UNIVERSITAT POLITÈCNICA DE VALÈNCIA

DISCA
DEPARTAMENTO DE INFORMÁTICA DE SISTEMAS Y COMPUTADORES

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Dpto. de Informática de Sistemas y Computadores

# Capacidades de interconexión de FIWARE con ecosistemas IoT

Trabajo Fin de Máster

Máster Universitario en Ingeniería de Computadores y Redes

AUTOR/A: Madiouni , Eyateleh

Tutor/a: Campelo Rivadulla, José Carlos

Cotutor/a: Bonastre Pina, Alberto Miguel

CURSO ACADÉMICO: 2022/2023

UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Máster en Ingeniería de Computadores y Redes

Trabajo Fin de Máster

# Capacidades de interconexión de FIWARE con ecosistemas IoT


Autor: **Eyateleh Madiouni**

tutores : **Alberto Miguel Bonastre Pina ,**
**José Carlos Campelo Rivadulla**



2022/2023

# Agradecimientos

En primer lugar, mi más sincero agradecimiento a **DIOS** que tanto me ha guiado y me ha protegido .

**A mis queridos padres, JALEL y EMNA**

Gracias a sus ánimos y sacrificios, han sabido crear un clima de afecto propicio para la realización de mis estudios.

Ninguna dedicatoria podría expresar mi respeto, consideración y profundos sentimientos hacia ellos.

Ruego a Dios que los bendiga y los cuide, esperando que siempre estén orgullosos de mí.

**A mis queridos hermanos y hermanas**

Nunca podré agradecerles lo suficiente su amabilidad, su generosidad y su inestimable ayuda.

**A mis tutores**

Gracias por su dirección y tiempo.

Sin su apoyo, este trabajo no habría sido posible. A todos ustedes, **¡muchas gracias!**

# Resumen

Internet of Things (IoT) ha experimentado una rápida evolución en los últimos años. Al principio, se trataba principalmente de conectar dispositivos y recopilar datos, pero ha evolucionado hacia la creación de ecosistemas inteligentes y conectados que brindan servicios y experiencias más avanzadas.

En el ámbito del IoT, han surgido diversas plataformas que facilitan el desarrollo de aplicaciones y servicios conectados. Estas plataformas proporcionan herramientas y recursos para la conectividad, procesamiento de datos, análisis y gestión de dispositivos. Permiten a los desarrolladores crear soluciones IoT más eficientes y escalables.

Una de estas plataformas es FIWARE, que se destaca por ser una plataforma de código abierto y basada en estándares abiertos. FIWARE ofrece un conjunto de componentes y servicios que permiten la interconexión y el procesamiento de datos en tiempo real. Proporciona una arquitectura flexible y modular, lo que facilita la integración con diferentes tecnologías y sistemas existentes.

FIWARE se enfoca en la gestión de datos y la interoperabilidad. Permite la recopilación de datos desde diferentes fuentes y su procesamiento en la nube, lo que habilita la toma de decisiones basadas en información actualizada. Además, FIWARE permite la interoperabilidad entre dispositivos y servicios, lo que facilita la creación de soluciones completas y escalables.

Fiware, impulsada por la Unión Europea para el desarrollo y despliegue de aplicaciones de Internet del futuro, es una de las plataformas que se pueden explotar en el marco de Internet de las cosas. Empresas y organismos han desarrollado, en el marco de las smart cities, smart agrifood o smart industry, iniciativas para su uso y generación de estándares. En este contexto, el trabajo tiene como objetivo poner en marcha, implementar y evaluar las capacidades de Fiware para interconectarse con otras plataformas ya existentes y poder requerir y suministrar la información a éstas.

# Abstract

The Internet of Things (IoT) has undergone a rapid evolution in recent years. Initially, it was mainly about connecting devices and collecting data, but it has evolved into the creation of smart, connected ecosystems that provide more advanced services and experiences.

In the IoT arena, a number of platforms have emerged that facilitate the development of connected applications and services. These platforms provide tools and resources for connectivity, data processing, analytics and device management. They enable developers to create more efficient and scalable IoT solutions.

One such platform is FIWARE, which stands out as an open source and open standards-based platform. FIWARE offers a set of components and services that enable interconnection and real-time data processing. It provides a flexible and modular architecture, which facilitates integration with different technologies and existing systems.

FIWARE focuses on data management and interoperability. It enables the collection of data from different sources and its processing in the cloud, which enables decision making based on up-to-date information. In addition, FIWARE enables interoperability between devices and services, which facilitates the creation of complete and scalable solutions.

Fiware, promoted by the European Union for the development and deployment of future Internet applications, is one of the platforms that can be exploited within the framework of the Internet of Things. Companies and organizations have developed, within the framework of smart cities, smart agrifood or smart industry, initiatives for its use and generation of standards. In this context, the work aims to implement, deploy and evaluate the capabilities of Fiware to interconnect with other existing platforms and to be able to request and supply information to them.

**Lista de acrónimos**

**IoT:** Internet of Things

**MQTT:** Message Queuing Telemetry Transport protocol

**CoAP:** Constrained Application Protocol

**M2M:** Machine-to-Machine

**LwM2M:** Lightweight M2M

**HTTP:** Hypertext Transfer Protocol

**AMQP:** Advanced Message Queuing Protocol

**NGSI:** Next Generation Service Interface

**API:** Application Programming Interface

**NoSQL:** Not Only Structured Query Language

**GE:** Generic Enabler

**JSON:** JavaScript Object Notation

# Índice

## Índice de figuras

## Índice de tablas

## 1. Introducción

En los últimos años, el cloud computing y el IoT han avanzado rápidamente como las dos tecnologías fundamentales del concepto de Internet del Futuro.

La Internet of Things (IoT) es la red de objetos físicos, dispositivos y otros elementos que llevan incorporados sensores y actuadores que permiten a estos objetos recopilar e intercambiar datos, pero los componentes de esta tecnología están limitadas al nivel de capacidad de la batería, computación y almacenamiento.

Cloud computing es una tecnología que se refiere a la distribución de servicios informáticos como servidores, almacenamiento, bases de datos y aplicaciones a través de Internet en lugar de hacerlo localmente en la computadora y permite a los usuarios el acceso remoto.

En los últimos años en la literatura científica se ha estudiado y se ha propuesto un nuevo modelo de servicio que intenta unir IoT y Cloud Computing que van a formar un entorno ideal para datos masivos,almacenamiento y manipulación. Aqui hay algunas plataformas que ofrecen esta combinación entre IoT y cloud computing :

- FIWARE
- Sofia2
- ThingWorx

En este contexto,la elección de la plataforma FIWARE se basa en en varios factores, incluyendo el impacto que ha tenido en el sector académico y empresarial, así como su origen europeo.Esta iniciativa ha sido respaldada y promovida por la Comisión Europea, lo que le ha dado reconocimiento y apoyo a nivel regional.

FIWARE es una plataforma abierta y modular con el objetivo de facilitar el desarrollo de aplicaciones inteligentes en diversos campos como las ciudades inteligentes, la agricultura inteligente, la movilidad, la energía y la salud. Proporciona un conjunto de herramientas, estándares y componentes listos para usar para administrar datos en tiempo real, conectar dispositivos, analizar datos y crear servicios innovadores.

En este trabajo se profundizará en el estudio de la plataforma FIWARE, sus componentes principales y su capacidad para la gestión de datos públicos en tiempo

real. Además, se explorará el funcionamiento de la herramienta Wirecloud, sus características y su integración con FIWARE para la creación de una plataforma de visualización de datos públicos.

### 1.1. Motivaciones

La elección de FIWARE como plataforma se basa en varios factores clave. En primer lugar, FIWARE se destaca por su enfoque en estándares abiertos y su arquitectura modular y escalable. Esto permite a los desarrolladores utilizar componentes específicos y adaptar la plataforma a las necesidades de su proyecto.

Además, FIWARE ha sido ampliamente adoptado en Europa y cuenta con una comunidad activa de desarrolladores y usuarios. Esto proporciona un entorno de colaboración y apoyo, lo que facilita el intercambio de conocimientos y la reutilización de soluciones.

Otro aspecto importante es el respaldo institucional que FIWARE ha recibido de la Comisión Europea. Este respaldo brinda credibilidad y confianza en la plataforma, especialmente en el ámbito académico y empresarial.

Además, FIWARE ha demostrado su eficacia en diversos casos de uso, como ciudades inteligentes, transporte inteligente, agricultura inteligente, entre otros. Estos casos de uso exitosos respaldan la capacidad de FIWARE para implementar soluciones inteligentes de manera efectiva.

### 1.2. Objetivo

El objetivo de este trabajo es aprovechar las funcionalidades que ofrece FIWARE para la recopilación, procesamiento y análisis de datos públicos de fuentes públicas, y utilizar Wirecloud para su visualización de forma clara y accesible. La plataforma FIWARE proporcionará la capacidad de gestionar grandes volúmenes de datos de manera eficiente, mientras que WireCloud permitirá la presentación de la información en una plataforma amigable y fácil de usar.

## 2. Contexto

En este capítulo, se explorarán los conceptos básicos de FIWARE, así como su arquitectura y componentes principales. También se abordarán las técnica y herramientas de visualización de datos utilizadas.

### 2.1. Internet of Things(IOT)

El Internet of Things (IoT) se refiere a la interconexión de objetos físicos a través de Internet, permitiendo la comunicación y colaboración entre ellos sin intervención humana directa. En términos de alcance, IoT abarca una amplia gama de aplicaciones y sectores industriales[1].

En cuanto a las aplicaciones, IoT se utiliza en ciudades inteligentes para mejorar la gestión de infraestructuras urbanas, como el transporte, la energía y la seguridad. También se aplica en la salud, permitiendo el monitorio remoto de pacientes y la gestión eficiente de dispositivos médicos. En la agricultura inteligente, IoT se utiliza para optimizar el riego, el control de plagas y la monitorización de cultivos. Además, se encuentra presente en la industria manufacturera, donde se utilizan sensores y dispositivos inteligentes para mejorar la eficiencia y la calidad de los procesos de producción.

