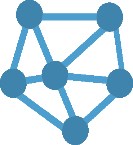
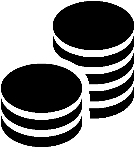
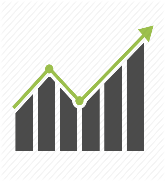
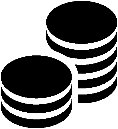
**The N5GEH Entirety**

1. **Goal**

There are varies sources of data from historical facts to human surveys. These data have very less use if it is not process or if it is not analyzed. There could be different ways to analyze the data one of them could be by graphical visualization. Now, from getting the data from the source to the graphical representation may include countless number of steps depending on type of project. Some of the common steps are getting the data, input the data (manually or automatically) processing the data, uploading the data into a data base and finally display the data graphically from accessing it from the data base.   
  
Instead of going through all the process one by one, or having different platforms to do all of these one my one, the goal is to use and modify N5GEH Entirety as one platform to make the process simpler. For our case, from getting various types of data from the sensors to different types of graphical visualization of the data, the goal will be to use N5GEH Entirety to do it in the simpler way. Also keeping in mind that where there is data, there we also need security, this is also something that the N5GEH Entirety providing us.

The model of the N5GEH Entirety is as follows:

**MQTT Broker**



Port: 3000

Port: 8080

Port: 27017

Port: 8668

Port: 4200

Port: 8090

**Mongo DB**

**Grafana**

**Orion Context Broker**

**Crate** **DB**

**Quantum Leap**

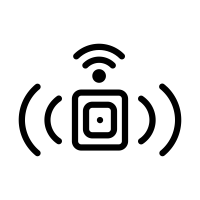


**Key** **Cloak**



Port: 7896/4041

Port: 7896/4041



**Sensor**

**IoT Agent**

**Entirety**

**About the Components:**

There are total of 8 components Some of them are Fiware components and Some of them are open source.

**FIWARE**

FIWARE is an open source platform that helps in the easy build and maintain software applications and services. FIWARE comprises of components that make complex process simple, cost effective and secure, increasing the overall efficiency of the smart solutions. The components include high performance databases, advance level Representational State Transfer (REST) application programming interfaces (API). The platform components are integrated by standardized Next Generation Service Interface (FIWARE NGSI) which is the API exported by a FIWARE context Broker.

For the N5GEH project, some of the main FIWARE platform are used that provide optimum features in the field of majority of IoT-Applications. These include the Orion Context Broker, Quantum Leap and IoT agent show in Blue in Figure 2 below. They are so-called Generic Enablers (GE).

