

Smart City - Tarea 2: Construcción infraestructura FIWARE

Vamos a construir una solución completa de gestión de datos de IoT.

1. Configuración del entorno FIWARE:

- Utilizaremos **Docker Compose** para levantar los servicios necesarios:
 - orion: El Orion Context Broker.
 - mongo: La base de datos MongoDB para el estado actual.
 - quantumleap: El componente para el histórico.
 - cratedb: La base de datos para el histórico de datos.
- Asegúrate de que todos los contenedores se comuniquen correctamente a través de la red de Docker.

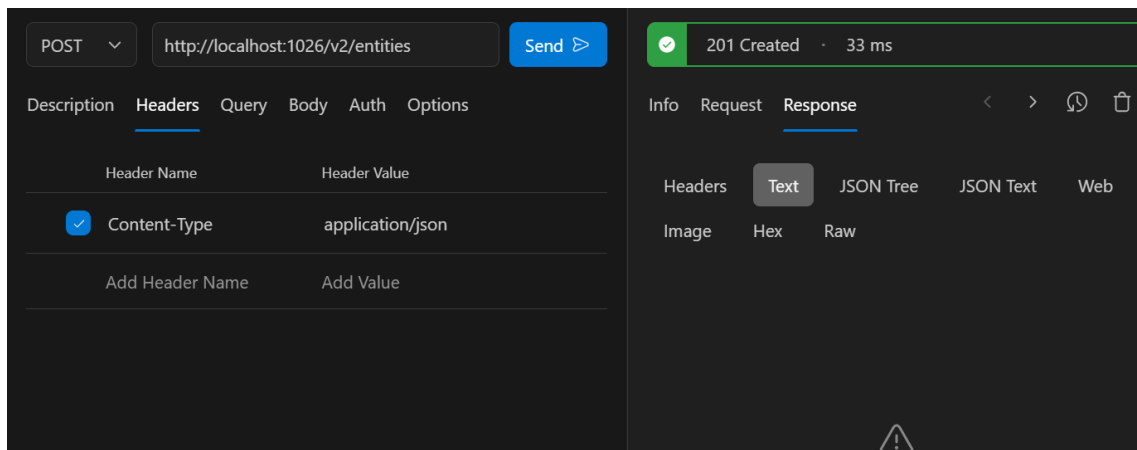
2. Creación de las 3 entidades:

- Usando la **API REST de Orion Context Broker**, crearemos las 3 entidades que definimos en la *Tarea 1: Diseño del ADN de los Sensores*.
- La llamada API enviará un POST a `/v2/entities` con el JSON de cada entidad.

Lo he realizado con RapidAPIClient, en el post recordar poner los Headers en este caso es:

Header: Content-Type: application/json

Con el 201 devuelve que ha sido creado.



Y en el body con formato json ponemos cada entidad que esta subida en formato json:

```
{
  "id": "SensorC0201",
  "type": "SensorC02",
  "dateObserved": {
```

```

    "type": "DateTime",
    "value": "2025-07-30T18:00:00-05:00"
  },
  "CO2": {
    "type": "Number",
    "value": 0
  }
}

```

Con el GET, podemos comprobar que está realizado correctamente los posts:

The screenshot shows a web client interface with the following details:

- Method:** GET
- URL:** http://localhost:1026/v2/entities/SensorTemp1
- Status:** 200 OK
- Time:** 26 ms
- Response Body (JSON Text):**

```

1 {
2   "id": "SensorTemp1",
3   "type": "SensorTempHum",
4   "dateObserved": {
5     "type": "DateTime",
6     "value": "2025-07-30T23:00:00.000Z",
7     "metadata": {}
8   },
9   "humedad": {
10    "type": "Number",
11    "value": 80,
12    "metadata": {}
13  },

```

También puedo realizar un get desde la consola Powershell para comprobar que esta subido con : `Invoke-WebRequest -Uri "http://localhost:1026/v2/entities" -Method GET`

