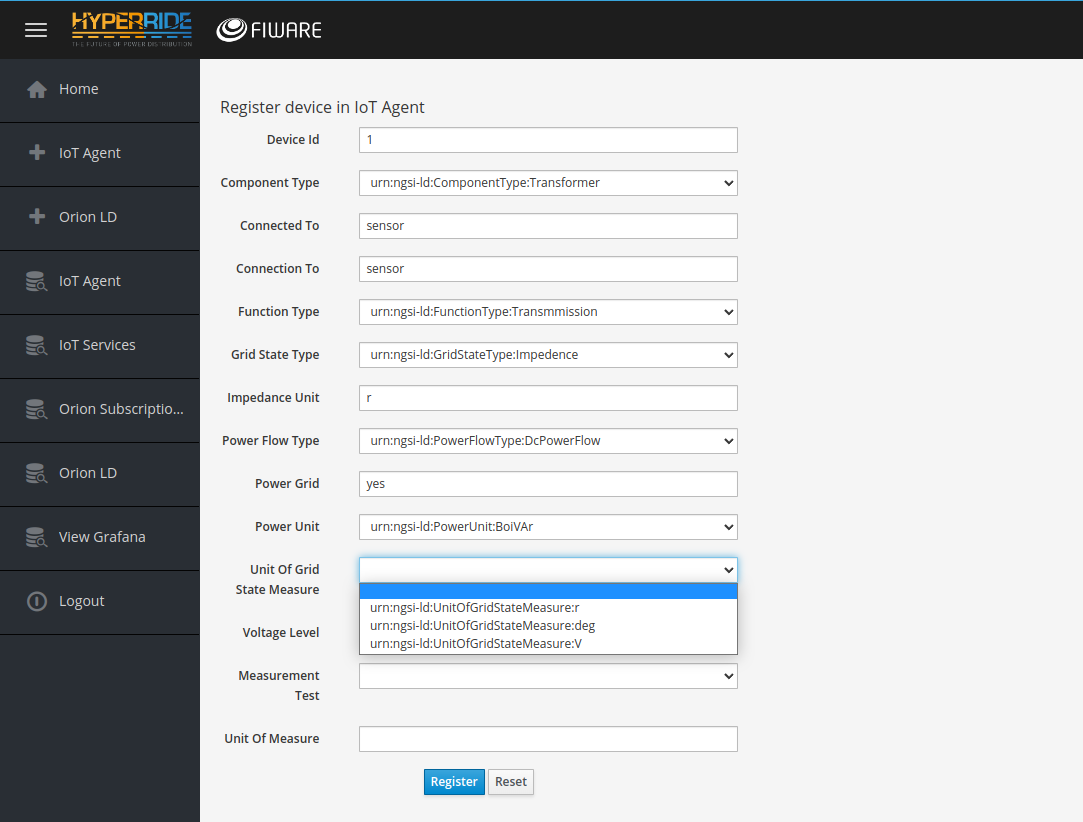


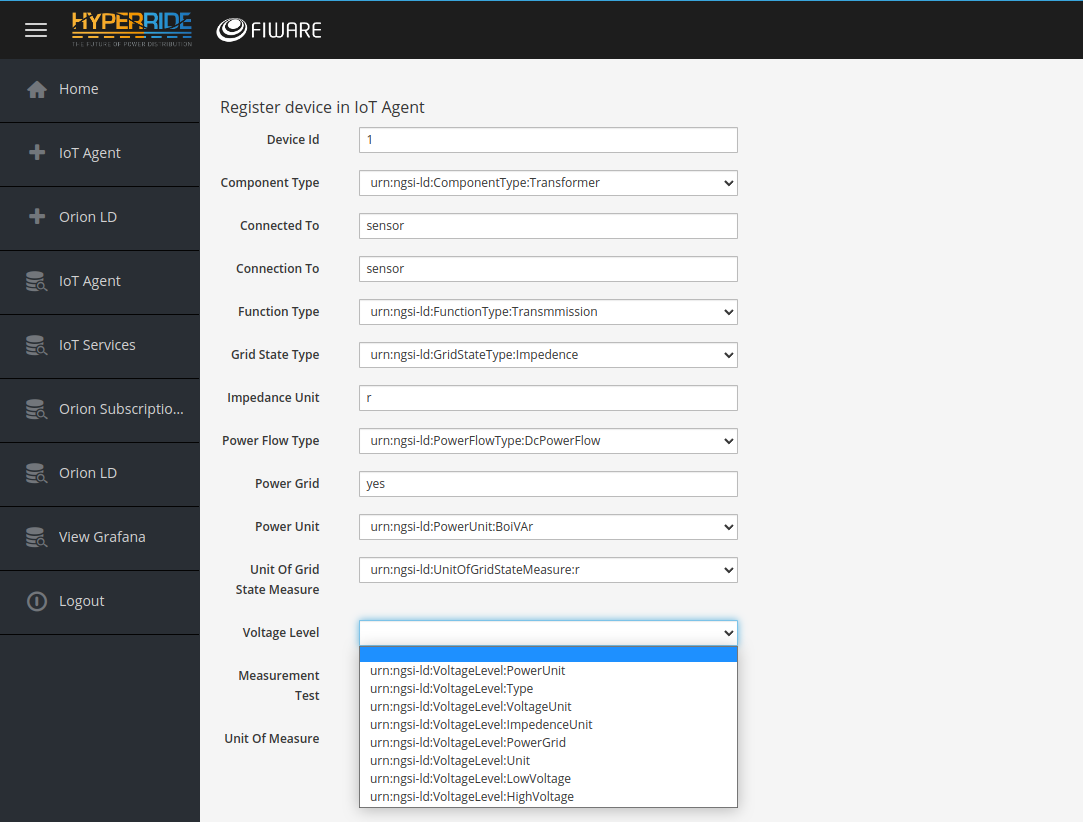


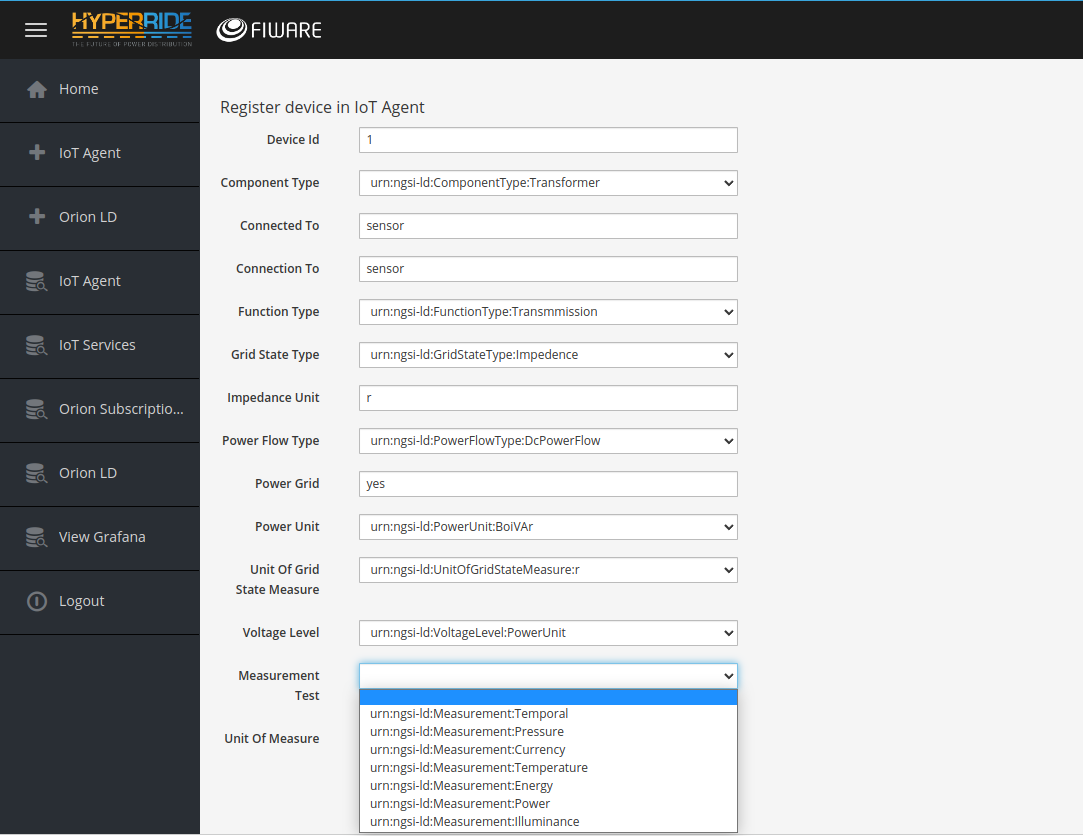