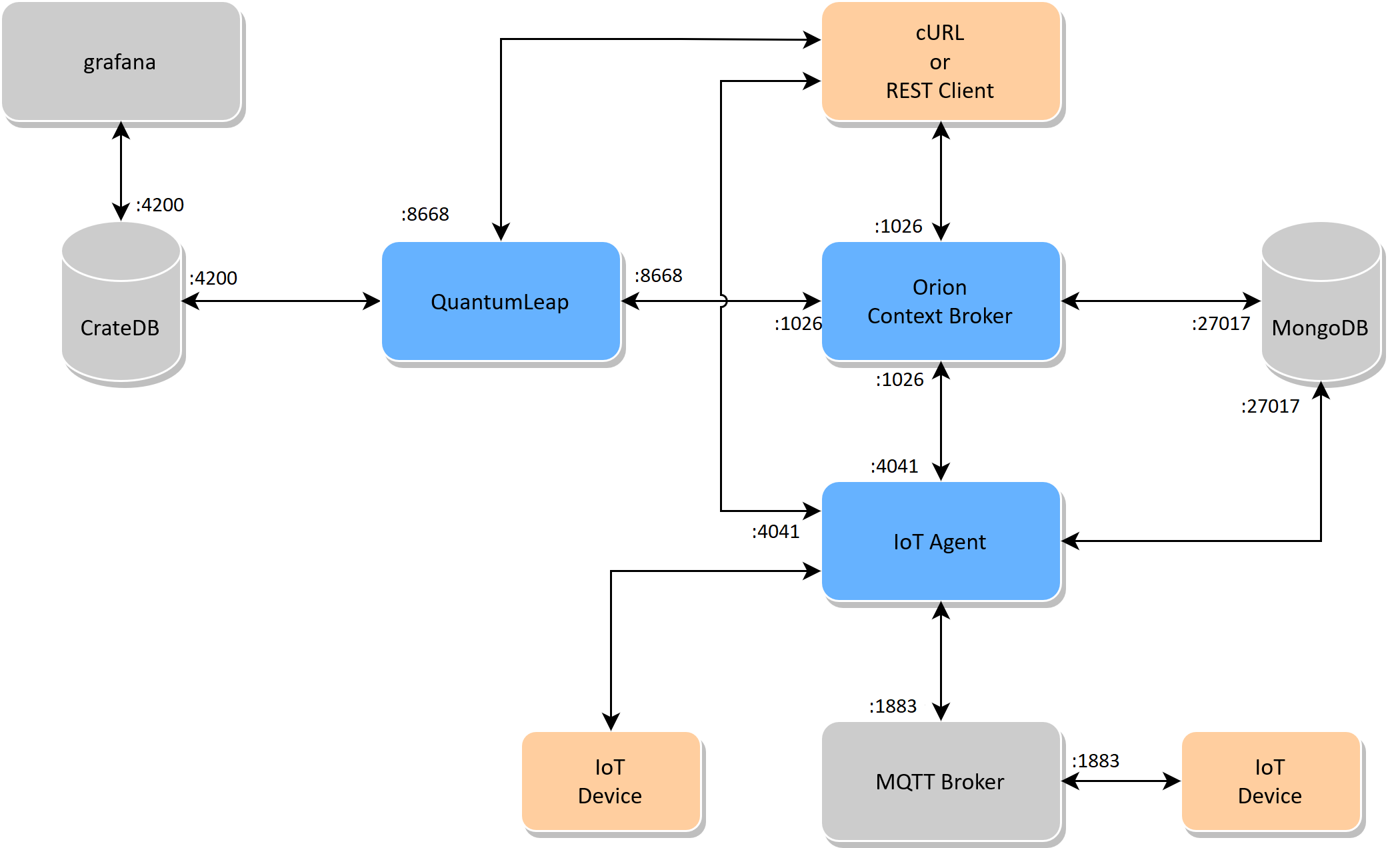


Figure 2 [12]

**2. Components and Description:**

**Components:**

**Grafana (https://grafana.com/):**

Grafana is a fully managed cloud based open source observability platform which allows the user to implement their applications and infrastructure. The property that makes Grafana a very good choice is that it consists of very promising open source observability software such as Prometheus, Loki and Tempo. Not only the features of the mentioned software are made into optimal usage but also it is free of any kind of expenses related to installation, maintenance and scaling. Grafana cloud specializes in the visualizing high level graph and insightful logs in a very short period of time in just few simple steps. Generally doing so from scratch, that is creating an integrated observability stack from any other open source takes a lot of time that a user might not be able to afford.

There are quite some features that Grafana Cloud provides. To begin with, Grafana provides full stack metrics and logging facility in a very short period of time. Furthermore, Grafana allows the user can bring their own data, this data can be processed by queries and later dashboards can be created with it. Most importantly, Grafana is fully managed, although the visualization, dashboards, data sources and etc would be running from the user’s environment but it will be managed in Grafana Cloud.

**CrateDB (https://crate.io/):**

CrateDB is an open source SQL database. It is combination of SQL concepts and features of NoSQL which includes scalability and flexibility. CrateDB gains popularity because of the simplicity and affordability when it comes to manage the size, speed and types of machine and log data. Therefore, leading customers to have very positive feedbacks.

Because CrateBD is based on the concepts or syntax of SQL, user can get compatible with the system very easily. Users can process structured and unstructured data using SQL queries. Also, Different types of JOINS and aggregates can be used similar to SQL and the performance in the speed remains the same. Therefore, all together scaling becomes very simple.

**Quantum Leap:**

QuantumLeap is a REST service for storing, querying and retrieving NGSI v2 and NGSI-LD (experimental support) spatial-temporal data [18]. QuantumLeap converts NGSI semi-structured data into tabular format and stores it in a time-series database, associating each database record with a time index and, if present in the NGSI data, a location on Earth. REST clients can then retrieve NGSI entities by filtering entity sets through time ranges and spatial operators. Note that, from the client's stand point, these queries are defined on NGSI entities as opposed to database tables. However, the query functionality available through the REST interface is quite basic and most complex queries typically require clients to use the database directly.

**Entirety:**

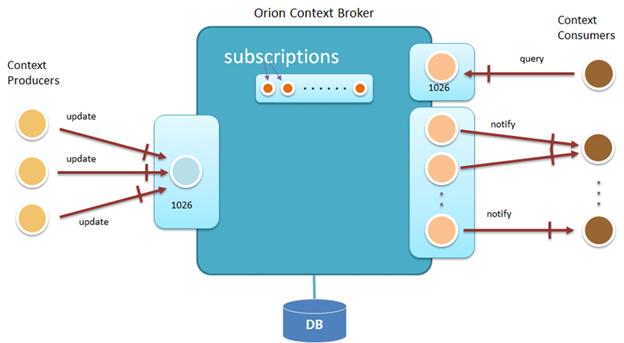
ENTIRETY is an open-source web-based application that gives consistency to the devices in IoT platforms. This is done by governing and provisioning in the simple and unified way. Currently, the main focus of ENTIRETY is the large scale smart energy domain applications. However, other domain-specific devices can also be an extension of ENTIRETY.In order to overcome the IoT diversity challenges, the core of the ENTIRETY is a SARGON ontology which is the semantic data model designed for cross-cutting the domain-specific information in the smart energy system which is modular, extendable and reusable. Moreover, the approach of entirety is template based making it easily be runnable as a Docker image. [19]

**Orion Context Broker:**

The Orion Context Broker is an implementation of the Publish/Subscribe Context Broker GE, providing an NGSI interface. This interface can be used by the clients to do numerous functionalities.

Key functionalities:

* Context information updating (e.g. update humidity)
* Notification on context information change/updates (e.g. change in humidity)
* Context provider applications registration (e.g. provider for humidity sensor within a room)

  
 Figure 3 [3]

**MongoDB (https://www.mongodb.com/):**

MongoDB is an open-source, cross-platform, and distributed document-based database designed for ease of application development and scaling. It is a NoSQL database developed by MongoDB Inc. MongoDB database is built to store a huge amount of data and also perform fast.

MongoDB works on the concept of “NoSQL” database, which is completely opposite to a SQL. Hence it is not a Relational Database Management System. Data in MongoDB is not stored or categorized in tables where as it is stored as a collection JSON based document. So, no schemas, tables, rows or columns. But, similarly in MongoDB, a collection can have multiple documents which are basically rows in this case, each document has multiple "fields" which are basically columns in this case and documents in a single collection can be defined as having different fields in this case.

**FIWARE IoT Agent (https://www.fiware.org/):**

IoT Agent is one of the component of the FIWARE. The purpose of IoT agent is to allow devices to send their data to and be managed from context Broker via their own protocols. It also essential for the IoT agents to be compatible with the security features of the FIWARE platform and then the device programmer should receive the other common services.

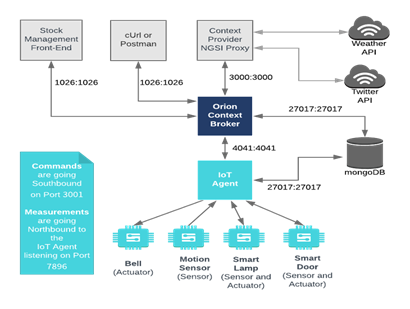
In effect, this brings a standard interface to all IoT interactions at the context information management level. Each group of IoT devices are able to use their own proprietary protocols and disparate transport mechanisms under the hood whilst the associated IoT Agent offers a facade pattern to handle this complexity. [20]

Figure 4 [20]

**Keycloak(https://www.keycloak.org/):**

Keycloak is basically a web base GUI platform for managing the accessibility of a user by maintaining secured authorization of the user. Also, Keycloak is an open-source platform currently licensed with Apache License 2.0.

**Keycloak Key Features:**

## **Single-Sign On**

The authentication of a specific application can be done via Keycloak and this needs to be done only once. That means Keycloak takes care of the login forms, authentications and etc for the user’s specific application and the user does not have to worry about it.