En cuanto a los sectores industriales, IoT tiene un amplio alcance. Se utiliza en la industria automotriz para habilitar características de automóviles conectados, como el seguimiento de vehículos, la navegación inteligente y la seguridad mejorada. En la logística y el transporte, IoT se emplea para rastrear y gestionar la cadena de suministro, optimizar la distribución de mercancías y mejorar la eficiencia de los medios de transporte. En la energía, IoT facilita la gestión inteligente de redes eléctricas, el monitorio de consumo y la integración de fuentes de energía renovable.


Figura 1: IOT

### 2.1.1. Arquitectura de IoT

La arquitectura de IoT típicamente se compone de cinco capas, como se describe en [2]:

**Capa de Percepción (Perception Layer)**: En esta capa, se lleva a cabo la recolección de datos del entorno u objeto a través de sensores. Los sensores capturan información sobre variables físicas o ambientales, como temperatura, humedad, movimiento, presión, entre otros.

**Capa de Transporte (Transport Layer)**: Los datos recolectados en la capa de percepción son gestionados y transmitidos en esta capa. Aquí se definen los protocolos y tecnologías de comunicación utilizados para enviar los datos hacia la siguiente capa.

**Capa de Procesamiento (Processing Layer)**: En esta capa se realiza el procesamiento, análisis y almacenamiento de la información proveniente de los sensores. Los datos son transformados y se les aplican algoritmos y técnicas para extraer conocimientos o patrones relevantes. También se puede llevar a cabo la agregación, filtrado y enriquecimiento de los datos.

**Capa de Aplicación (Application Layer)**: En esta capa, la información ya procesada y gestionada en la capa inferior es presentada y utilizada según las necesidades y requerimientos de cada entorno o aplicación. Aquí se desarrollan y despliegan los servicios y aplicaciones que brindan valor añadido a los usuarios finales.

**Capa de Negocio (Business Layer)**: Esta capa se encarga de la monetización y gestión de los servicios y aplicaciones ofrecidos en la capa de aplicación. Aquí se definen los modelos de negocio, se establecen acuerdos comerciales y se gestionan los aspectos relacionados con la rentabilidad y sostenibilidad de las soluciones IoT implementadas.

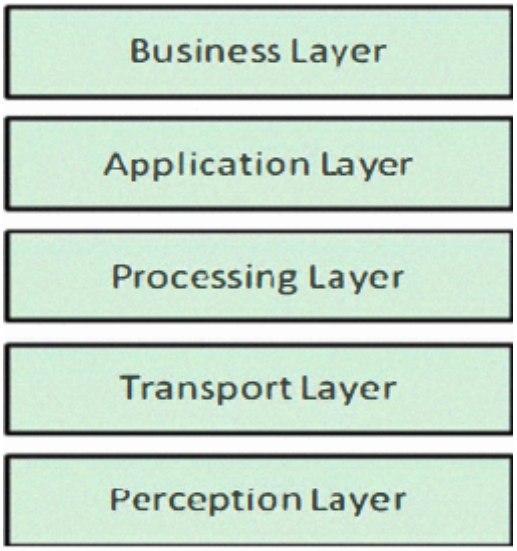



Figura 2: Arquitectura IoT

### 2.2. FIWARE

FIWARE es una plataforma estandarizada de código abierto para desarrollar e implementar soluciones tecnológicas para Internet de las cosas (IoT) y ciudades inteligentes. Fue fundada en 2011 por la Unión Europea para fomentar la innovación en el sector de las tecnologías de la información y la comunicación (TIC) y promover la interoperabilidad entre sistemas y servicios dispares. La plataforma FIWARE está compuesta por un conjunto de elementos llamados "Generic Enablers (GE)" que son unas librerías de servicios de propósito general. Estos servicios están disponibles a través de interfaces API's estándares y abiertos para que los desarrolladores puedan utilizarlos para el desarrollo de sus aplicaciones. Cada GE puede tener un conjunto de API's para facilitar la construcción de aplicaciones inteligentes.


Figura 3: FIWARE

Por lo tanto, además de la propia plataforma FIWARE, existen otros miembros de la comunidad FIWARE. Además, se compone de las siguientes organizaciones:

• **Plataforma FIWARE:** Proporciona una serie de API de código abierto sencillas pero efectivas que respaldan la creación de aplicaciones inteligentes en muchas industrias. Además, las implementaciones de referencia de cada uno de los componentes de FIWARE se pueden encontrar en la plataforma, lo que permite a los desarrolladores implementar sus soluciones de manera más rápida y sencilla[3].

• **FIWARE Lab:** Es un entorno no comercial dedicado a la experimentación y la innovación,basada en la tecnología FIWARE. Se implementa a través de una red geográficamente dispersa de nodos federados sobre una variedad de infraestructuras experimentales[4].

• **FIWARE Accelerate:** Es un programa de aceleración para promover la adopción de la tecnología FIWARE por parte de integradores y desarrolladores de soluciones y sistemas, dirigido a pequeñas y medianas empresas y nuevas empresas[5].

• **FIWARE Mundus:** Este proyecto tiene como objetivo llevar FIWARE a diferentes partes del mundo como América del Norte, América Latina, África y Asia[6].

• **FIWARE iHubs:** Su objetivo es de crear una red de desarrolladores locales. Actualmente hay 11 iHubs en todo el mundo que se centran en la fabricación y el servicio de FIWARE[7].

### 2.3. Arquitectura de FIWARE

La arquitectura de FIWARE es modular y escalable, lo que permite la integración de varias tecnologías y componentes con el objetivo de facilitar el desarrollo de soluciones técnicas avanzadas y la interoperabilidad entre sistemas dispares.

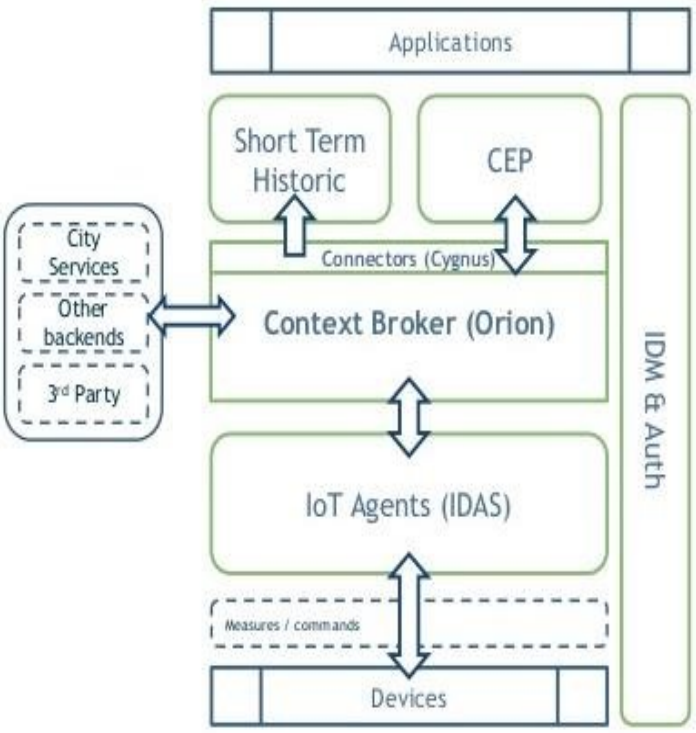



Figura 4: Arquitectura de FIWARE

Esta figura muestra la arquitectura general del FIWARE que se basa en el Orion Context Broker , por lo que circula toda la información y organiza la interconexión entre los componentes distintos de la arquitectura.

#### 2.3.1. Dispositivos físicos

Son las fuentes de datos que pueden ser sensores fijos, sensores móviles, datos públicos, etc.

#### 2.3.2. IDAS

Este GE es una puerta de entrada para que los dispositivos físicos puedan comunicarse con el Context Broker, que se basa en la arquitectura de agentes que actúan como traductores entre los protocolos que utilizan los dispositivos físicos para enviar o recibir información[8]. Al uso los agentes , los dispositivos se representan en FIWARE como entidades en Orion Context Broker, este significa que los usuarios pueden realizar consultas o suscribir para recibir los cambios de estatus de los parámetros monitorizados por los sensores y también enviar comandos de activación a los sensores para que realicen una acción.

Estos agentes IoT soportan diversos protocolos IoT específicos (LWM2M a través CoaP, JSON o UltraLight a traves HTTP/MQTT, OPC-UA, Sigfox o LoRaWAN) para recibir datos de sensores y se connecte al context broker en el cual los almacena y los gestiona.

•**Agente IoT para el protocolo LWM2M:** Este agente de Internet de las cosas es un puente que se puede utilizar para comunicar dispositivos mediante el protocolo OMA Lightweight M2M y NGSI Context Broker. El Lightweight M2M es un protocolo ligero destinado a dispositivos y comunicaciones restringidos donde el ancho de banda y la memoria del dispositivo pueden ser recursos limitados[9].

•**Agente IoT para el protocolo JSON:** Este agente de Internet de las cosas es un puente que se puede usar para comunicar dispositivos mediante un protocolo JSON (con transportes AMQP , HTTP y MQTT) simple y NGSI Context Broker. Este protocolo se basa en objetos JSON simples de un solo nivel que codifican pares (atributo, valor). Este protocolo tiene como objetivo la simplicidad y la facilidad de uso[10].

•**Agente IoT para el protocolo UltraLight:** Este IoT agente es un puente que se puede usar para comunicar dispositivos que usan el protocolo Ultralight 2.0 y NGSI Context Broker. Ultralight 2.0 es un protocolo ligero basado en texto destinado a dispositivos y comunicaciones restringidos donde el ancho de banda y la memoria del dispositivo pueden ser recursos limitados. Este IoT Agent proporcionará diferentes enlaces de protocolo de transporte para el mismo protocolo: HTTP, MQTT[11].

