FIWARE
Global
Summit

From Data
to Value

OPEN SOURCE
OPEN STANDARDS
OPEN COMMUNITY

Options for visualization of IoT Data

Jason Fox, Technical Evangelist, FIWARE Foundation

Vienna, Austria
12-13 June, 2023
#FIWARESummit

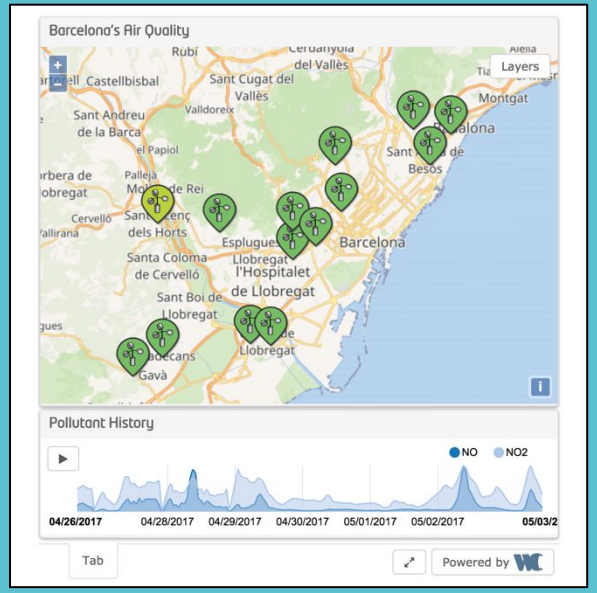
FIWARE
Open APIs for Open Minds

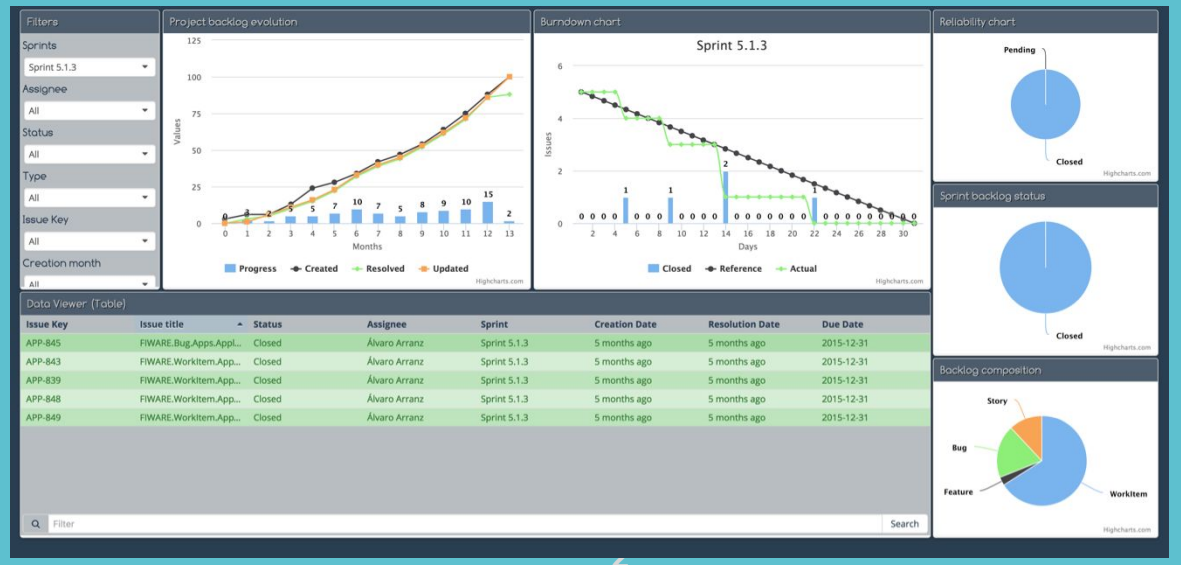
Application Mashup with Wirecloud and NGSI-JS

FIWARE

WireCloud - Dashboards

- Fully customizable and extendable by the development of new widgets and operators
- Ready-to-use support for the FIWARE data models
- Share your dashboards with specific users or make them public. Moreover, your can embed your dashboards on third-party web pages





What is Grafana ?