**Identity Brokering Social Login**

Via the admin console, social networks login can be activated by just selecting the social network that the user wants to activate. Same goes for Identity provider.

## **User Federation**

Built-in support to connect to existing LDAP or Active Directory servers are available. Own provider can be implemented as well, given that Keycloak user have their application users in other stores.

## **Admin Console**

Admin console allows administrators to centrally manage all aspects of the Keycloak server like Different kinds of access and permissions.

## **Account Management Console**

Just like normal account management, users can update profile info, change passwords and link profiles from additional providers if they have enabled social login or identity brokering.

## **Authorization Services**

Keycloak provides fine-grained authorization. This lets the user to manage permissions for all services from the Keycloak admin console and allows the user to be more specific to the policies.

**MQTT Broker (https://mqtt.org/):**

**About MQTT:**

MQTT (Message Queuing Telemetry Transport) is an open source communication protocol. Generally, it runs over the TCP/IP stack. When it comes to bandwidth and system resource, MQTT is very much efficient. For this reason, MQTT is highly cost efficient and is very much popular in in the field of IoT (Internet of things) and also IIoT (Industrial Internet of Things).

Main uses of MQTT include data collection and transmission to server. Moreover, Publishing data from sensor node to user device directly. Also, Configuration of IoT and IIoT devices remotely.

**MQTT Broker**

In the MQTT architecture, an MQTT broker is a central software entity that acts just like a real estate broker which first does background checks on the parties involved and then after making sure that the relevant rules are enforced, the broker initiates a transaction [10]. Generally, there are two types of brokers which include the Managed Brokers and Self-Hosted Brokers

**Sensors:**  
  
Sensors are used in the architecture of IOT devices [11]. Application of the **Sensors** are to sense and take input of different physical parameters such as heat, light, sound, speed, sound, vibrations and etc. The physical parameters obtained are converted into a signal that is suitable for processing the properties of any device or material to detect the presence of a particular physical quantity [11]. The processed data is used to produce a response according to the requirement with use of output devices such as actuators and displays.

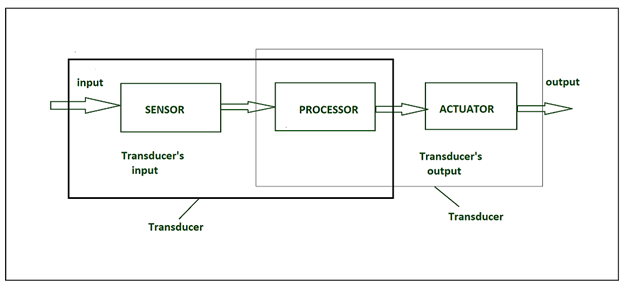


Fig 7 [11]

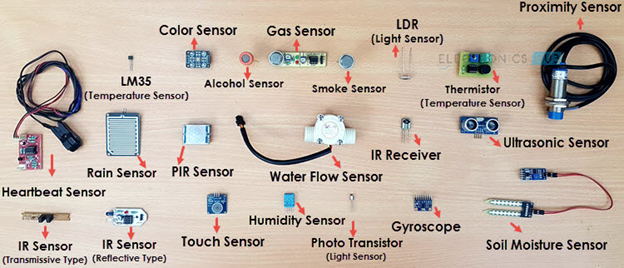


Figure 8 [13]

**Docker (https://www.docker.com/):**

The whole project is compiled in Docker and the project is deployed from it.  
  
**Description:**

Docker is a software platform for building, testing and deploying applications. Docker isolates applications in the form of containers. The container contains all the necessary dependencies (such as libraries, system tools, codes) that the application needs in order to run properly. Therefore, by using Docker, the user can instantly deploy and scale their application into any environment being guaranteed that their application code will run in the same way as tested.