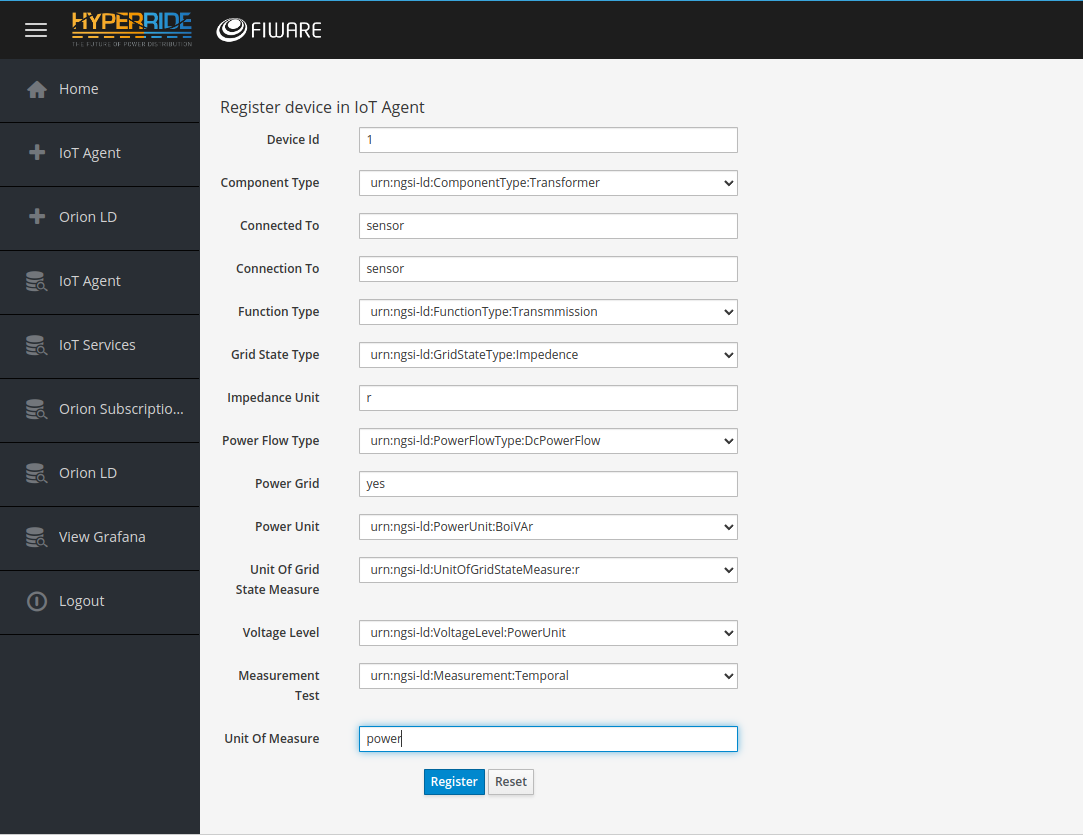




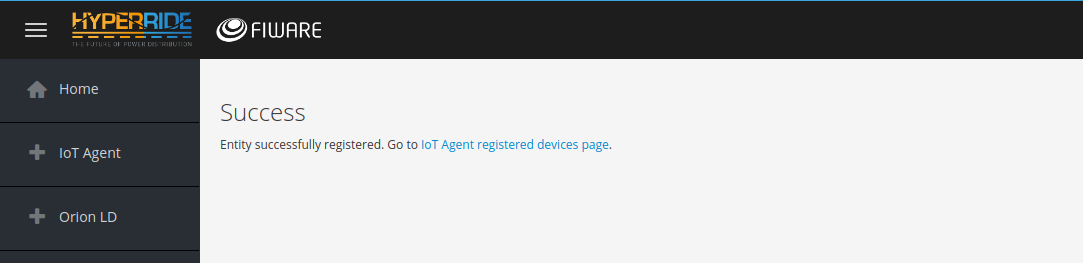