- Typically used to create complex operations monitoring dashboards
  - GitHub: https://github.com/grafana/grafana
- **Open Source** 

- Allows for ad-hoc **queries**
- Not fixed to a single **data source**
- Able to explore **logs** and **raise** alerts
- Flexible and extensible **visualisations**:
  - Choose what metrics to display
  - Choose how to display them
- On screen visualisations driven by **plugins**.
  - https://grafana.com/grafana/plugins?type=panel
  - https://grafana.com/docs/grafana/latest/installation/docker/#installing-plugins-from-other-sources

Grafana Dashboard with QuantumLeap

FIWARE

What is Grafana ?

- Typically used to create complex operations monitoring dashboards
  - GitHub: https://github.com/grafana/grafana
- **Open Source** 

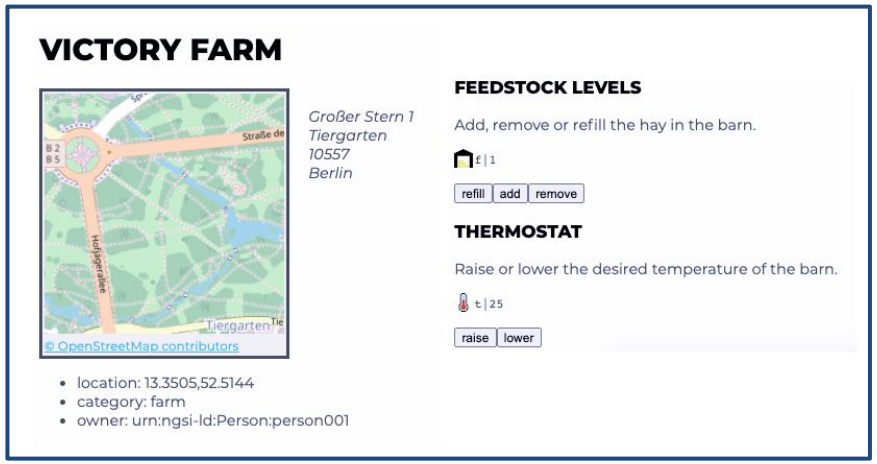
- Allows for ad-hoc **queries**
- Not fixed to a single **data source**
- Able to explore **logs** and **raise** alerts
- Flexible and extensible **visualisations**:
  - Choose what metrics to display
  - Choose how to display them
- On screen visualisations driven by **plugins**.
  - https://grafana.com/grafana/plugins?type=panel
  - https://grafana.com/docs/grafana/latest/installation/docker/#installing-plugins-from-other-sources

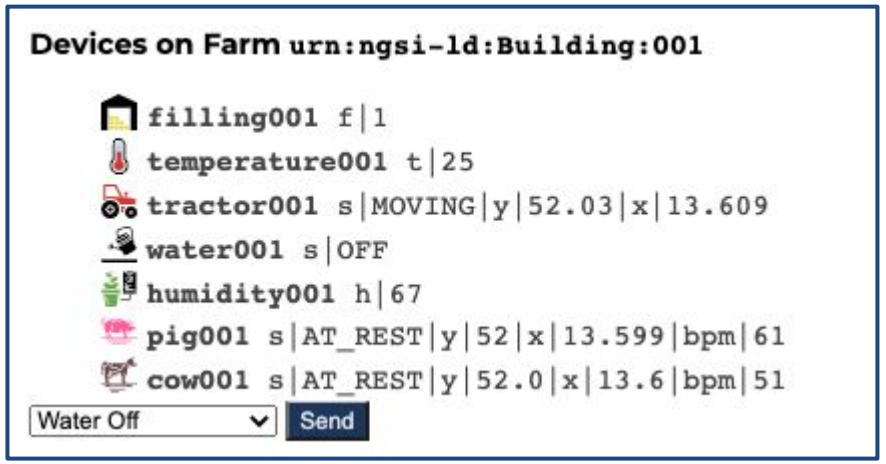
Sources Sinks and Transformations

- Grafana reads in Time-series and Non Time-series data from a Data Source
- A Dozen Common Data Sources available on Startup (e.g. Postgres, MySQL)
- Additional Sources can be added using the plugin mechanism
  - https://grafana.com/grafana/plugins?type=datasource

FIWARE

Monitoring a Filling Station (Smart Farm Simulation)