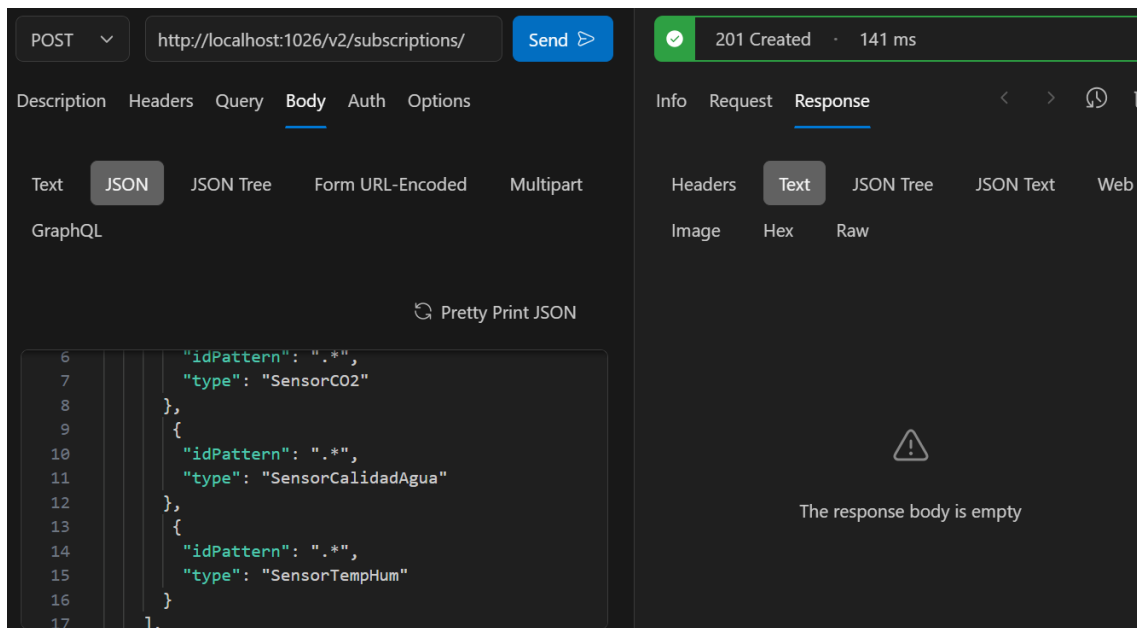
3. Creación de una suscripción:

- Esta es la parte clave para el histórico. Crearemos una **suscripción** en Orion que "escucha" cualquier cambio en nuestras entidades.

Crearé primero el json con la suscripción, que en mi caso es el archivo subscriptions.json adjuntado en el repositorio

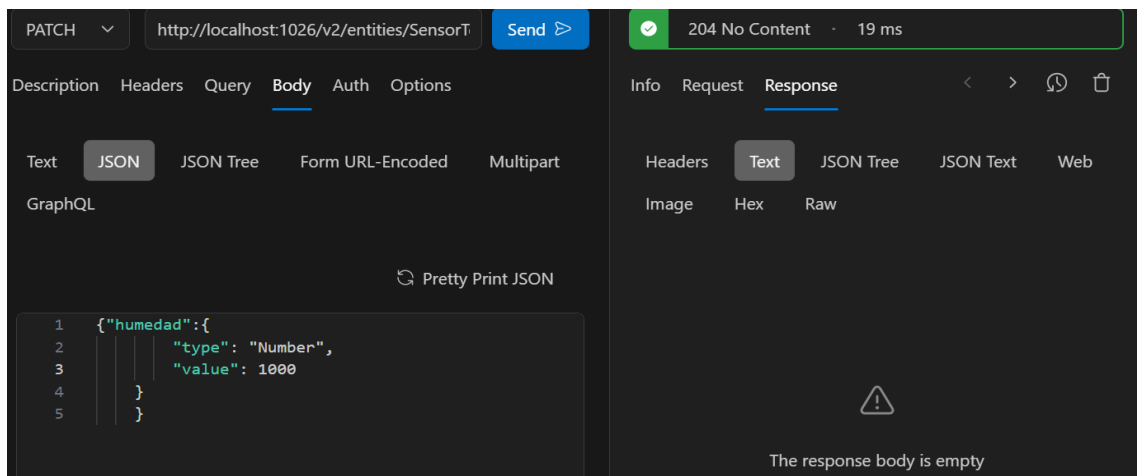
- Cuando Orion detecte una actualización, enviará automáticamente una copia de la entidad modificada al servicio **QuantumLeap**.
- La suscripción se crea con un POST a /v2/subscriptions.