•**Agente IoT para el protocolo OPC-UA:** El agente F4I IDAS OPC UA es un componente de código abierto destinado a permitir la captura de datos de dispositivos OPC UA en el taller y proporcionarlos a los niveles superiores de un sistema basado en FIWARE. Por lo tanto, el enfoque principal de este componente está en la comunicación de los dispositivos de campo que implementan un servidor OPC UA a FIWARE, lo que permite la comunicación con FIWARE Orion Context Broker[12].

•**Agente IoT para el protocolo Sigfox:** Este IoT Agent está diseñado para ser un puente entre el protocolo de devoluciones de llamada de Sigfox y el protocolo OMA NGSI utilizado por Context Broker , así como por otros componentes del ecosistema FIWARE[13].

•**Agente IoT para el protocolo LoRaWAN:** Este agente de Internet de las cosas es un puente que se puede usar para comunicar dispositivos que usan el protocolo LoRaWAN permite el intercambio de datos y

comandos entre dispositivos IoT y FIWARE NGSI Context Broker utilizando el protocolo LoRaWAN[14].

#### 2.3.3. Orion Context broker

Orion Context Broker es una parte importante de FIWARE y está diseñado para administrar y procesar datos en tiempo real de múltiples fuentes en el contexto del Internet de las cosas(IoT).

La función principal de Orion Context Broker es recopilar, almacenar y entregar información contextual en tiempo real sobre entidades que forman parte del ecosistema IoT.

La arquitectura de publicación/suscripción de Orion Context Broker permite que las aplicaciones se suscriban a las actualizaciones contextuales de una entidad específica. Esto significa que siempre que se produzca un cambio en las características de una entidad supervisada, las aplicaciones pueden recibir notificaciones en tiempo real.

Además, Orion Context Broker ofrece una API que, a través de solicitudes RESTful, permite a los desarrolladores acceder, consultar y actualizar los datos contextuales. Y el servidor NGSI v2 para gestionar la información de contexto y su disponibilidad[15].

### NGSIv2

NGSIv2 (Next Generation Service Interface Version 2) es una especificación API estandarizada desarrollada por la comunidad FIWARE para administrar información contextual en sistemas distribuidos. Esta es una evolución de la especificación NGSIv1 y proporciona un conjunto de API más completo en funciones para la gestión de contexto[16].

Las características y conceptos de NGSIv2 son:

- **Entidades y atributos:** NGSIv2 introduce el concepto de entidades para representar objetos o entidades dentro del sistema de gestión de contexto. Una entidad puede tener múltiples atributos que describen sus propiedades y características. Los atributos pueden ser de diferentes tipos como cadenas, números, booleanos o geolocalizaciones.


Figura 5: Entidad y sus atributos

•**Operaciones CRUD:** NGSIv2 admite las operaciones CRUD (Create, Read, Update, Delete) estándar para gestionar entidades y sus atributos. Proporciona API para crear nuevas entidades, recuperar información de entidades, actualizar valores de atributos y eliminar entidades o atributos.

Para las operaciones en las que el <entity-id>aún no esta conocido o no esta especificado , utilizamos el punto final /v2/entities si no utilizamos /v2/entities/<entity-id>[17].

| HTTP | /v2/entities | /v2/entities/<entity-id> |
|---|---|---|
| POST | Crear una nueva entidad y agregarla al contexto | Crear o Actualizar un atributo de una entidad específica |
| GET | Leer datos de entidad del contexto. Esto devolverá datos de varias entidades. Los datos se pueden filtrar | Leer datos de entidad de una entidad especificada. Esto devolverá datos de una sola entidad solamente.Los datos se pueden filtrar |
| PUT | X | X |
| PATCH | X | X |
| DELETE | X | Eliminar una entidad del contexto |

Tabla 1: Operaciones CRUD de entidades

Para realizar operaciones CRUD en atributos, debe que el <entity-id>esta conocido[17].
Los tres puntos finales de cada atributo son los siguientes :

- /v2/entities/<entity-id>/attrs: Sólo se utiliza para una operación de actualización de uno o varios atributos existentes.

- /v2/entities/<entity-id>/attrs/<attribute>: Se utiliza para manipular un atributo en su conjunto.

• /v2/entities/<entity-id>/attrs/<attribute>/value: Se utiliza para leer o actualizar el valor de un atributo, dejando el tipo intacto.

| HTTP | /v2/entities/<entity-id>/attrs | /v2/entities/<entity-id>/attrs/<attribute> | /v2/entities/<entity-id>/attrs/<attribute>/value |
|---|---|---|---|
| POST | X | X | X |
| GET | X | X | Leer el valor de un atributo de una entidad especificada. Esto devolverá un único campo. |
| PUT | X | X | Actualizar el valor de un único atributo de una entidad especificada. |
| PATCH | Actualizar uno o más atributos existentes de una entidad existente | X | X |
| DELETE | X | Eliminar un atributo existente de una entidad existente | X |

Tabla 2: Operaciones CRUD de atributos

•**Suscripciones y notificaciones:** Posibilidad de suscribirse a la información contextual. Esto permite que el aplicación reciba notificaciones asincrónicas cuando sucede 'algo'. De esa forma, el aplicación no tiene que seguir enviando solicitudes de consulta. Orion Context Broker envía información generada por notificaciones.

• **Consulta y filtrado:** NGSIv2 proporciona capacidades de consulta robustas para recuperar información particular sobre entidades basadas en valores de atributos u otros criterios. Se admiten también el filtrado para que podemos consultar las entidades y los atributos lo que deseamos verla en la consulta.

FIWARE utiliza MongoDB como una de las opciones de base de datos para el almacenamiento de datos.

**MongoDB:** MongoDB es un sistema de gestión de bases de datos orientado a documentos, también conocido como NoSQL. Se diferencia de las bases de datos relacionales tradicionales por utilizar un modelo de datos flexible basado en documentos JSON. MongoDB está diseñado para ser escalable, de alto rendimiento y fácil de usar[18].

#### 2.3.4. IDM y Auth

Los componentes de identidad y autorización brindan los mecanismos para garantizar la confiabilidad , seguridad y privacidad en la entrega y uso de servicios, también cubran un seria de aspecto con el acceso de los usuarios a las redes,servicios y aplicaciones incluyendo :

⇒ La autenticación segura y privada de usuarios , dispositivos, redes y servicios.

⇒ Gestión de perfil de usuarios,dispositivos de preservar la privacidad de datos personales.

⇒ Control de acceso

⇒ Administración de credenciales

#### 2.3.5. Short Term Historic (STH)

Este componente se encarga de proporcionar información agregada de series de tiempo, sobre la evolución en el tiempo de los valores de los atributos de un entidad, los cuales se registraron utilizando al Orion ontext broker.

Este componente es muy útil debido a que en ocasiones, las aplicaciones consultan información explícita que podría manejada de un manera más rápida y simple que la que ofrece una herramienta analítica de big data.

#### 2.3.6. Complex Event Processing

Este componente analiza datos de eventos de tiempo real y reaccionar a los eventos, generando una respuesta inmediata o desencadenando acciones de acuerdo a las condiciones cambiantes.

#### 2.3.7. Aplicaciones

La parte de aplicación de FIWARE se basa en la utilización de componentes de software reutilizables, denominados «enables», que proporcionan funcionalidades específicas y se pueden combinar para construir soluciones personalizadas. Estos enables incluyen tecnologías como el procesamiento de datos en tiempo real, el almacenamiento y consulta de datos, la gestión de identidad y acceso, la visualización de datos con por ejemplo CKAN o Wirecloud.

**CKAN** es una plataforma de código abierto diseñada para gestionar y publicar conjuntos de datos de forma abierta y accesible. CKAN, que significa Comprehensive Knowledge Archive Network, permite a las organizaciones y gobiernos compartir y difundir datos abiertos de manera efectiva[19].

Algunas características y funcionalidades clave de CKAN incluyen:

• **Gestión de metadatos**: CKAN proporciona una interfaz para ingresar y administrar metadatos detallados sobre los conjuntos de datos, lo que incluye información como título, descripción, etiquetas, formato, licencia, entre otros. Esto facilita la búsqueda y el descubrimiento de datos.

• **Portal de datos**: CKAN ofrece un portal web donde los usuarios pueden explorar y buscar los conjuntos de datos disponibles. Los datos se presentan de manera estructurada, lo que facilita la navegación y la comprensión de los diferentes conjuntos de datos.

• **Descargas y acceso a datos**: CKAN permite a los usuarios acceder y descargar los conjuntos de datos en diferentes formatos, como CSV, JSON, XML, entre otros. También es posible establecer restricciones de acceso y licencias específicas para los datos.

• **Integración con otras herramientas y servicios**: CKAN se integra con otras herramientas y servicios populares, como visualizaciones interactivas, aplicaciones de análisis y sistemas de gestión de autenticación, lo que permite una mayor funcionalidad y personalización.

• **API y extensiones**: CKAN proporciona una API que permite a los desarrolladores acceder y utilizar los datos y metadatos almacenados en la plataforma. Además, CKAN es altamente extensible, lo que significa que se pueden agregar funcionalidades adicionales mediante extensiones personalizadas.

CKAN es utilizado por diversas organizaciones, incluidos gobiernos, instituciones académicas, organizaciones sin fines de lucro y empresas, para gestionar y compartir datos abiertos de manera efectiva. Su naturaleza de código abierto y su comunidad activa permiten una continua mejora y desarrollo de la plataforma.

## 2.4. El uso de FIWARE en el marco industrial

FIWARE es una enactiva de código abierto que tiene como objetivo promover la creación de los estándares necesarios para desarrollar aplicaciones Smart en diferentes dominios como Smart Cities , Smart Agri Food, Smart Industry .