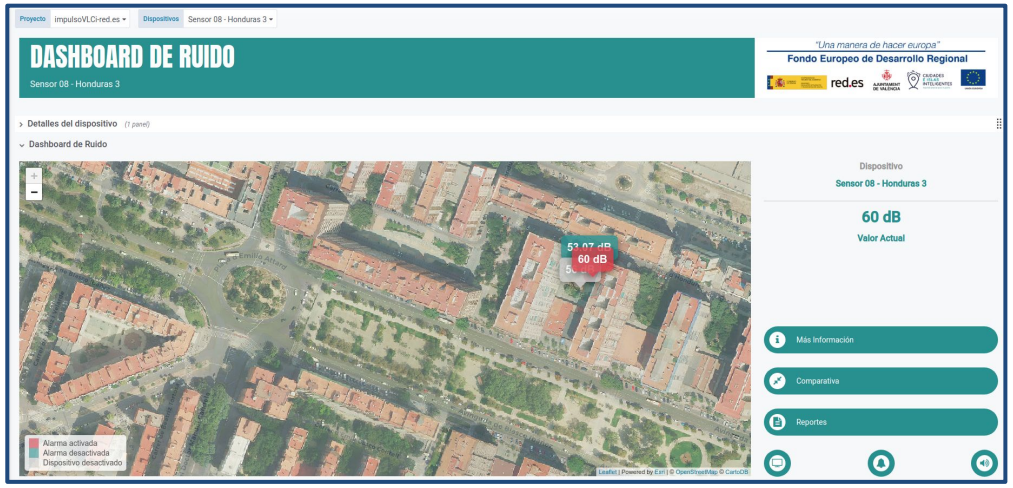


- NGSI-LD Context Data is held in the **Context Broker**
  - IoT Devices → IoT Agent → NGSI-LD Context Broker

- Raise an NGSI-LD Subscription to **QuantumLeap**
- **QuantumLeap** can persist to a Time-series Database (Crate-DB)
- Crate-DB can be used as a Data Source for **Grafana**
- Use **Grafana** panel plugins to display line graph or map or whatever.

FIWARE

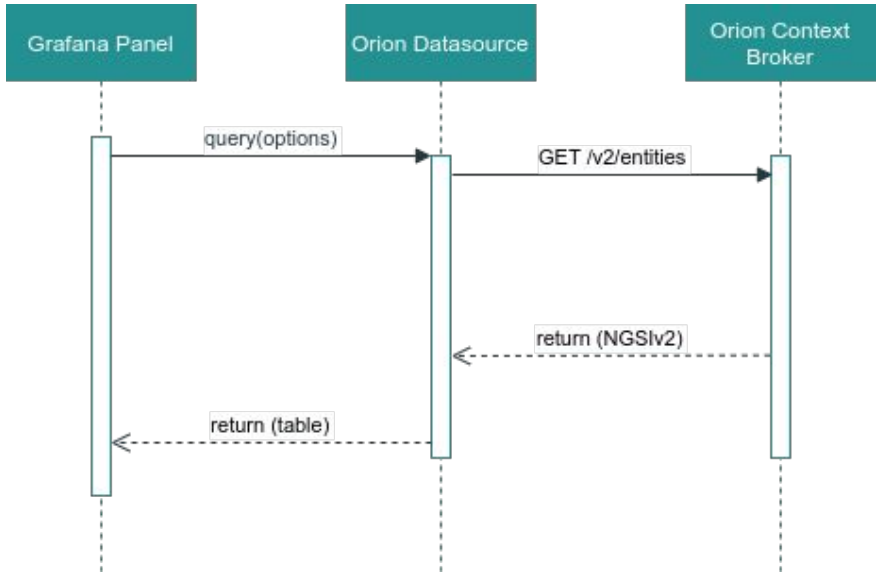
Real Life Use Cases and Customizations (Noise Pollution)