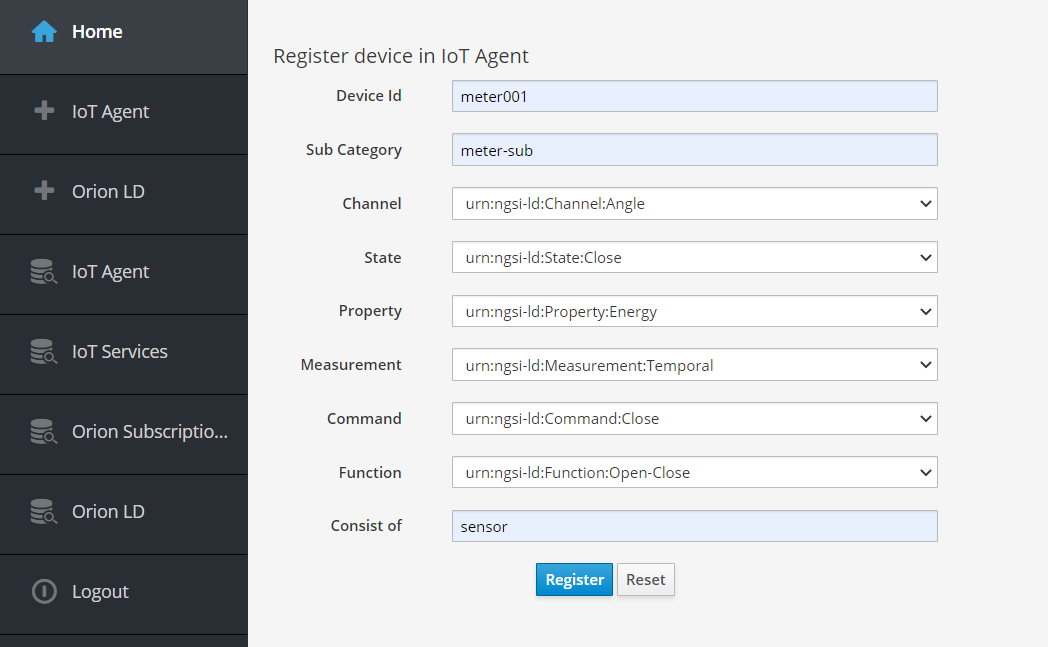
Docket consist of several components that include the Docker file, the Docker image, the Docker run utility, the Docker Hub, the Docker Engine, the Docker Compose and the Docker Desktop.

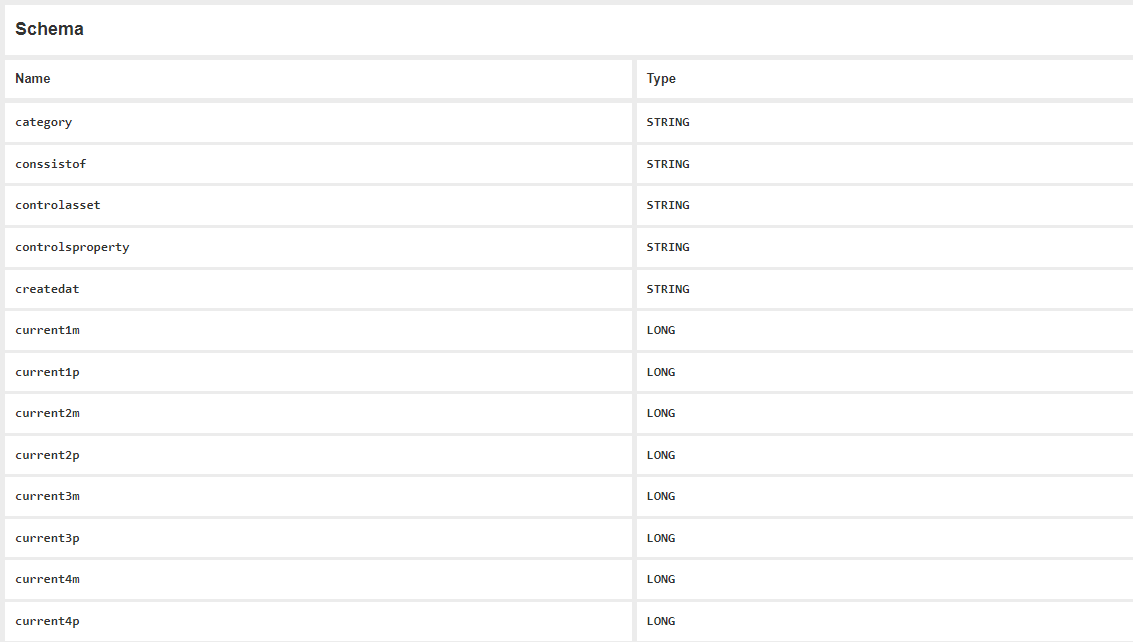
**3. Our Experiment with the project:**

**Project Deployment:**

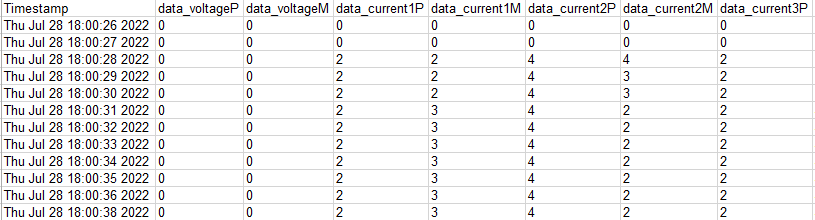
1. We need docker engine and docker compose as precuisite.
2. We clone with the git and run the commands and the project is deployed
3. we change secret keys for security measure login. This is the part where the Keycloak comes in.
4. Now we update restart and the platform is ready.

**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

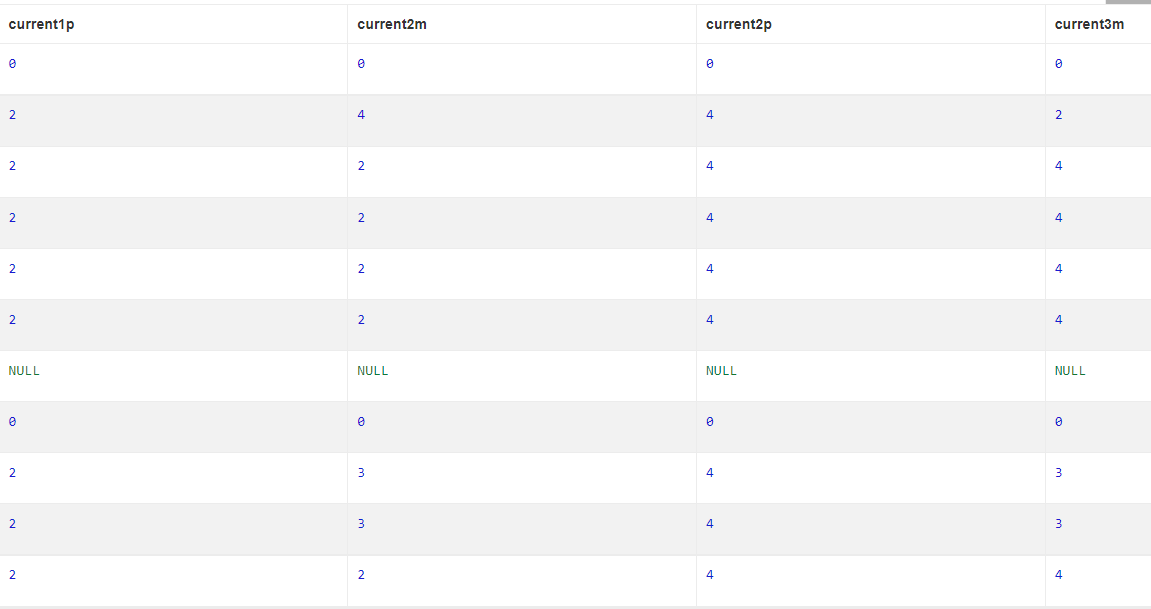
    print (count," row inserted in the table")