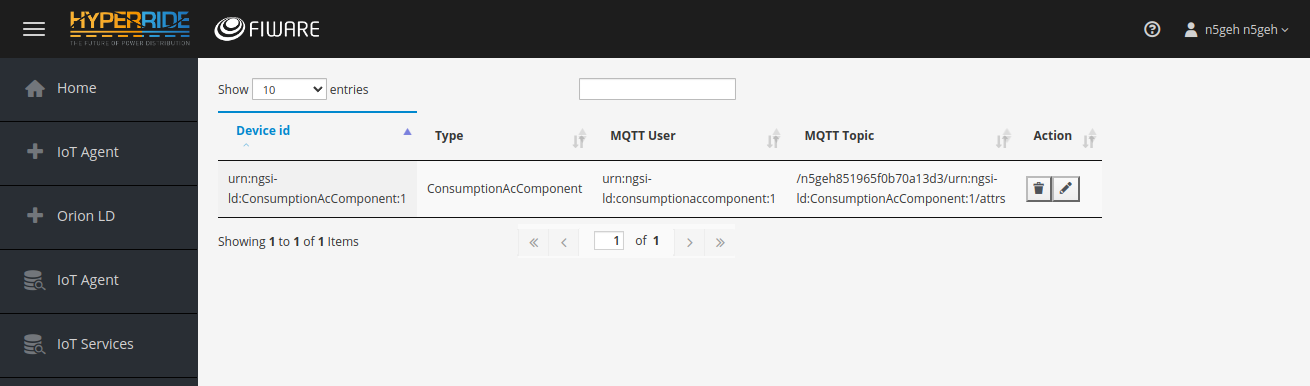




Finally, click on “Register” and this is notification will appear:



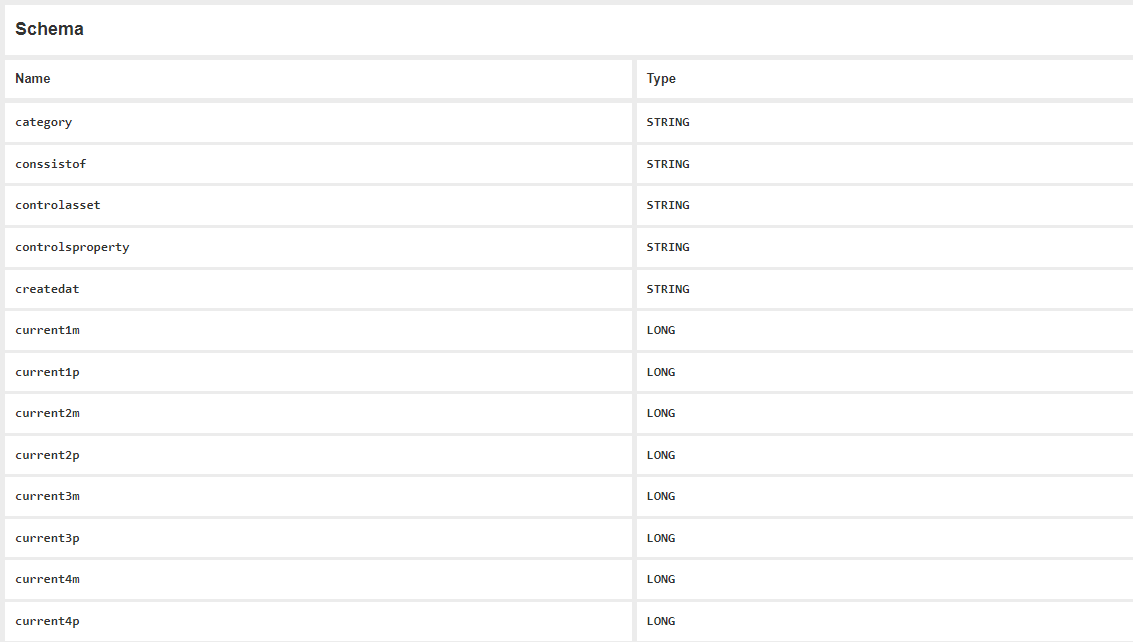
Finally, we got to IoT Agent Data Table and new device should appear as shown:



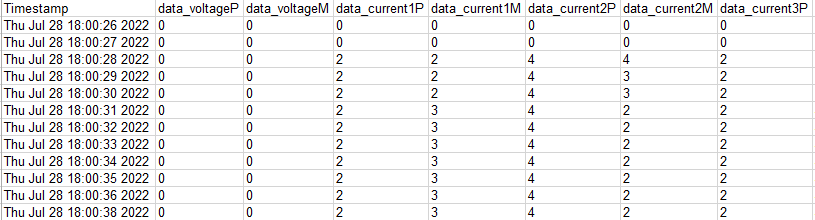
**CrateDB and Graphana Part**

**Using the service:**

We create database table in entirety according to the sensor we will use. This the entirety part. We do this as follows:

The table is created in the CrateDB as follows:

1. The parameters of the table are defined in the data model in the code as NGSIv2, NGSI-LD, JSON formats. This The Quantum Leaps part to read the formats of NGSIv2, NGSI-LD, JSON. Also the Orion Context Broker has role here for the Format checking.

1. we have data from sensor as csv. This is the sensor part and in our case the sensor is switchgear. The csv Files looks as follows:
2. We send the data with the MQTT Broker via the IOT Agent. This is done by python code. The data goes and is stored in the CrateDB.

The code:

import paho.mqtt.client as mqtt

import pandas as pd

import time

df = pd.read\_excel('D:\switch\_gear\Switchgear1.xls')

df["Timestamp"]=pd.to\_datetime(df.Timestamp)

col\_name=[]

for col in df.columns:

    col\_name.append (col)

col\_name = col\_name[:-2]

for index, row in df.iterrows():

    client = mqtt.Client()

    client.username\_pw\_set("urn:ngsi-ld:boiler:boiler001", "secret")

    client.connect("137.226.248.224", 1883, 60)

    client.publish("/n5geh849992df6d1f7922/urn:ngsi-ld:Boiler:boiler001/attrs", "dvP|{}|dvM|{}|dc1P|{}|dc1M|{}|dc2P|{}|dc2M|{}|dc3P|{}|dc3M|{}|dc4P|{}|dc4M|{}".format(

        int(row[col\_name[1]]),

        int(row[col\_name[2]]),

        int(row[col\_name[3]]),

        int(row[col\_name[4]]),

        int(row[col\_name[5]]),

        int(row[col\_name[6]]),

        int(row[col\_name[7]]),

        int(row[col\_name[8]]),

        int(row[col\_name[9]]),

        int(row[col\_name[10]]),

    ));

    client.disconnect();