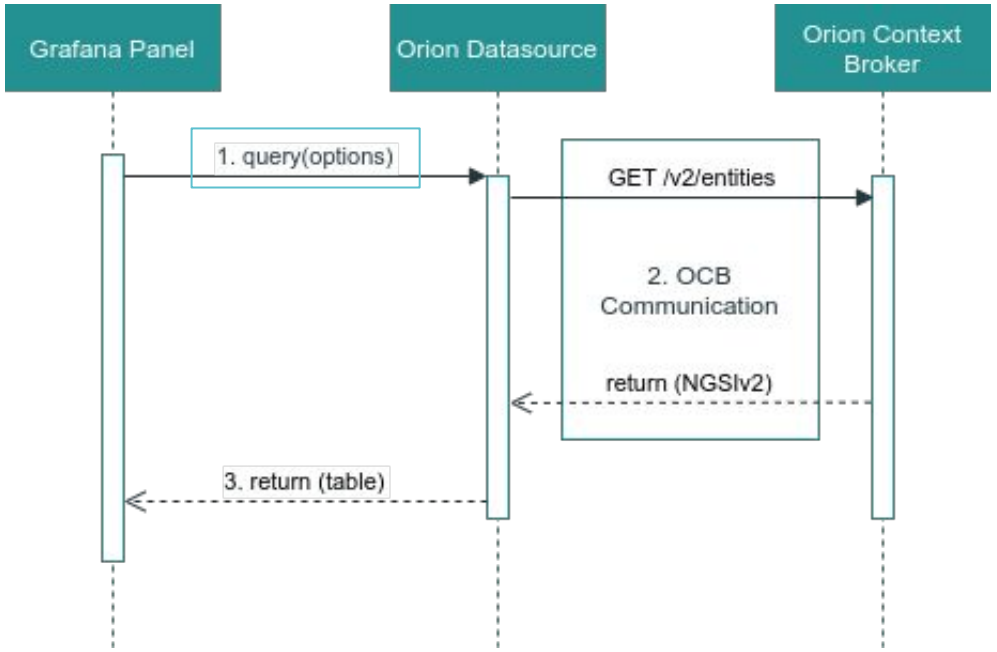
- HOPU has created a FIWARE-based Noise Monitoring System for Valencia
  - IoT Devices → IoT Agent → NGSI-v2 Context Broker
- Custom **Grafana Data Source** via NGSI requests
  - No need for a subscription
  - More Info: https://grafana.com/tutorials/build-a-data-source-plugin/#9
- Altered default look-and-feel according to customer requirements

Real Life Use Cases and Context-Broker-Grafana Custom Data Source

- Allow to connect external data sources/services with Grafana panels.
- Requirements
  - Git
  - Node (nvm)
  - Grafana version < 7.0 AngularJS (TypeScript) https://github.com/grafana/simple-json-datasource
  - Grafana version > 7.0 React (TypeScript) https://grafana.com/tutorials/build-a-data-source-plugin/#3
- Actors
  - Grafana panel
  - Orion Context Broker **Data Source**
  - Orion Context Broker (**NGSI-v2**)
- Sequence diagram overview
  - Grafana panel → Orion Data Source → Context Broker
  - Context Broker → Orion Data Source → Grafana panel



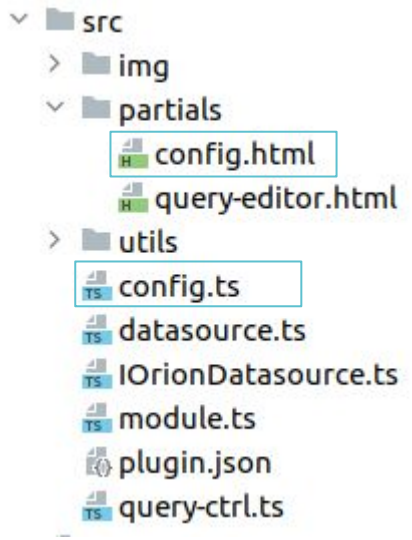
Custom Data Source Sequences
Splitting the problem