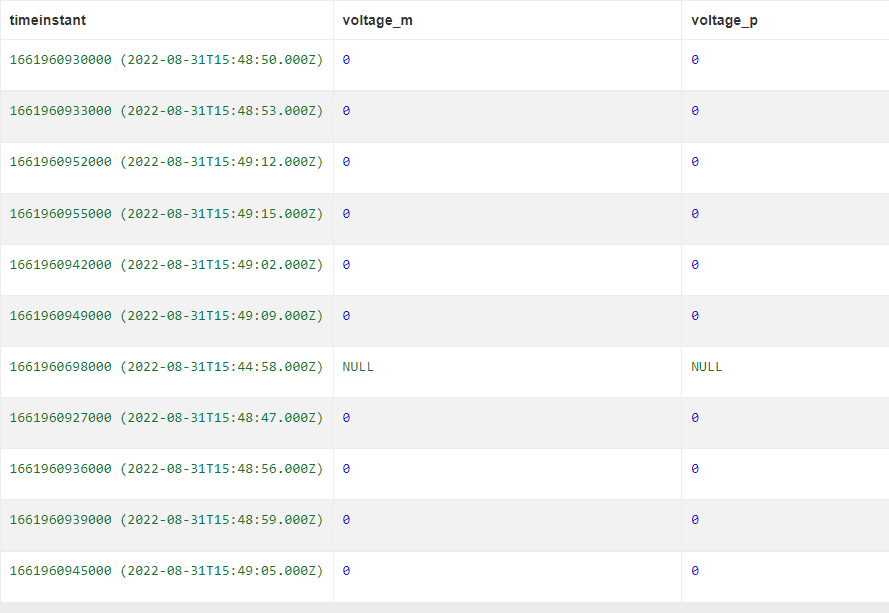
    time.sleep(3)

    if count == 10:

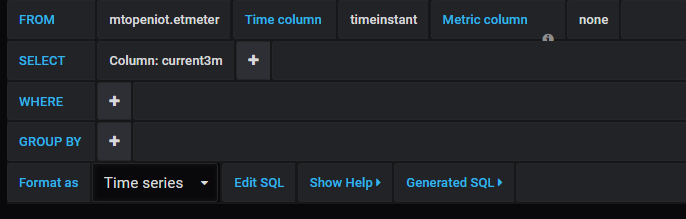
        break

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

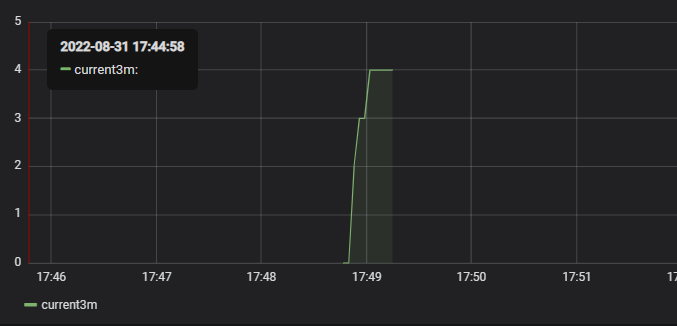
The records look as follows:



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



Then the data appears as follows:



**Referances:**

1. <https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html>
2. <https://github.com/telefonicaid/fiware-orion>
3. <https://www.researchgate.net/figure/Orion-Context-Broker-in-a-nutshell_fig2_330318399>
4. <https://www.tutorialsteacher.com/mongodb/what-is-mongodb>
5. <https://www.guru99.com/what-is-mongodb.html>
6. <https://docs.datadoghq.com/agent/iot/?tab=deb>
7. <https://dzone.com/articles/what-is-keycloak-and-when-it-may-help-you>
8. https://www.keycloak.org/
9. <https://www.techtarget.com/iotagenda/definition/MQTT-MQ-Telemetry-Transport>
10. <https://www.catchpoint.com/network-admin-guide/mqtt-broker>
11. <https://www.geeksforgeeks.org/sensors-in-internet-of-thingsiot/>
12. <https://github.com/N5GEH/n5geh.platform>
13. [What is a Sensor? Different Types of Sensors, Applications (electronicshub.org)](https://www.electronicshub.org/different-types-sensors/)
14. <https://www.fiware.org/catalogue/>
15. [Grafana Cloud | Grafana Labs](https://go2.grafana.com/grafana-cloud.html?src=ggl-s&mdm=cpc&camp=b-grafana-exac-emea&cnt=118483912276&trm=grafana&device=c&gclid=CjwKCAjw4c-ZBhAEEiwAZ105RfFYzpld4964JGRwzenrBo2GoZ5hrdUO4B5zNkRTPx7qZYJpBfpayRoCXasQAvD_BwE)
16. <https://crate.io/products/cratedb>
17. <https://github.com/orchestracities/ngsi-timeseries-api>
18. <https://quantumleap.readthedocs.io/en/latest/>
19. 2022 The Author(s). Published by Elsevier B.V. This is an open access article under the CC BY license
20. https://fiware-tutorials.readthedocs.io/en/stable/iot-agent/