### 2.4.1. Smart Cities

La transformación de las ciudades pretende adoptar una gestión más eficiente de los servicios, para promover la innovación, el crecimiento económico y el bienestar común. En este caso, la utilización de FIWARE facilita este trabajo a las ciudades, y ayudarlas a conseguir grandes resultados.

### 2.4.2. Smart AgriFood

Es una nueva forma de agricultura que se refiere a los alimentos inteligentes y tiene el objetivo de optimizar la producción en las granjas a través de usar recursos modernos de manera sostenible , aumentar la producción y entregar los

mejores productos en términos de calidad al tiempo que ofrece la maximización de rendimiento . a través de utilizar una amplia gama de tecnologías como sensores IoT, wearables, servicios GPS, UAVs, robots y drones que están en el campo y que proporcionan datos en tiempo real a los sistemas ayudando a monitorizar la línea de producción y apoyando las decisiones.Esto permite reducir los residuos y maximizar la eficiencia de las operaciones. Con FIWARE podemos obtener estos objetivos .

### 2.4.3. Smart Industry

La transparencia de la información es el más importante en Industrie. Para gestionar y mantener la línea de producción de un taller en una fábrica inteligente puede utilizar las máquinas , los sensores inteligentes que generan los datos, pero la mayoría de las veces estos datos permanecen en silos de información. El uso de información contextual, tanto de fabricación como de no fabricación, puede mejorar la eficiencia y acelerar considerablemente los procesos de producción, y esto es lo que ofrece FIWARE a las fábricas .

## 2.5. Tecnología Docker

Docker es una tecnología de virtualización de contenedores que permite a los desarrolladores crear, implementar y ejecutar aplicaciones en entornos virtualizados aislados.

Docker proporciona una forma eficiente y estandarizada de empaquetar y distribuir aplicaciones junto con sus dependencias, lo que facilita la implementación de aplicaciones en diferentes entornos y sistemas operativos.

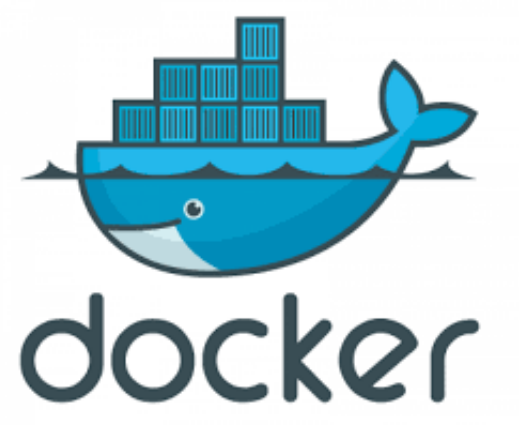



Figura 6: Docker

Un contenedor Docker es una unidad de software portátil que contiene todo lo que su aplicación necesita para ejecutarse, incluido el código, las bibliotecas y otras dependencias. Cada contenedor se ejecuta en un entorno aislado. Es decir, no afecta a otras aplicaciones que se ejecutan en el mismo sistema.[20]

Algunas características y conceptos clave relacionados con los contenedores de Docker son :

**Imagen de Docker:** Una imagen de Docker es un paquete ejecutable autónomo que contiene todo lo que su aplicación necesita para ejecutarse, incluido el código, las bibliotecas, las dependencias y la configuración del entorno. Una imagen se construye a partir de un archivo llamado Dockerfile que contiene instrucciones para construir la imagen.

**Contenedor:** Una contenedor es una instancia en ejecución de una imagen de Docker. Cada contenedor se ejecuta de forma independiente con su propio sistema de archivos y recursos asignados, pero comparte el mismo kernel que el sistema operativo host. Los contenedores proporcionan un entorno coherente y reproducible para ejecutar aplicaciones.

**Docker Hub:** Docker Hub es un registro público de Docker donde puede buscar y compartir imágenes de Docker. Brinda acceso a un gran conjunto de imágenes ya creadas por la comunidad de Docker, lo que facilita la implementación y el uso de aplicaciones listas para usar.

**Dockerfile:** Un Dockerfile es un archivo de texto que contiene las instrucciones necesarias para crear una imagen de Docker. Se utiliza para especificar el entorno base, instalar dependencias, copiar archivos y configurar aplicaciones dentro de la imagen.

**Docker Compose:** Docker Compose es una herramienta que le permite definir y administrar aplicaciones de varios contenedores. Docker Compose facilita la creación y ejecución de aplicaciones de varios contenedores mediante el uso de archivos YAML para describir la configuración de su servicio, red y volumen.

**Orquestación Docker Swarm y Kubernetes:** es una popular herramienta de orquestación de contenedores que puede administrar y escalar aplicaciones basadas en contenedores en múltiples nodos o máquinas. Ofrece características avanzadas como balanceo de carga, detección de servicios y escalado automático.

**Portabilidad:** Los contenedores Docker son portátiles. Esto significa que puede ejecutar el mismo contenedor en diferentes entornos, como su máquina original, su servidor de desarrollo o la nube. Esto facilita la migración de sus aplicaciones y asegura que sus aplicaciones funcionen de manera consistente en diferentes sistemas.
**Eficiencia:** Los contenedores Docker son económicos en términos de recursos y tiempo de inicio. Una arquitectura común permite que varios contenedores se ejecuten en un solo host sin sobrecargarse, lo que da como resultado un uso más eficiente de los recursos y una escalabilidad mejorada.
Además, Docker proporciona herramientas de administración y monitoreo de contenedores que facilitan la administración de aplicaciones en producción. Estos incluyen herramientas para orquestación de contenedores, escalado automático, administración de redes y monitoreo de contenedores.

### 2.6. Bases de datos públicas

Los datos públicos son información y conjuntos de datos que son accesibles y están disponibles para el público en general sin restricciones significativas de acceso, uso o redistribución. Estos datos se consideran de dominio público y están disponibles para cualquier persona que desee utilizarlos, ya sea con fines personales, educativos, comerciales o de investigación. Los datos públicos están disponibles a partir de una variedad de fuentes, incluidos gobiernos, organizaciones no gubernamentales, instituciones académicas, empresas y otras entidades.

Estos datos incluyen varios temas, como demografía, datos geoespaciales, datos económicos, datos de salud, información ambiental, datos científicos y datos financieros.


Figura 7: Datos públicos

A continuación se citan algunos ejemplos de bases de datos públicas populares:

**Datos abiertos de gobierno:** Muchos gobiernos ofrecen portales de datos abiertos que contienen conjuntos de datos en áreas como demografía, educación, salud y transporte, como el Portal de datos abiertos de Estados Unidos (data.gov), el

Portal de datos abiertos de la Unión Europea (data.europa.eu) y el Portal de datos abiertos del gobierno del Reino Unido (data.gov.uk).

**Datos científicos:** Las instituciones académicas y de investigación a menudo comparten datos científicos que están disponibles públicamente. Los ejemplos incluyen NASA (NASA Open Data), CERN (Open Data Portal), National Institutes of Health (NIH Data Sharing) y Earth Data and Environmental Science Institute (Earth Data).

**Datos financieros:** Hay bases de datos públicas que proporcionan datos financieros como precios de acciones, datos económicos y estadísticas financieras, como Yahoo Finance, Google Finance y la Reserva Federal de Estados Unidos (FRED).

**Datos de investigación:** Existen repositorios y bases de datos públicas que almacenan conjuntos de datos utilizados en diversos campos de investigación, como el repositorio de aprendizaje automático de UCI, los conjuntos de datos de Kaggle y el archivo de datos generales de ciencias sociales.

**Datos de salud:** Las organizaciones de salud y las agencias gubernamentales proporcionan bases de datos públicas sobre salud y medicina. Por ejemplo, los Centros para el Control y la Prevención de Enfermedades (CDC) de Estados Unidos proporcionan varios conjuntos de datos de salud y enfermedades.

La disponibilidad de datos públicos es fundamental para promover la transparencia, la participación pública y la cooperación en la sociedad. La publicación de estos datos impulsa la innovación, impulsa la toma de decisiones informada y acelera el desarrollo de aplicaciones y servicios basados en datos.

## 2.7. Wirecloud

Wirecloud es una plataforma para desarrollar y ejecutar aplicaciones web que simplifican la creación de interfaces de usuario y la gestión de datos.


Figura 8: Wirecloud

Wirecloud permite a los desarrolladores diseñar paneles interactivos e interfaces de usuario utilizando componentes reutilizables llamados "widgets". Puede arrastrar y soltar y conectar estos widgets para crear una interfaz personalizada para las necesidades específicas de cada aplicación[21].

Además del diseño de la interfaz de usuario, Wirecloud también permite la integración de fuentes de datos de varias plataformas y servicios, como sensores, bases de datos y servicios web de IoT. Permite a los desarrolladores combinar y utilizar datos de diferentes fuentes en sus aplicaciones.

Wirecloud también brinda la capacidad de administrar y compartir sus aplicaciones. Los desarrolladores pueden implementar sus aplicaciones en la plataforma y ponerlas a disposición de otros, promoviendo la reutilización y la colaboración.

Wirecloud consta de varios componentes clave que le permiten desarrollar y ejecutar aplicaciones web. Los componentes principales de Wirecloud son:

**Espacio de trabajo(workspace):** Este es el entorno principal de Wirecloud donde los usuarios pueden crear, organizar y administrar sus aplicaciones. Proporciona una interfaz fácil de usar para diseñar interfaces de usuario.

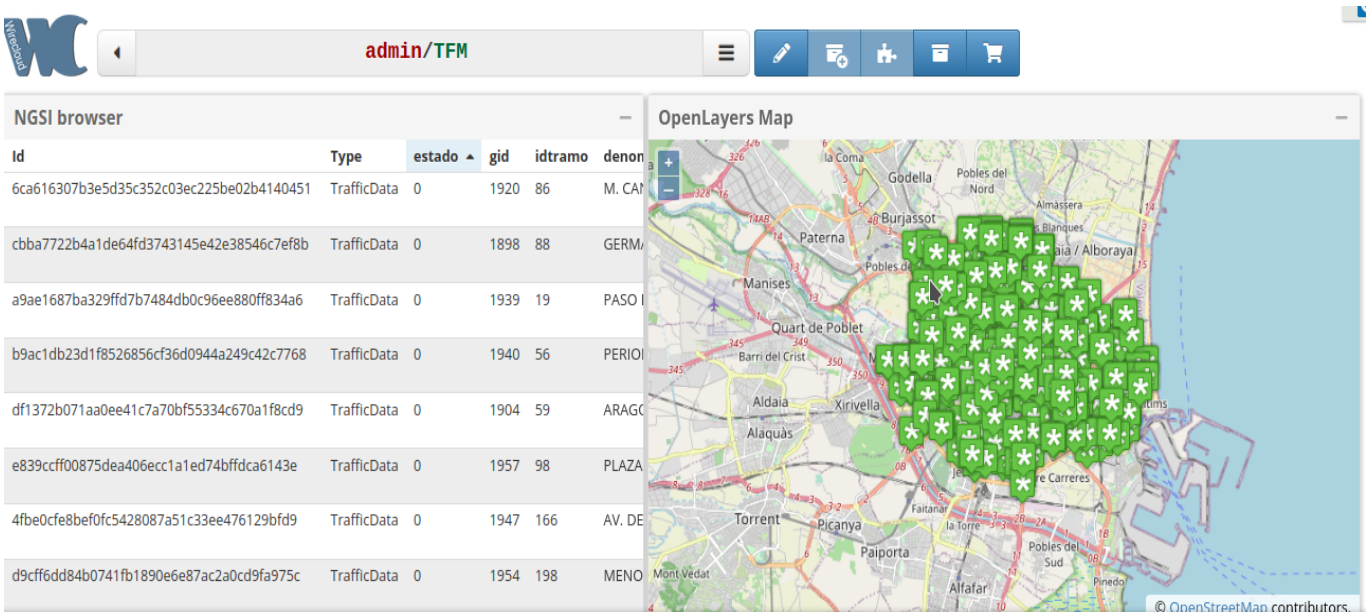



Figura 9: Interfaz del entorno de trabajo de Wirecloud

**Widgets:** Los widgets son componentes reutilizables en el espacio de trabajo para crear interfaces de usuario interactivas. Wirecloud proporciona una gran cantidad de widgets predefinidos, como gráficos, botones, listas desplegables, mapas, etc. Los usuarios también pueden crear sus propios widgets personalizados[22].

Los principales widgets utilizados en este proyecto:

> **OpenLayers Map:** Es un visor de mapas que representa la entidad especificada en Wirecloud.
>
> **NGSI Browser:** Es un componente de Wirecloud que ofrece a los usuarios de explorar y visualizar las entidades de fiware en Wirecloud.
>
> **HighCharts:** Es un componente que muestra los gráficos.

**Operadores:** Hay muchos tipos diferentes de operadores, pero en general su función principal es procesar todo tipo de datos y garantizar que la información se introduce al operador, se transforme y se envíe.

> **NGSI Source:** Es un operador que recibe datos del dominio y lo especifica en la configuración, en nuestro caso es el dominio es FIWARE.
>
> **Value List Filter:** Es un operador que permite de filtrar los datos en función de los valores seleccionados en una lista predefinida.
>
> **NGSI Entity To POI:** Es un operador que permite de convertir las entidades NGSI que vienen desde FIWARE en puntos geográficos en la mapa de wirecloud.

**Operador Lables to dataserie:** Es un operador que permite de convertir un conjunto de etiquetas en un conjunto de números y cuente la cantidad de veces que se encuentra cada etiqueta para poder utilizarlas en un HighCarts.

**Column chart generator:** Es un operador que permite de Generar el modelo de datos del gráfico de columnas para el widget highcharts.

**Editor de cableado(Wiring):** Uno de los apartados más importantes es Wiring. Esta herramienta permite conectar widgets e operadores. Los usuarios pueden definir conexiones entre widgets para establecer interacciones e intercambiar datos entre widgets. El Wiring generalmente se realiza conectando cables virtuales entre las entradas y salidas de los widgets e operadores. Ellos tienen entradas que reciben y procesan información y le devuelven procesada a través de salidas, pero no todos los widgets y operadores tienen entradas y salidas por ejemplo el NGSI Source no tiene entrada ya en este caso a través de la configuración de datos que quieren devolverlos por la salida.

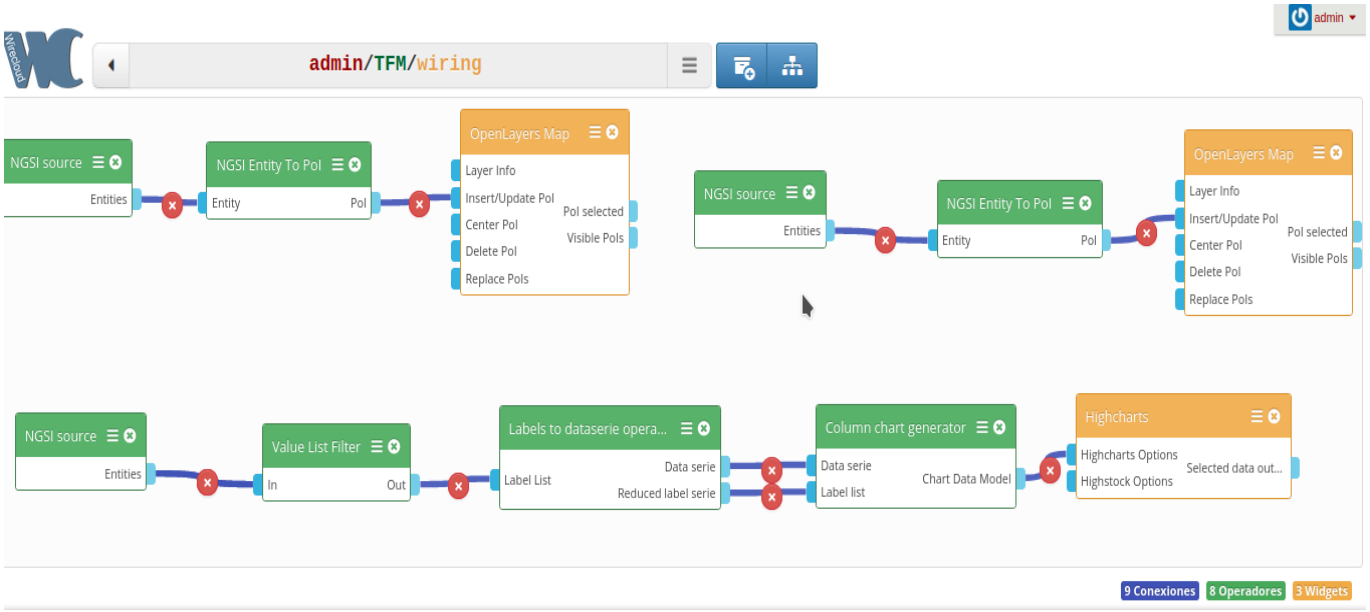



Figura 10: Wiring

# 3. Descripción de la solución

## 3.1. Arquitectura del proyecto

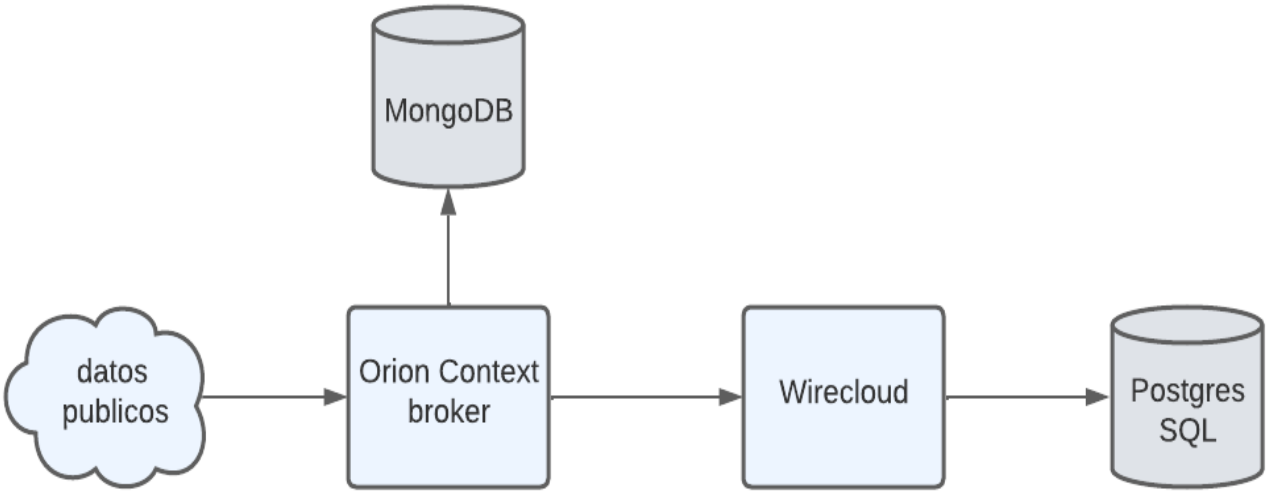



Figura 11: Arquitectura del proyecto

El proyecto se compone de tres partes principales:

**1. Importación de datos publicos:** Para esta tarea, se utilizará un programa desarrollado en el lenguaje de programación Python. Este programa se encargará de acceder a las APIs públicas proporcionadas por el ayuntamiento de Valencia y extraer los datos necesarios. Python ofrece diversas bibliotecas y herramientas para realizar solicitudes a APIs y procesar los datos obtenidos.

**2. Definiciones de las entidades:** Una vez que los datos se hayan obtenido de las APIs, serán definidas en entidades, y podemos realizar de operaciones CRUD (crear, leer, actualizar y eliminar) sobre estos entidades. Se desarrollarán programas adicionales en Python para interactuar con Orion y ejecutar las operaciones necesarias en los datos almacenados.

**3. Visualización de datos:** Después que todos los datos están definidos en entidades, utilizamos Wirecloud para visualizar estos datos en diferentes formas a través de los operadores y las widgets.

### 3.2. Despliegue del proyecto con Docker

Todo el trabajo esta desplegado con Docker, cada componente que necesitar-lo en trabajo esta implementado en un contenedor Docker. Además, utilizamos docker-compose para instalar el entorno de trabajo, hemos crear un fichero docker-compose.yml con todo la configuración de cada componente. entonces empezare-mos con la instalación de FIWARE y después Wirecloud.
En este tabla resumen los componentes y puertos que estan utilizados en el pro-yecto.

| Servicio | Puerto | Usado para |
|---|---|---|
| Orion | 1026 | Administrar y acceder a información de contexto |
| MongoDB | 27017 | La base de datos de FIWARE |
| Wirecloud | 8000 | La plataforma de Wirecloud |
| PosgresSQL | 5432 | La base de datos de wirecloud |
| Ngsiproxy | 3000 | hacer el intercambio entre FIWARE y Wirecloud |
| Nginx | 80 | Servidor web y proxy inverso para el servicio Wirecloud |
| Elasticsearch | 9200/9300 | Almacenamiento e indexación de datos en Wirecloud |
| Memcached | 11211 | almacenar datos en caché y mejorar el rendimiento |

Tabla 3: Los servicios del proyecto

### 3.3. Instalación de FIWARE

FIWARE consta de dos componentes muy importantes que son Orion Context Broker y la base de datos MongoDB.

Crearemos un archivo de configuración llamado docker-compose.yml , en el que especificaremos los componentes de FIWARE y sus configuración.

```
orion:
    image: fiware/orion:2.4.0
    ports:
        - 1026:1026
    depends_on:
        - mongo
    command:
        -dbhost mongo
        -db orion

mongo:
    image: mongo:3.6.16
    command: --nojournal --smallfiles
    volumes:
        - ./data/mongo-data:/data/db
```

Figura 12: Configuración de Orion y MongoDB en docker-compose

Y después de crear y ejecutar el ficher docker-compose.yml con el comando **"docker-compose up"** verificar con el comando **"docker ps"** para ver que los contenedores están bien hecho.


Figura 13: los contenedores de Orion y MongoDB

Finalmente accedemos a URL **"http://localhost:1026/version"** para verificar que FIWARE esta instalado y funciona bien.

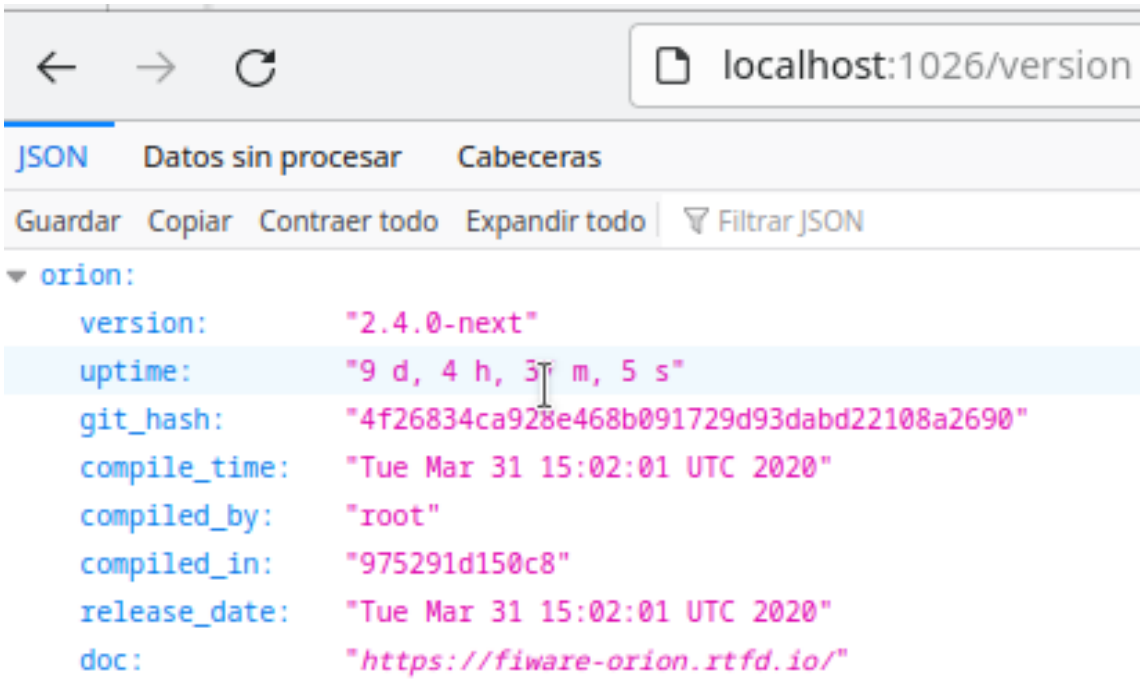



Figura 14: Verificación de funcionamiento de FIWARE

### 3.4. Instalación de Wirecloud

Con la misma manera de instalación que FIWARE vamos a instalar Wirecloud. Anadimos los componentes requeridos por Wirecloud al fichero docker-compose.yml. Estos componentes incluyen el servidor web(nginx), la base de datos(PostgresSQL) y los servicios auxiliares (Memcached,elasticsearch).

```
version: "3"

services:

    nginx:
        image: nginx:latest
        ports:
            - 80:80
        volumes:
            - ./config/nginx.conf:/etc/nginx/nginx.conf:ro
            - ./data/wirecloud/wirecloud-static:/var/www/static:ro

        depends_on:
            - wirecloud

    wirecloud:
        image: fiware/wirecloud:1.3
        depends_on:
            - postgres
            - elasticsearch
            - memcached
        environment:
            - DEBUG=false
            - LOGLEVEL=INFO
            - DB_HOST=postgres
            - DB_PASSWORD=wirepass
            - FORWARDED_ALLOW_IPS=*
            - ELASTICSEARCH2_URL=http://elasticsearch:9200/
            - MEMCACHED_LOCATION=memcached:11211
        volumes:
            - ./data/wirecloud/wirecloud-data:/opt/wirecloud_instance/data
            - ./data/wirecloud/wirecloud-static:/var/www/static

    elasticsearch:
        image: elasticsearch:2.4
        volumes:
            - ./data/wirecloud/elasticsearch-data:/usr/share/elasticsearch/data
        command: elasticsearch -Des.index.max_result_window=50000

    memcached:
        image: memcached:1
        command: memcached -m 2048m

    postgres:
        image: postgres:9.6
        environment:
            - POSTGRES_PASSWORD=wirepass
        volumes:
            - ./data/wirecloud/postgres-data:/var/lib/postgresql/data

    ngsiproxy:
        image: fiware/ngsiproxy:1.2.0
        ports:
            - 3000:3000
```

Figura 15: Configuracion de Wirecloud en docker-compose

después, para verificar que los contenedores de los servicios están bien hecho utilizamos el comando **"docker ps"**

```
68c85244b396   nginx:latest              "/docker-entrypoint.…"   2 weeks ago   Up 5 minutes            0.0.0.0:80->80/tcp, :::80->80/tcp
ed8af6555669   fiware/wirecloud:1.3      "/docker-entrypoint.…"   2 weeks ago   Up 5 minutes (healthy)  8000/tcp
4cb8554dbeee   fiware/ngsiproxy:1.2.0    "docker/entrypoint.sh"   2 weeks ago   Up 5 minutes            0.0.0.0:3000->3000/tcp, :::3000->3000/tcp
b367fa1981f6   elasticsearch:2.4         "/docker-entrypoint.…"   2 weeks ago   Up 5 minutes            9200/tcp, 9300/tcp
43887753b5a1   postgres:9.6              "docker-entrypoint.s…"   2 weeks ago   Up 5 minutes            5432/tcp
4b2a0debb57d   memcached:1               "docker-entrypoint.s…"   2 weeks ago   Up 5 minutes            11211/tcp
```

Figura 16: Los contenedores de Wirecloud

Finalmente accedemos a URL **"http://localhost"** para verificar que Wirecloud están instalado y funciona bien.

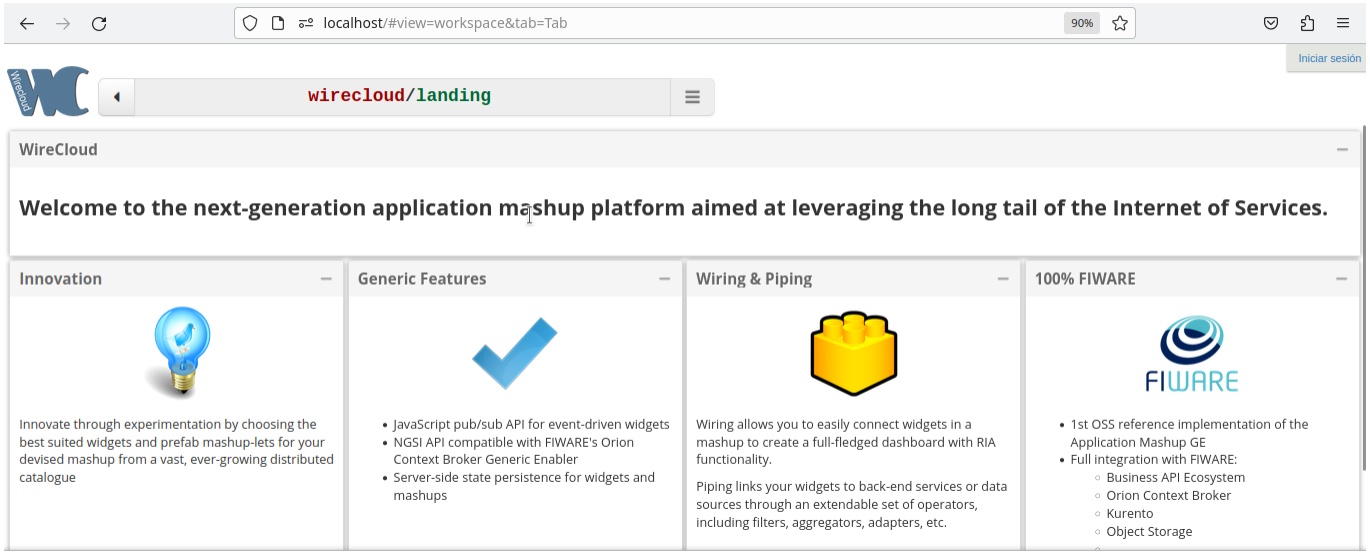



Figura 17: Interfaz de Wirecloud

### 3.5. Configuración de Wirecloud

Una vez que se haya confirmado que Wirecloud se ha iniciado correctamente, es necesario crear un superusuario para poder gestionar la plataforma. Para ello, se puede ejecutar el siguiente comando:

```
root@lpfc2004:/home/usuario/lets-fiware.tutorials/wirecloud# docker-compose exec wirecloud python manage.py createsuperuser
```

Figura 18: creación de superusuario de Wirecloud

Al ejecutar este comando, se solicitará ingresar la siguiente información:

Nombre de usuario: que es **admin**

Dirección de correo electrónico: que es **eyamadiouni1@gmail.com**.

Contraseña del superusuario: que es **admin**.

Una vez que haya proporcionado toda la información requerida, accedemos a la plataforma con el nombre y la contraseña.

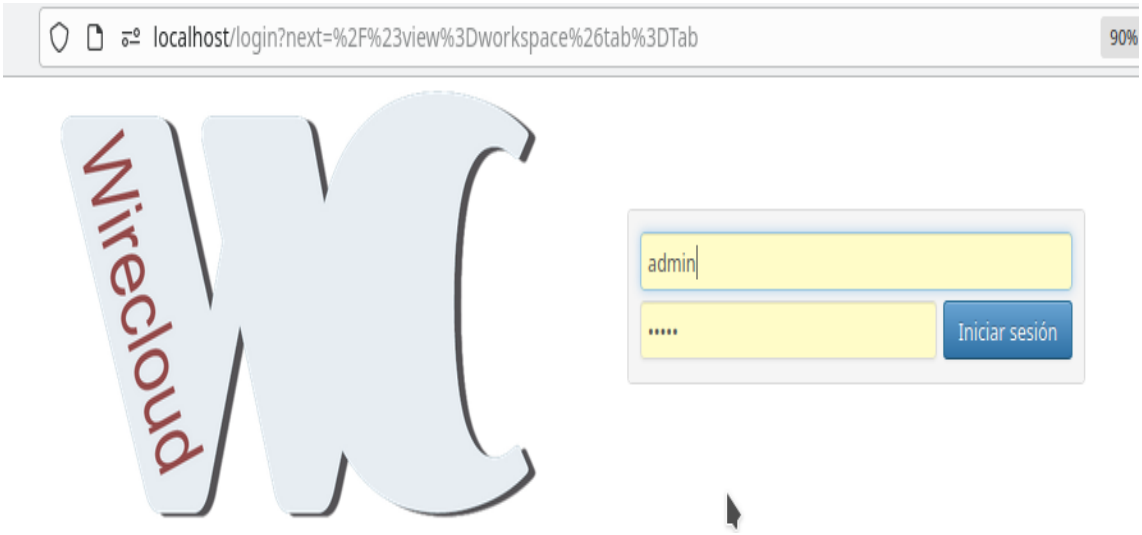



Figura 19: login de Wirecloud

Antes de comenzar a trabajar en Wirecloud, hay dos aspectos importantes que se deben abordar: la creación del entorno de trabajo y la instalación de los widgets y operadores necesarios.

### 3.5.1. Creación del entorno de trabajo

Para crear un entorno de trabajo en Wirecloud, desplegar el botón de menú y crear un nuevo entorno.

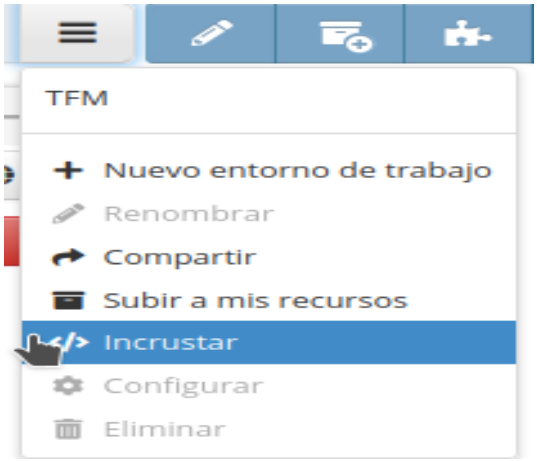



Figura 20: Creación del entorno de trabajo

En este proyecto el nombre de entorno de trabajo va ser **TFM**.


Figura 21: del nombre de entorno

### 3.5.2. Instalación de Widgets e Operadores

Para subir los archivos de los widgets y operadores, desde nuestra plataforma de WireCloud tenemos que ir a mis recursos:

Figura 22: entrar a mis recursos

Una vez que estamos ahí pulsamos en subir:

Figura 23: Obtener los widgets e operadores

Y nos aparecerá por último subir el archivo '.wgt', hacer clic en esto y subir los ficheros necesitamo.

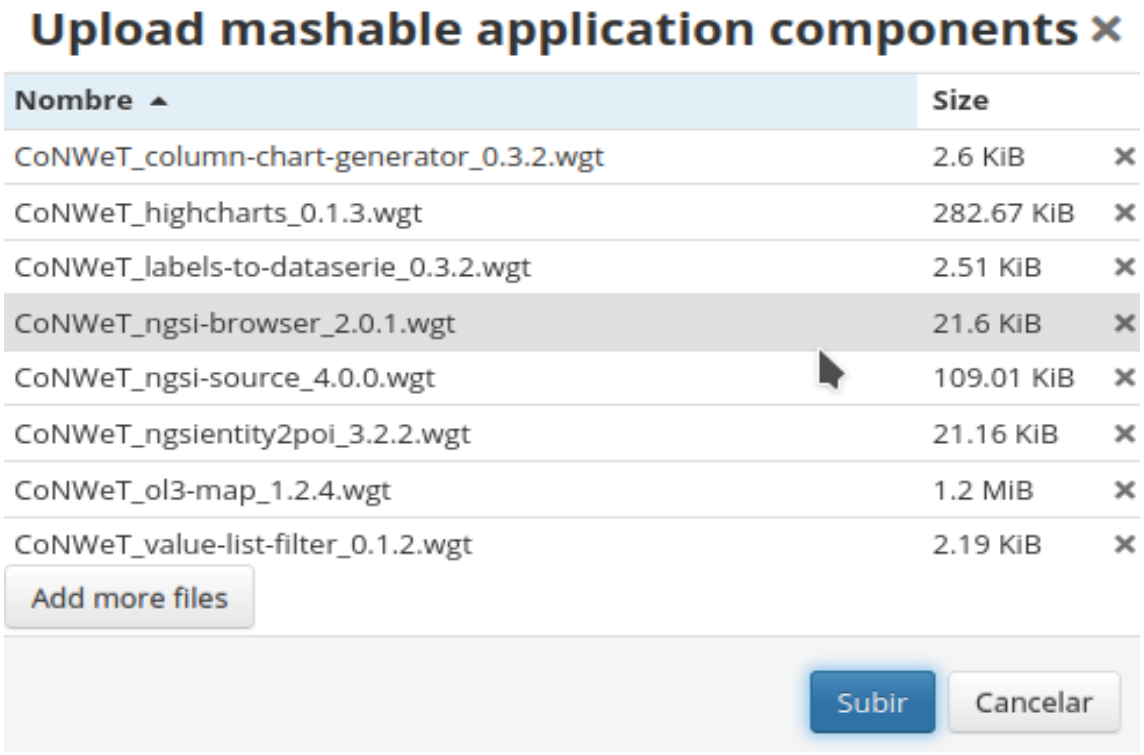



Figura 24: Subir los widgets e operadores

Y con esto todos los widgets y operadores que necesitamos están instalados.