Ahora haré un post con su cabecera correspondiente y en el body copiando y pegando del json creado en subscriptions.json



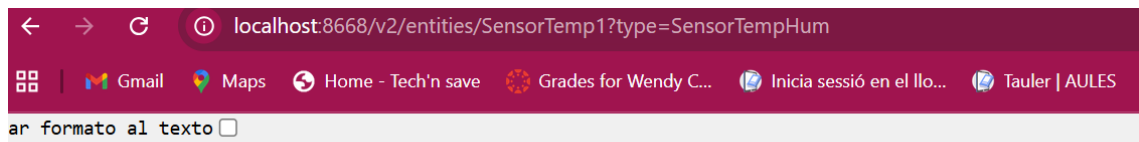
Para comprobarlo voy a realizar un patch, recordar poner en headers-> Content type: application/json y en el body el formato json lo que se visualiza en la imagen, y la ruta:

<http://localhost:1026/v2/entities/SensorTemp1/attrs>



Lo compruebo en el QuantumLeap, con el enlace

<http://localhost:8668/v2/entities/SensorTemp1?type=SensorTempHum>



```
"attributes": [
  {
    "attrName": "dateObserved",
    "values": [
      "2025-07-30T23:00:00.000+00:00",
      "2025-07-30T23:00:00.000+00:00"
    ]
  },
  {
    "attrName": "humedad",
    "values": [
      90.0,
      1000.0
    ]
  },
  {
    "attrName": "temperature",
    "values": [
      25.0,
      25.0
    ]
  }
],
"entityId": "SensorTemp1",
"entityType": "SensorTempHum",
"index": [
  "2025-10-30T16:27:18.103+00:00",
  "2025-10-30T16:28:56.512+00:00"
]
```

4. Carga de datos:

- Ahora viene la carga de los datos masivos. Utilizaremos un script de Python con la API de Orion.

Primeramente, he comprobado que tenía instalado

`pip install requests`

Para poder ejecutar el script, una vez instalado he generado el script:

`loader.ipynb`

- El script deberá **enviar 400 actualizaciones por atributo** a la API de Orion para cada una de nuestras 3 entidades.

En este caso como son 3 entidades y en total 6 atributos se han creado 2400 actualizaciones con un retardo de 0,1 segundos

- Cada vez que el script envíe una actualización, Orion hará dos cosas:
 - Actualizará el documento en **MongoDB** con el nuevo estado.
 - Debido a la suscripción, enviará una copia de los datos a **QuantumLeap**, que a su vez los insertará en **CrateDB** como una nueva entrada de serie de tiempo.

En el CrateDB, se puede comprobar que se ha generado bien las 2400 actualizaciones con el script creado en Python, con el retardo de 0,1 sec.

Filtrar tablas...	etsensorcalidadagua	good
▼ Doc Tablas	Réplicas configuradas	Fragmentos configurados
etsensorcalidadagua 1,164 Registros (570.5 KiB) 4 Fragmentos / 2-all Réplicas	2-all	4
Tablas	Fragmentos iniciados	Fragmentos perdidos
etsensorco2 399 Registros (272.2 KiB) 4 Fragmentos / 2-all Réplicas	4	0
etsensortempum 795 Registros (340.6 KiB) 4 Fragmentos / 2-all Réplicas	Fragmentos no replicados	Registros totales
	0	1,164
	Registros no disponibles	Registros no replicados
	0	0

CrateDB						
Nodos: 1 Estado: Datos Chequeos Carga del sistema: 0.03/0.03/0.06 crate						
Consultar CrateDB con SQL						
<input checked="" type="checkbox"/> Formatear resultados <input checked="" type="checkbox"/> Almacenar historial de consultas <input type="checkbox"/> Ver traza de error <input type="button" value="Borrar historial"/>						
<pre>SELECT entity_id, entity_type, time_index, fiware_servicepath, __original_ngsi_entity__, instanceid, dateobserved, ph, temperature, cloro FROM "doc"."etsensorcalidadagua" LIMIT 100;</pre>						
ENVIAR CONSULTA <input type="button" value="Truco: Pulsa ⬆ + ⬆ para enviar consulta"/> SELECT OK, 100 registros devueltos (0.011 segundos)						
Resultado de consulta						
entity_id	entity_type	time_index	fiware_servicepath	__original_ngsi_entity__	instanceid	dateobserved
SensorCalidad1	SensorCalidadAgua	1761846638180 (2025-10-30T17:50:38.180Z)	/	NULL	urn:ngsi-Id:bba74d7b-77bc-4958-a844-3766e2f651ad	1753916400000 (2025-10-30T17:50:38.180Z)
SensorCalidad1	SensorCalidadAgua	1761846638885 (2025-10-30T17:50:38.885Z)	/	NULL	urn:ngsi-Id:c2cafecb-3d4e-4bb2-	1753916400000 (2025-10-30T17:50:38.885Z)