    count= index+1

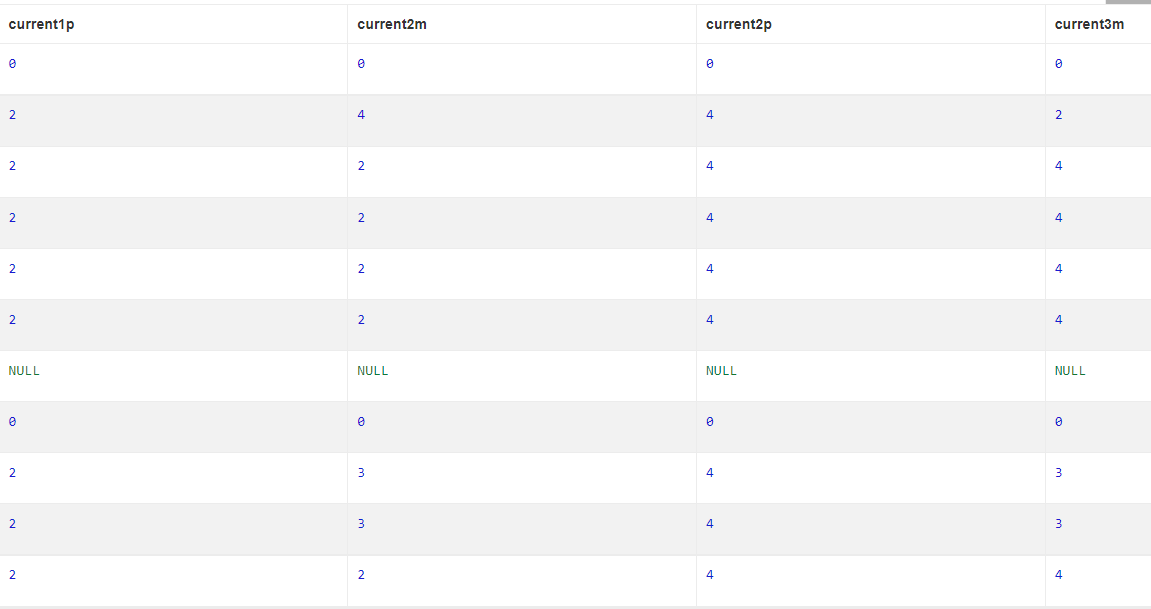
    print (count," row inserted in the table")

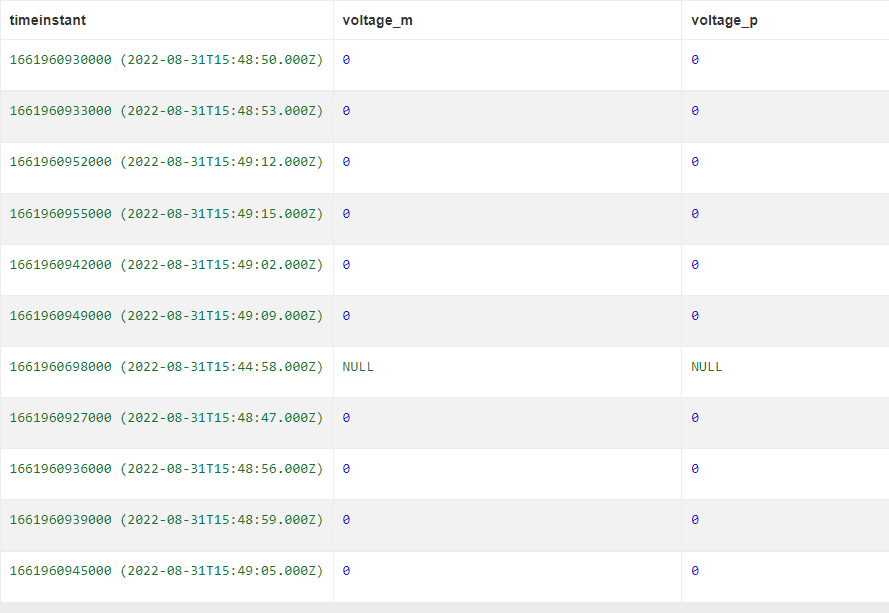
    time.sleep(3)

    if count == 10:

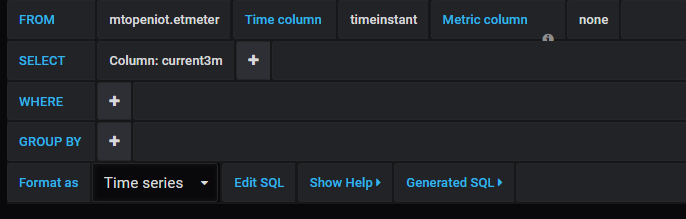
        break

1. The data is uploaded in the crate DB**.**

The records look as follows:



1. We can portray this into grafana like this by the Query example:



Then the data appears as follows:

