AUMENTANDO LA INTEROPERABILIDAD DE UNA RED DE MONITORIZACIÓN DE CALIDAD DEL AIRE MEDIANTE ESTÁNDARES WEB DE IOT

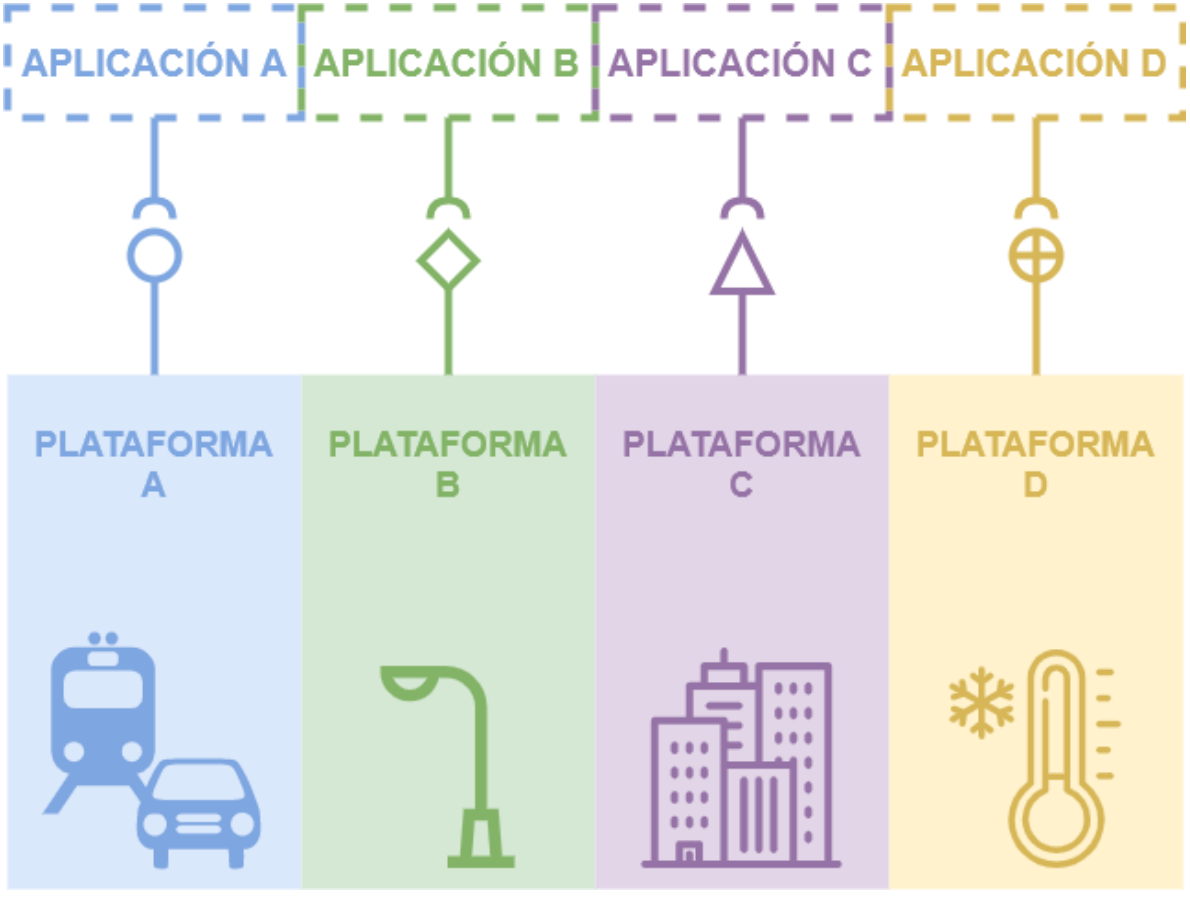
Sergi Trilles, *Jesús Velayos, José Ivan San José, José Carlos Gamazo*

Universidad Isabel I

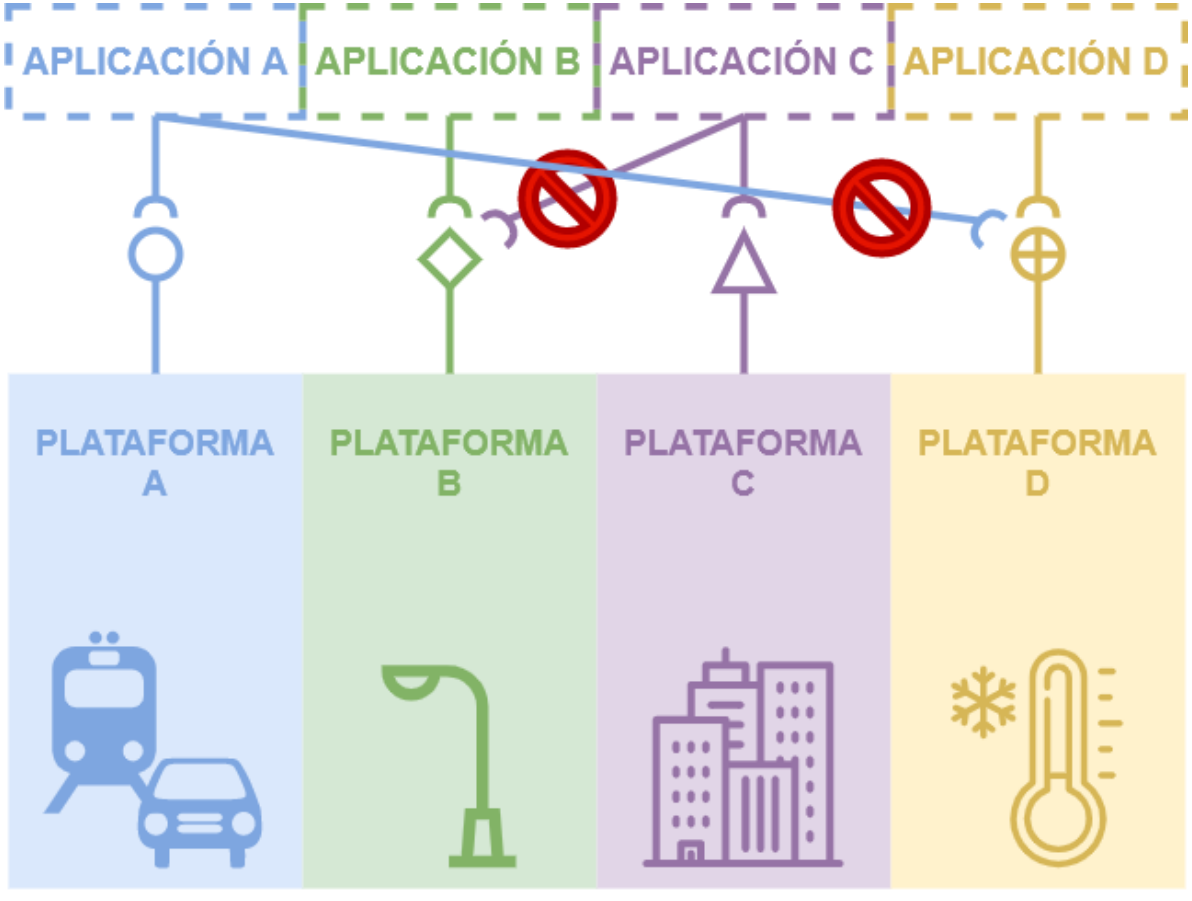
Índice de contenidos

1. Introducción
2. *OGC SensorThings API*
3. Red de monitorización de calidad del aire de Madrid
4. Arquitectura propuesta
5. Implementación de la arquitectura
6. Conclusiones

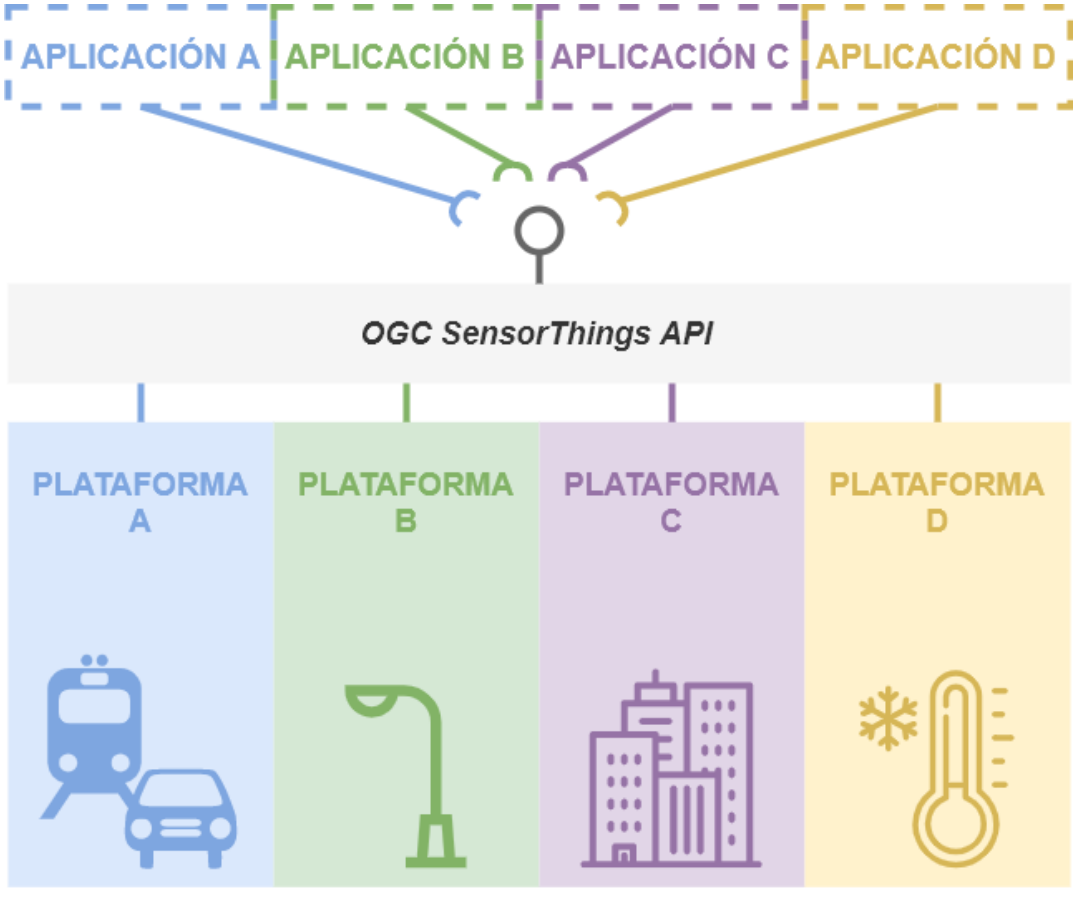
Introducción (I)



Introducción (II)



OGC SensorThings API (I)



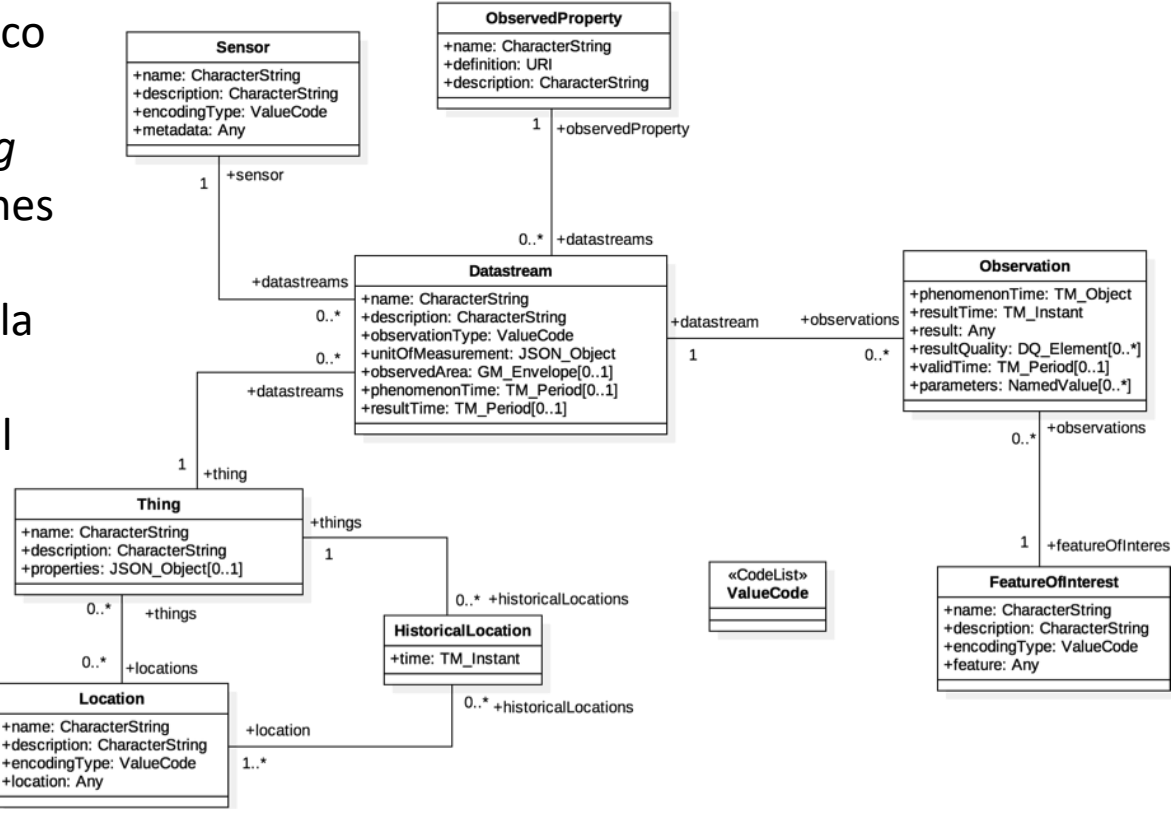
OGC SensorThings API (II)

- Estándar *OGC* desde 2016
- Pensado para dar soporte a *IoT*
- Inspirado en *OGC Sensor Web Enablement* (*SWE*)
- Dos perfiles
  - *Sensing* aprobado en 2016
  - *Tasking* aprobado en 2019
- Interfaz RESTful (*HTTP*) *CREATE, READ, UPDATE y DELETE*
- Soporte *MQTT y CoAP*
- Formato de codificación *JSON*

OGC SensorThings API (III)

- *Sensing Core* (basado en *O&M*) (I)

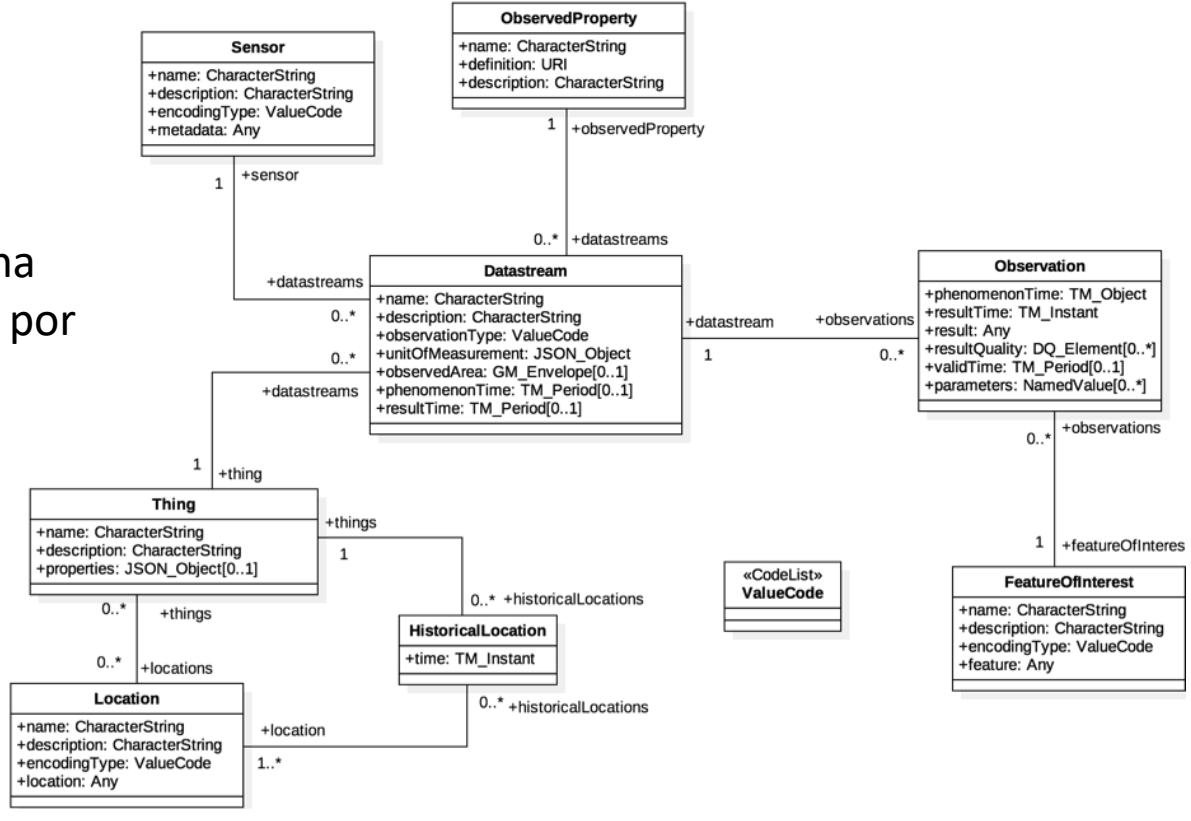
Thing: un objeto del mundo físico o virtual
Locations: localización del *Thing*
HistoricalLocations: localizaciones históricas
Datastream: observaciones de la misma propiedad y sensor
ObservedProperty: especifica el fenómeno de una observación
Sensor: un instrumento que observa una propiedad o fenómeno



OGC SensorThings API (III)

- *Sensing Core* (basado en *O&M*) (II)

Observation: acto de medir y determinar el valor de una propiedad
FeatureOfInterest: una observación se describe con una característica de interés, como por ejemplo el lugar



OGC SensorThings API (III)

- *Sensing Core* (basado en *O&M*) (III)

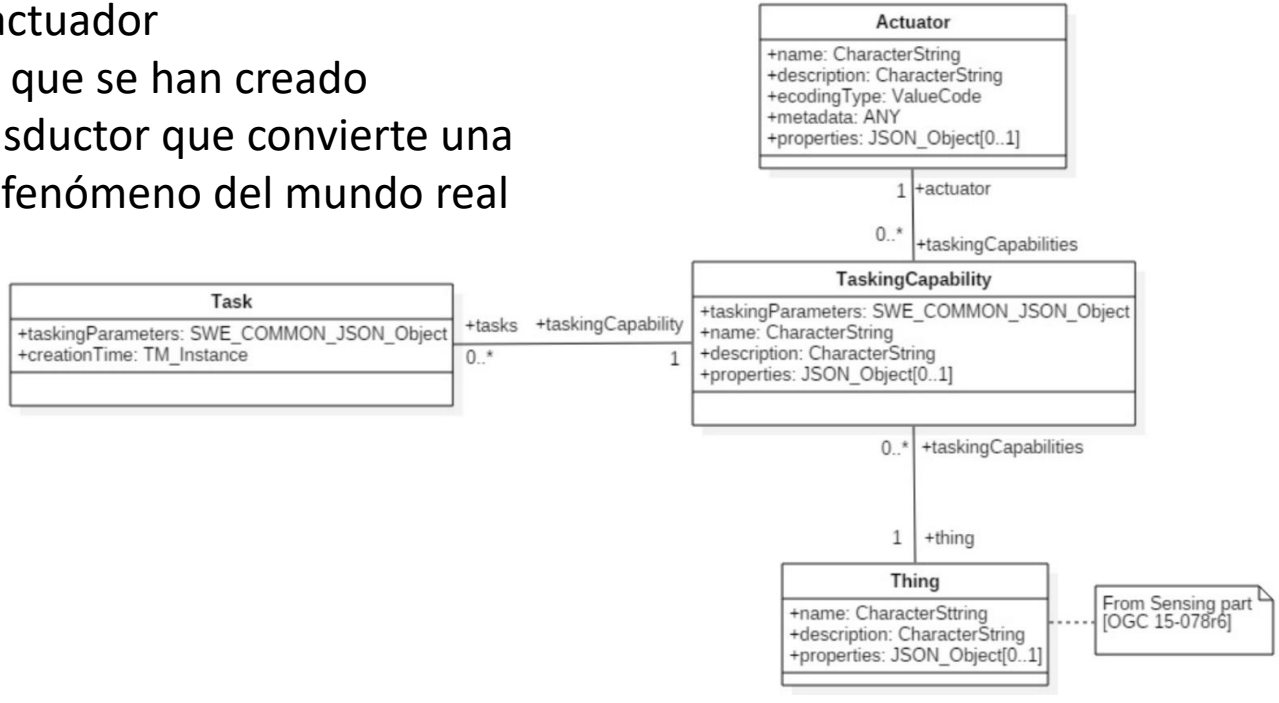
http://example.org/v1.0/Datastream(id)/Observations


```
{
  "@iot.count": 2,
  "value": [
    {
      "@iot.id": 1,
      "@iot.selfLink": "http://example.org/v1.0/Observations(1)",
      "phenomenonTime": "2016-01-01T05:00:00.000Z",
      "result": "-9",
      "resultTime": null,
      "Datastream@iot.navigationLink": "http://example.org/v1.0/Observations(1)/Datastream",
      "FeatureOfInterest@iot.navigationLink":
"http://example.org/v1.0/Observations(1)/FeatureOfInterest"
    },
    {
      "@iot.id": 2,
      "@iot.selfLink": "http://example.org/v1.0/Observations(2)",
      "phenomenonTime": "2016-01-01T04:00:00.000Z",
      "result": "-10",
      "resultTime": null,
      "Datastream@iot.navigationLink": "http://example.org/v1.0/Observations(2)/Datastream",
      "FeatureOfInterest@iot.navigationLink":
"http://example.org/v1.0/Observations(2)/FeatureOfInterest"
    }
  ]
}
```

OGC SensorThings API (IV)

- *Tasking Core*

TaskingCapabilities: especifica los parámetros aptos para tareas de un actuador
Task: colección de tareas que se han creado
Actuator: un tipo de transductor que convierte una señal en alguna acción o fenómeno del mundo real

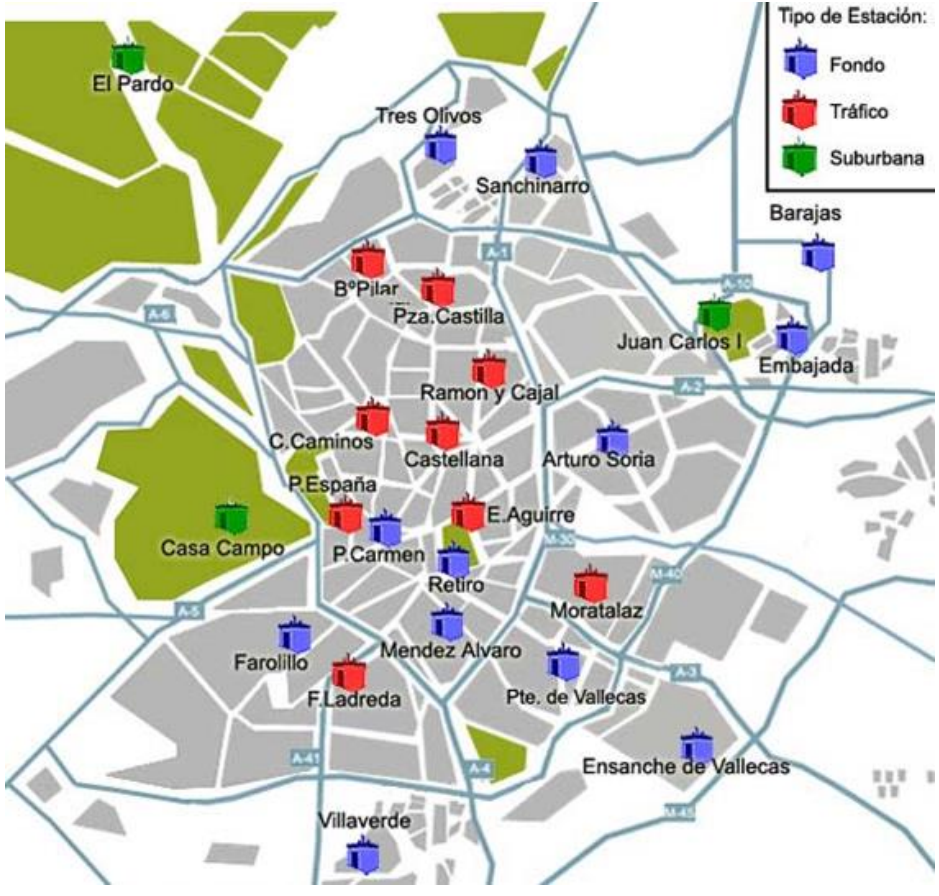


OGC SensorThings API (V)

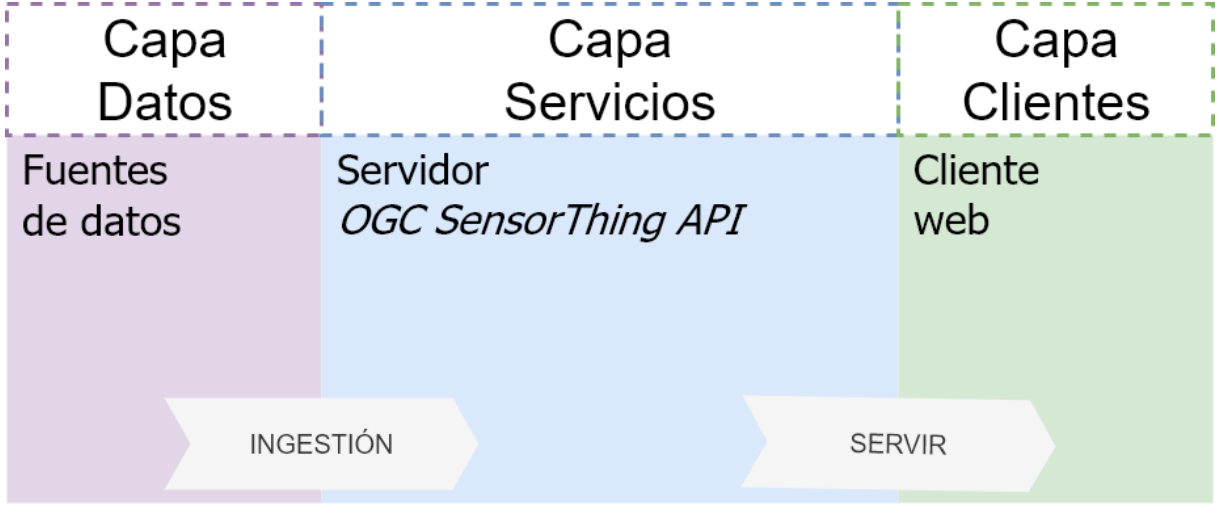
- *Implementaciones:*
  - ***GOST***: GO Sensor Things (*GOST*) fue desarrollado por Geodan utilizando el lenguaje de programación Golang (Go). El desarrollo de la implementación aún está en curso, pero presenta la funcionalidad completa *OGC SensorThings API Sensing* e incluye un *dashboard* para la visualización de datos
  - ***Mozilla-SensorThings***: *Mozilla* comenzó a desarrollar una implementación de *OGC SensorThings API*, pero el proyecto se ha estancado desde febrero de 2017
  - ***FROST***: Fraunhofer Open-Source Sensor Things (*FROST*) fue desarrollado por Fraunhofer IOSB en Java. Los desarrolladores de *FROST* continúan trabajando en el proyecto para ampliar sus características y completar el estándar *OGC SensorThings API*.

Red de calidad del aire de Madrid

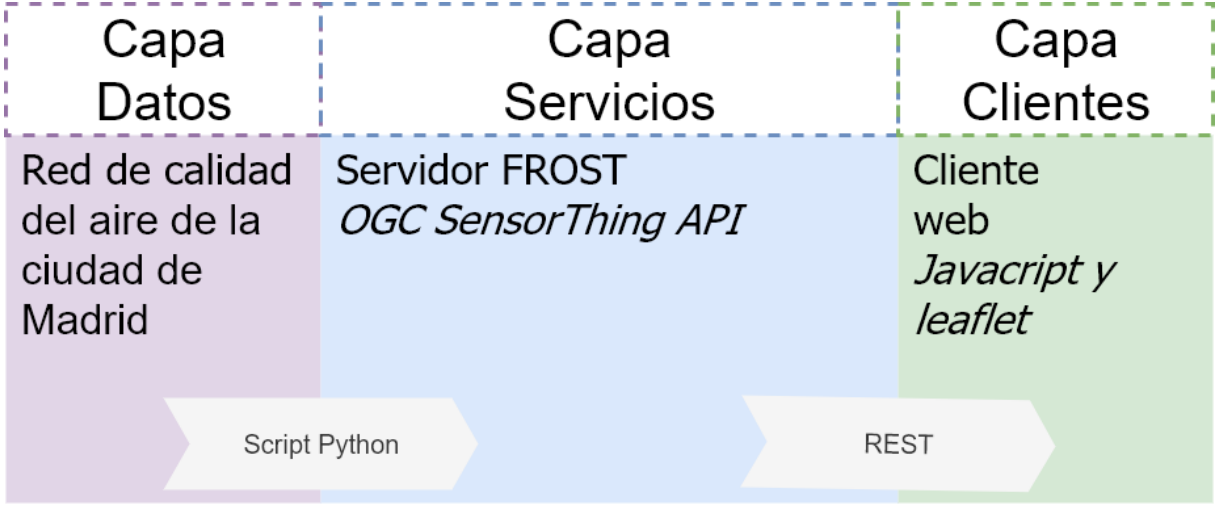
1. **24 estaciones** automáticas
2. **Material Particulado**, **Óxidos de Nitrógeno** (NO y NO2), **Ozono** o **Benceno**, entre otros
3. ***Urbanas de fondo*** *(población urbana en general),* ***De tráfico*** *(cerca de carreteras) y* ***Suburbanas*** *(a las afueras)*
4. **1 hora** de frecuencia
5. Publicadas en el ***open data***
6. Formato **CSV**



Arquitectura propuesta



Desarrollo de la arquitectura (I)



Instancia EC2 en Amazon Web Services

Desarrollo de la arquitectura (II)

FASE 0:

- Descripción de *Things* y *Datastrems (script Python)*

Ejemplo Thing

```
{
  "name": "Pza. de Espanya",
  "description": "Plaza de Espanya",
  "Locations": [
    {
      "name": "Pza. de Espanya",
      "description": "4 - Pza. de Espanya",
      "encodingType": "application/vnd.geo+json",
      "location": {
        "type": "Point",
        "coordinates": [
          -0.0000037122567,
          0.000040423882299999997
        ]
      }
    }
  ]
}
```

Desarrollo de la arquitectura (III)

Ejemplo Datastrems

```
Datastream:
[
  {
    "name": "Dioxido de carbono",
    "description": "Concentracion de dioxido de carbono en el aire",
    "observationType":  "http://www.opengis.net/def/observationType/OGC-
OM/2.0/OM_Measurement",
    "unitOfMeasurement": {
      "name": "µg/m3",
      "symbol": "SO2"
    },
    "Sensor": {
      "name": "Sensor",
      "description": "sensor dióxido de carbono",
      "encodingType": "application/pdf",
      "metadata":
"https://www.sparkfun.com/datasheets/Sensors/Temperature/DHT22.pdf"
    },
    "ObservedProperty": {
      "name": "Dioxido de carbono",
      "definition":
"https://es.wikipedia.org/wiki/Di%C3%B3xido_de_carbono",
      "description": "Concentracion de dioxido de carbono en el aire"
    }
  },
  …
]
```

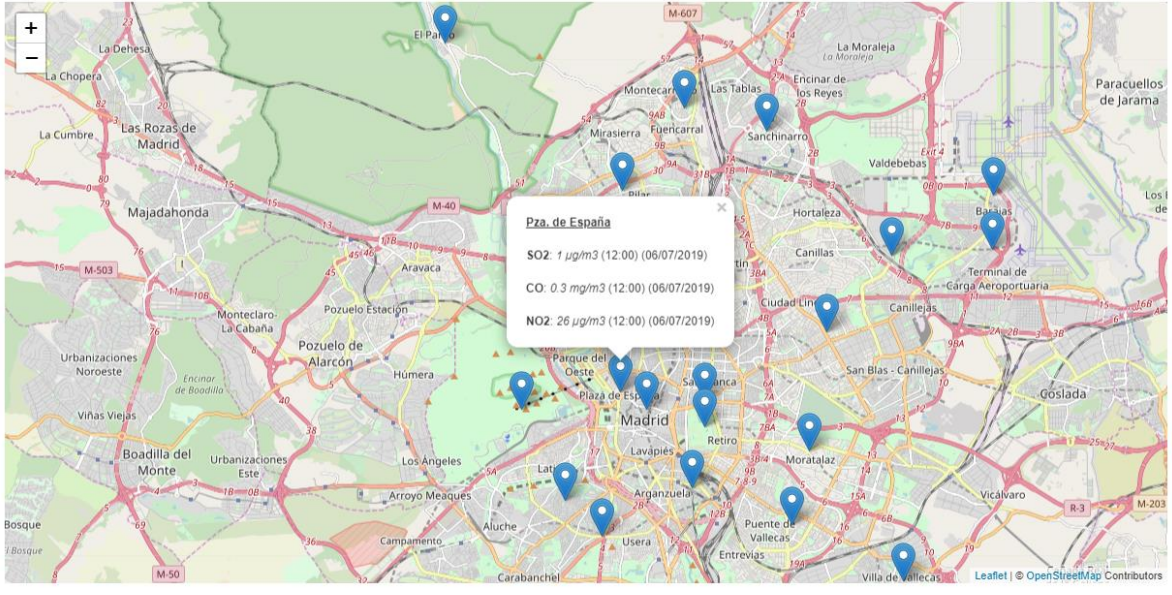
Desarrollo de la arquitectura (IV)

FASE 1:

- Inserción en tiempo real de las observaciones *(script python)*
- Se ejecutará cada hora justamente después de la actualización del *CSV* del *open data* (minutos 20-30)
- Por cada observación se realizará una petición *POST* de creación sobre el *datastream* de cada estación y fenómeno

```
{
  "resultTime": "2019-01-14T12:35:47.000Z",
  "result": 0.327
}
```

Desarrollo de la arquitectura (V)



1. **Cliente *responsive*** utilizando Javascript y Leaflet
2. Muestra **último valor**
3. **Interoperable** con el estándar

Conclusiones

1. **Validación del estándar** *OGC SensorThings API* para la gestión de datos de dispositivos *IoT* en el ámbito de la monitorización ambiental
2. **Arquitectura agnóstica** a cualquier ámbito de aplicación, por lo que favorece su **reusabilidad**
3. *OGC SensorThings API se* caracteriza por su **flexibilidad, facilidad y agilidad** en la publicación y consumo de los datos, al utilizar ***REST*** y ***JSON***
4. Aumento de la **interoperabilidad** de los datos de dispositivos IoT

Gracias por su atención

correo: *sergio.trilles@ui1.es*
web: *http://www3.uji.es/~strilles*