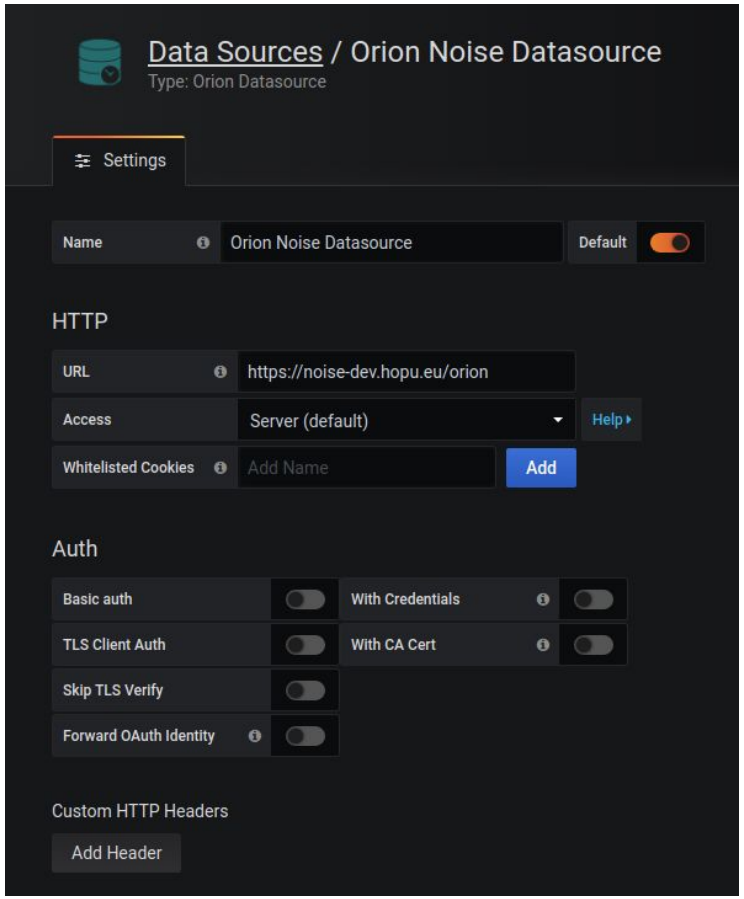


- Query executed by the panel:
  - **ConfigCtrl** contains the data source configuration; security, headers...
  - **QueryCtrl** parameters to build the NGSI-v2 query.
- Context Broker Communications using NGSI interfaces.
- Return data parsed in a Grafana panel understandable format.

Custom Data Source Sequences
1. Query: Using AngularJS to do “the trick”

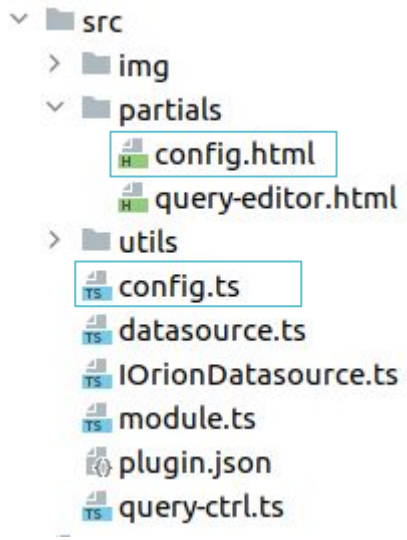
- Config:
  - **view**: represented by an HTML template
  - **controller**: handled by TypeScript logic

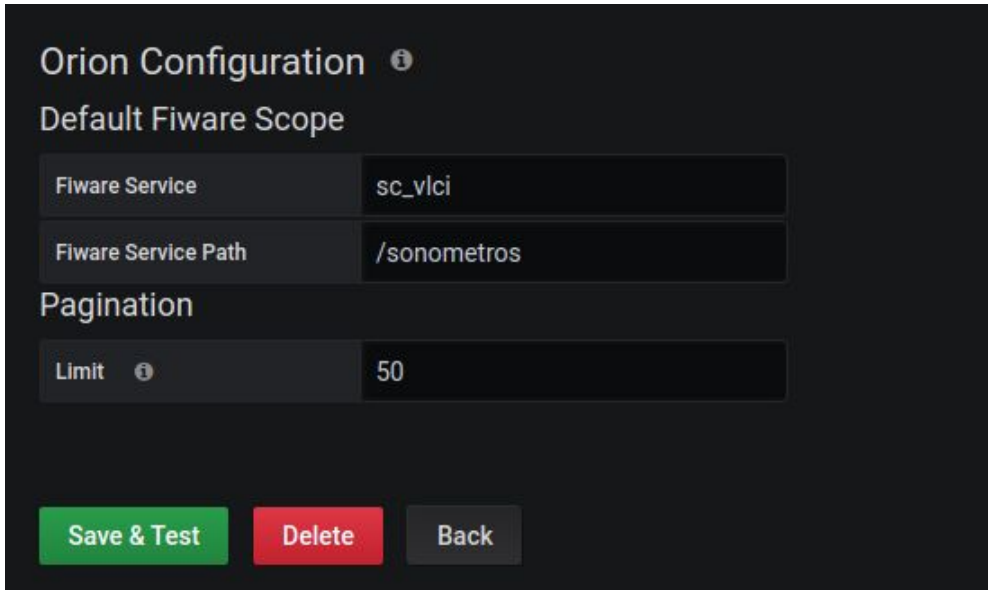




Custom Data Source Sequences
1. Query: Using AngularJS to do “the trick”

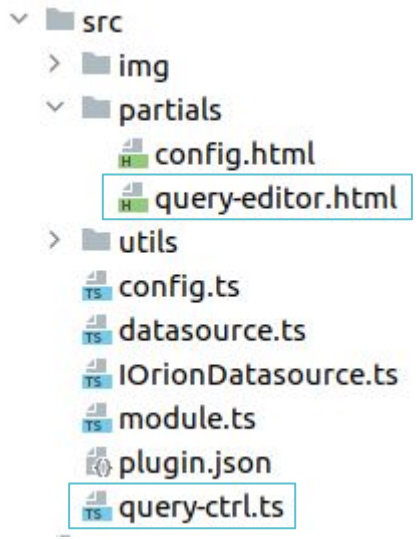
- Config:
  - **view**: represented by an HTML template
  - **controller**: handled by TypeScript logic

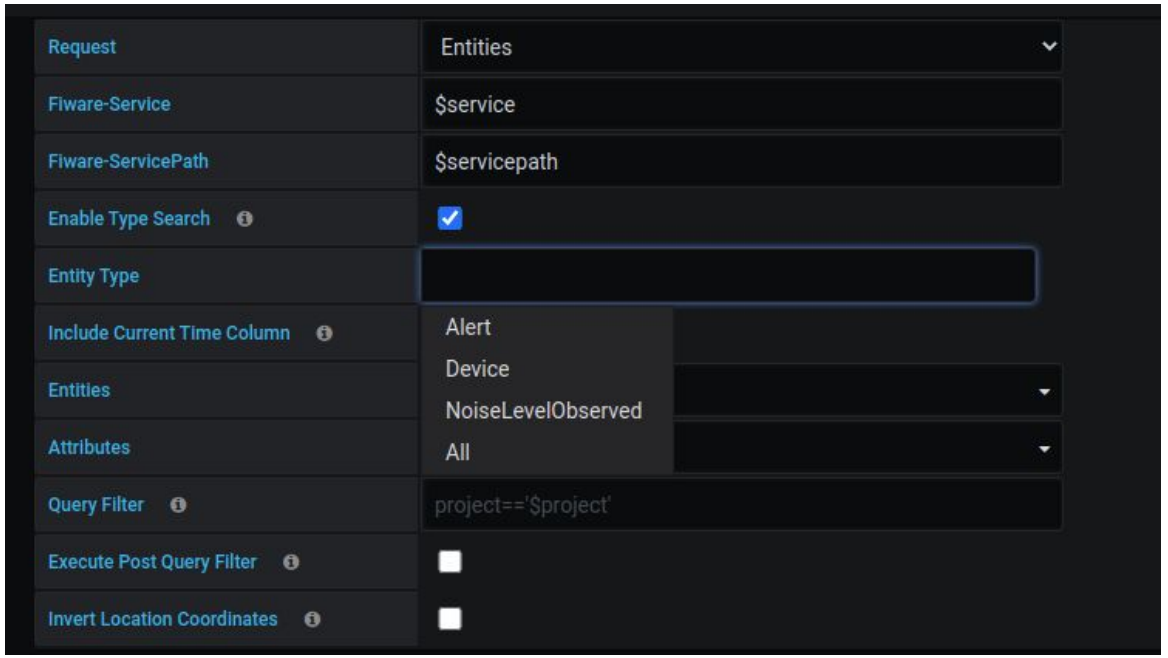




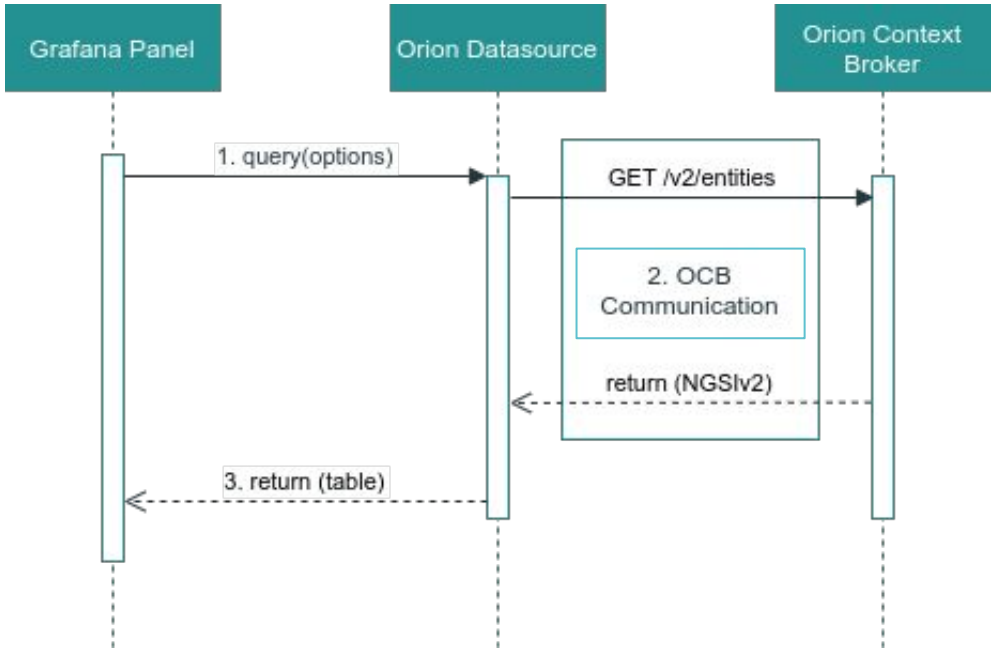
Custom Data Source Sequences
1. Query: Using AngularJS to do “the trick”

- Query:
  - **view**: represented by an HTML template
  - **controller**: handled by TypeScript logic





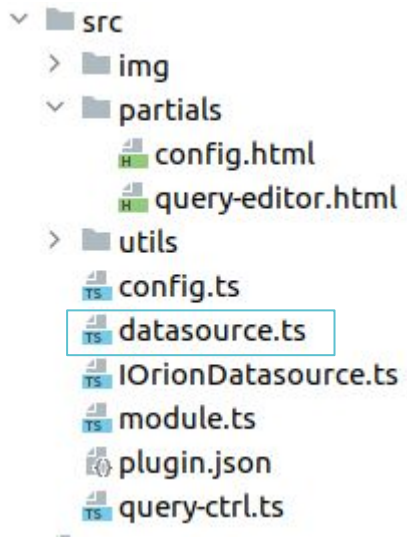
Custom Data Source Sequences
Splitting the problem



- Query executed by the panel:
  - **ConfigCtrl** contains the data source configuration; security, headers...
  - **QueryCtrl** parameters to build the NGSI query.
- Context Broker Communications using NGSI interfaces.
- Return data parsed in a Grafana panel in an understandable format.

Custom Data Source Sequences
2. Context Broker Query Operation

- Orion Context Broker communication:
  - Translating query options to NGSI




```
public query(options: any) {
    const queryData: QueryData = new QueryData(options);
    const orionQuery: Query = OrionQueryFactory.get(
        queryData.request,
        queryData.params
    )
    const orionResponse = orionQuery.execute()

    return Ngsi2Grafana.getTableFormat(orionResponse);
}
```

Custom Data Source Sequences
2. Context Broker Query Operation

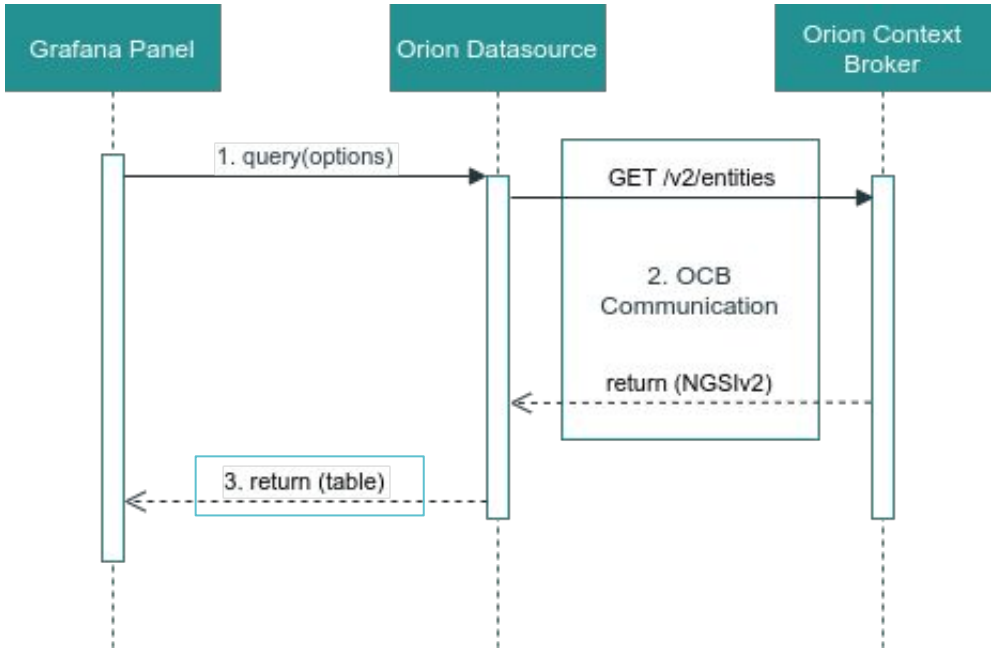
- Orion Context Broker communication:
  - Translating query options to NGSI

```
public query(options: any) {
    const queryData: QueryData = new QueryData(options);
    const orionQuery: Query = OrionQueryFactory.get(
        queryData.request,
        queryData.params
    )
    const orionResponse = orionQuery.execute()

    return Ngsi2Grafana.getTableFormat(orionResponse);
}
```


```
[{
    "id": "NoiseLevelObserved-HOP1",
    "type": "NoiseLevelObserved",
    "name": {
        "type": "Text",
        "value": "Test1",
        "metadata": {}
    },
    "LAeq": {
        "type": "Number",
        "value": 35.6,
        "metadata": {}
    },
    "latitude": {
        "type": "Number",
        "value": 38.077053,
        "metadata": {}
    },
    "longitude": {
        "type": "Number",
        "value": -1.271294,
        "metadata": {}
    }
}]
```

Custom Data Source Sequences
Splitting the problem



- Query executed by the panel:
  - **ConfigCtrl** contains the data source configuration; security, headers...
  - **QueryCtrl** parameters to build the NGSI query.
- Context Broker Communications using NGSI interfaces.
- Return data parsed in a Grafana panel in an understandable format.

Custom Data Source Sequences
3. NGSI v2 to Grafana

- Parsing NGSI format to a Grafana table

```
[{
    "id": "NoiseLevelObserved-HOP1",
    "type": "NoiseLevelObserved",
    "name": {
        "type": "Text",
        "value": "Test1",
        "metadata": {}
    },
    "LAeq": {
        "type": "Number",
        "value": 35.6,
        "metadata": {}
    },
    "latitude": {
        "type": "Number",
        "value": 38.077053,
        "metadata": {}
    },
    "longitude": {
        "type": "Number",
        "value": -1.271294,
        "metadata": {}
    }
}]
```

```
public query(options: any) {
    const queryData: QueryData = new QueryData(options);
    const orionQuery: Query = OrionQueryFactory.get(
        queryData.request,
        queryData.params
    )
    const orionResponse = orionQuery.execute()

    return Ngsi2Grafana.getTableFormat(orionResponse);
}
```

Custom Data Source Sequences
3. NGSI v2 to Grafana

- Parsing NGSI format to a Grafana table

```
[{
    "id": "NoiseLevelObserved-HOP1",
    "type": "NoiseLevelObserved",
    "name": {
        "type": "Text",
        "value": "Test1",
        "metadata": {}
    },
    "LAeq": {
        "type": "Number",
        "value": 35.6,
        "metadata": {}
    },
    "latitude": {
        "type": "Number",
        "value": 38.077053,
        "metadata": {}
    },
    "longitude": {
        "type": "Number",
        "value": -1.271294,
        "metadata": {}
    }
}]
```

```
{
    "columns": [
        {"text": "id"},
        {"text": "type"},
        {"text": "LAeq"},
        {"text": "latitude"}
    ,{"text": "longitude"},
        {"text": "name"},
        {"text": "time"}
    ],
    "rows": [
        [
            "NoiseLevelObserved-HOP1",
            "NoiseLevelObserved",
            35.6, 38.077054,
            -1.271294,
            "Test1",
            1612177614800]
    ],
    "type": "table"
}
```

Custom Data Source
Final result

servicepath /sonometros Dispositivos Sensor 02 - pub el Asesino

id	type	location	alarms	LAeq	operationalStatus	name	project	time
NoiseLevelObserved-HOP8	NoiseLevelObserved	39.47398, -0.34443	-	60.00	CONNECTED	Sensor 08 - Honduras 3	impulsoVLCi-red.es	1.61 Tri
NoiseLevelObserved-HOP7	NoiseLevelObserved	39.4739, -0.34455	-	56.00	CONNECTED	Sensor 07 - Honduras test2	impulsoVLCi-red.es	1.61 Tri

FIWARE
Global
Summit

Thanks!

Vienna, Austria
12-13 June, 2023
#FIWARESummit

FIWARE
Open APIs for Open Minds