5. Consulta Mongodb

- Realiza una captura de pantalla con la consulta Mongodb a la coleccion de entidades en la que se vean todos los atributos de las tres entidades

Primeramente, en mongo db en la consola tengo que comprobar que este usando Orion con el comando :

use orion

Después comprobar que estén las 3 entidades con:

db.entities.find()

```
> db.entities.find()
< {
  _id: {
    id: 'SensorTemp1',
    type: 'SensorTempHum',
    servicePath: '/'
  },
  attrNames: [
    'dateObserved',
    'temperature',
    'humedad'
  ],
  attrs: {
    dateObserved: {
      value: 1753916400,
      type: 'DateTime',
      creDate: 1761237743.7678304,
      modDate: 1761237743.7678304,
      mdNames: []
    },
    temperature: {
      value: 33.7,
      type: 'Number',
      creDate: 1761237743.7678304,
      modDate: 1761846863.4846869,
      mdNames: []
    },
    humedad: {
      value: 84,
      type: 'Number',
      creDate: 1761237743.7678304,
      modDate: 1761846863.5983565,
      mdNames: []
    }
  },
  creDate: 1761237743.7678304,
  modDate: 1761846863.5992937,
  lastCorrelator: '7809c750-b5b9-11f0-9868-9e95666ae9e0'
}
{
  _id: {
    id: 'SensorC0201',
    type: 'SensorC02',
```

```
      modDate: 1761238183.6451967,  
      mdNames: []  
    },  
    CO2: {  
      value: 923,  
      type: 'Number',  
      creDate: 1761238183.6451967,  
      modDate: 1761846863.0286956,  
      mdNames: []  
    }  
  },  
  creDate: 1761238183.6451967,  
  modDate: 1761846863.0295124,  
  lastCorrelator: '77b2d684-b5b9-11f0-9171-9e95666ae9e0'  
}  
{  
  _id: {  
    id: 'SensorCalidad1',  
    type: 'SensorCalidadAgua',  
    servicePath: '/'  
  },  
  attrNames: [  
    'dateObserved',
```

```
▶ ▼ _id: Object
  id: "SensorC0201"
  type: "SensorC02"
  servicePath: "/"
  ▼ attrNames: Array (2)
    0: "dateObserved"
    1: "C02"
  ▼ attrs: Object
    ▶ dateObserved: Object
    ▶ C02: Object
  createDate: 1761238183.6451967
  modDate: 1761846863.0295124
  lastCorrelator: "77b2d684-b5b9-11f0-9171-9e95666ae9e0"
```

```
▼ _id: Object
  id: "SensorCalidad1"
  type: "SensorCalidadAgua"
  servicePath: "/"
  ▼ attrNames: Array (4)
    0: "dateObserved"
    1: "ph"
    2: "temperature"
    3: "cloro"
  ▼ attrs: Object
    ▶ dateObserved: Object
    ▶ ph: Object
    ▶ temperature: Object
    ▶ cloro: Object
  createDate: 1761238806.1343844
  modDate: 1761846863.3737824
  lastCorrelator: "77e729fc-b5b9-11f0-8a4c-9e95666ae9e0"
```

```
☑ ▼ _id: Object
  id: "SensorTemp1"
  type: "SensorTempHum"
  servicePath: "/"
  ▼ attrNames: Array (3)
    0: "dateObserved"
    1: "temperature"
    2: "humedad"
  ▼ attrs: Object
    ▶ dateObserved: Object
    ▶ temperature: Object
    ▶ humedad: Object
  createDate: 1761237743.7678304
  modDate: 1761846863.5992937
  lastCorrelator: "7809c750-b5b9-11f0-9868-9e95666ae9e0"
```