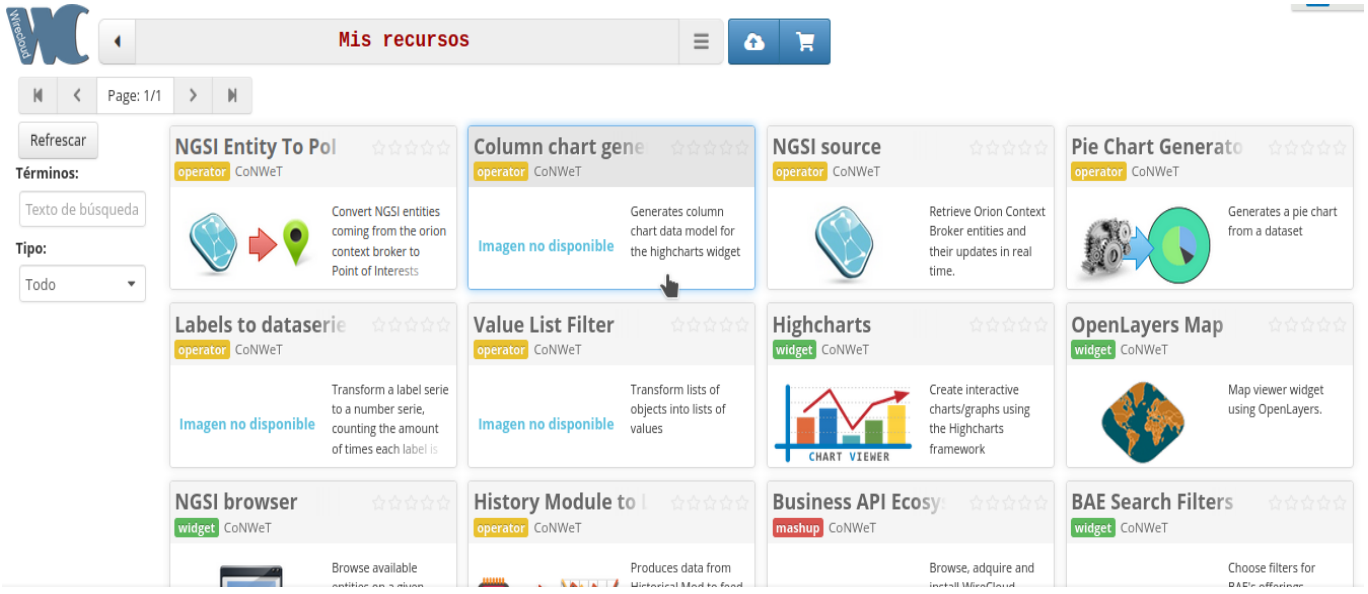



Figura 25: Los widgets e los operadores

# 4. Resultados

En el capítulo de resultados del proyecto, se presentarán los resultados obtenidos en cada una de las partes del proyecto. Comenzaremos con la primera parte, que se refiere a la importación de datos.

## 4.1. Importación de datos

En este proyecto, se utilizarán dos tipos de datos publicados por el Ayuntamiento de Valencia: datos de tráfico en tiempo real y datos de disponibilidad de bicicletas en las estaciones de bicicletas en Valencia.

Vamos utilizar un programa con la lenguaje Python para poder importar los APIs de los datos que están publicados en el portal de la ayuntamiento de Valencia.

Este es el esqueleto del código que vamos a utilizarlo por la importación de los dos tipos de datos (datos del trafico y datos de la disponibilidad de bicicletas) solamente por cada código añadimos la dirección del API, los atributos necesarios y la dirección del servicio por cada API.

```python
# -*- coding: utf-8 -*-
import requests
import json
# Obtener datos de la API abierta
api_url = "la direccion del API"
while True:
  try:
      response = requests.get(api_url)
      data = response.json()

      for record in data["records"]:
          fields = record["fields"]
          coordinates = fields["geo_shape"]["coordinates"]
          # Transformar los datos en el formato de FIWARE
          fiware_data =  {
              "id": record["recordid"],
              "type": "TrafficData o BiciData",
              #anadir todos los attributos necesarios con sus tipos y valores
          }
          # Insertar los datos en FIWARE
          fiware_url = "http://localhost:1026/v2/entities"
          headers = {
              "Content-Type": "application/json",
              "Fiware-Service": "SmartCity",
              "Fiware-ServicePath": "/trafico"
                  }
          response = requests.post(fiware_url, headers=headers, data=json.dumps(fiware_data))

          # Comprobar el resultado de la inserción en FIWARE

          if response.status_code == 204:
              print("Datos insertados correctamente en FIWARE.")
          else:
              print("Error al inserción  datos en FIWARE:", response.text)

  except Exception as e:
        print("Error:", str(e))
```

Figura 26: Código de la importación de API

En el codigo hemos utilizados 2 bibliotecas que los ofrecen Python que son ( requests, json)

- **Requests:** esta biblioteca permite facilitar el envío de solicitudes HTTP y la gestión de respuestas, también facilita la interacción con servicios web, la realización de los operaciones CRUD y realización de llamadas a APIs.

- **JSON**: utilizamos esta biblioteca para podemos utilizar datos de formatos JSON.

Para hacer la importación hemos utilizado la operación **POST** porque vamos a hacer la creación de los entidades por la primera vez, para la actualización de estos entidades vamos a utilizar otro código.

Y después de ejecutar la programa, vamos a utilizar la operación **GET** para verificar que los datos están bien importados.

```
root@lpfc2004:/home/usuario/lets-fiware.tutorials/wirecloud# curl -X GET http://localhost:1026/v2/entities -H 'fiware-service: SmartCity' -H 'fiware-servicepath: /valenbici'
```

Figura 27: ejecución de la operación GET

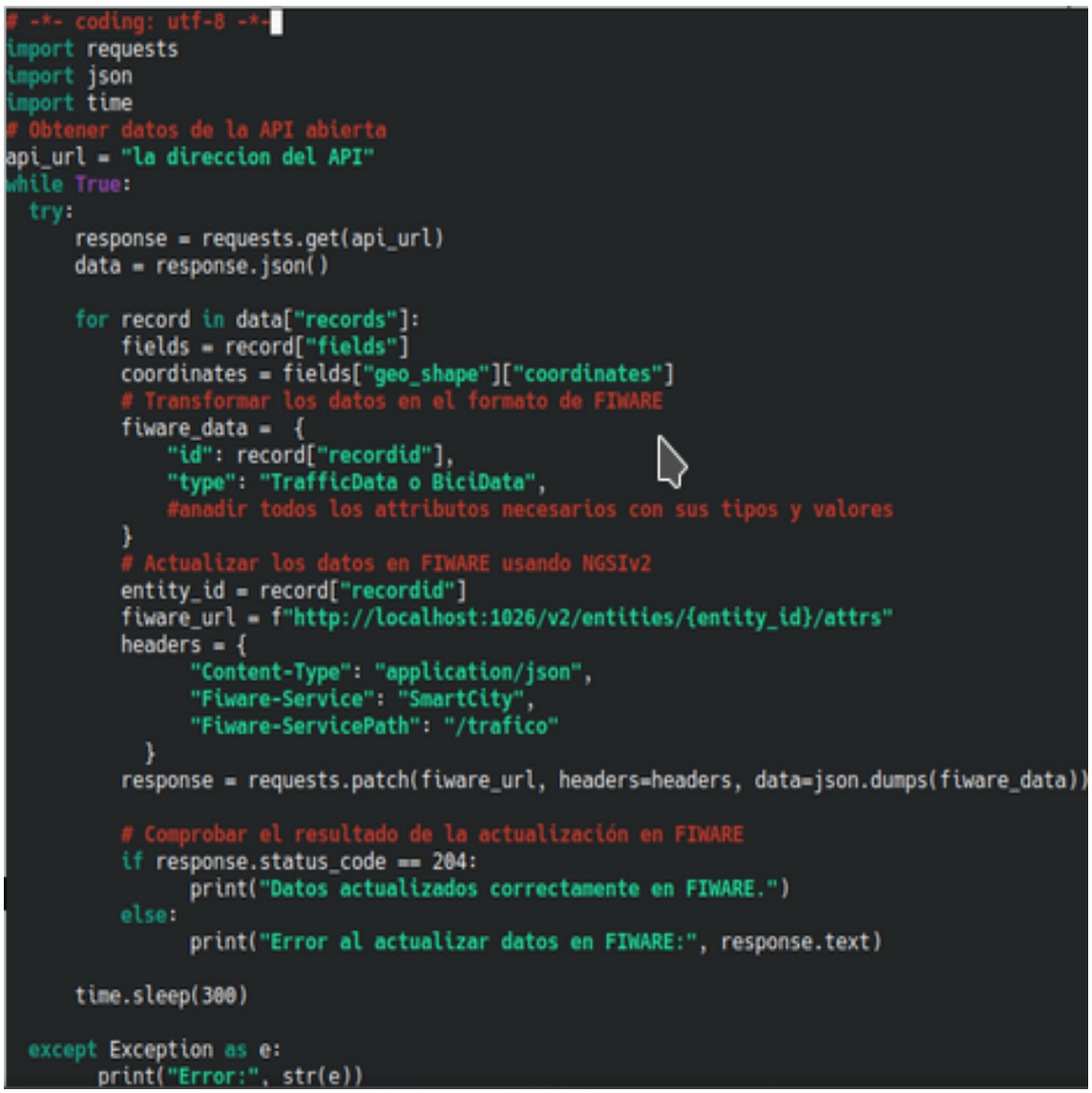
Y ya los datos están bien importados.


```
root@lpfc2004:/home/usuario/lets-fiware.tutorials/wirecloud# curl -X GET http://localhost:1026/v2/entities -H 'fiware-service: SmartCity' -H 'fiware-servicepath: /valenbici'
[{"id":"8e97cd0be70f01eb0621c562ec6236532b0faa41","type":"BiciData","address":{"type":"Text","value":"Regne de Valencia - Doctor Sumsi","metadata":{}},"available":{"type":"Number","value":3,"
metadata":{}},"free":{"type":"Number","value":14,"metadata":{}},"location":{"type":"geo:json","value":{"type":"Point","coordinates":[-0.369961373,39.464122297]},"metadata":{}},"number":{"type
":"Number","value":34,"metadata":{}},"open":{"type":"Text","value":"T","metadata":{}},"record_timestamp":{"type":"DateTime","value":"2023-06-13T10:20:04.00Z","metadata":{}},"ticket":{"type":"
Text","value":"F","metadata":{}},"total":{"type":"Number","value":17,"metadata":{}}},{"id":"35954c1f7e63721e6788e0a5337265a2b1c96fff","type":"BiciData","address":{"type":"Text","value":"Gener
al Urrutia - Av. de la Plata","metadata":{}},"available":{"type":"Number","value":5,"metadata":{}},"free":{"type":"Number","value":14,"metadata":{}},"location":{"type":"geo:json","value":{"ty
pe":"Point","coordinates":[-0.363014355,39.456429268]},"metadata":{}},"number":{"type":"Number","value":44,"metadata":{}},"open":{"type":"Text","value":"T","metadata":{}},"record_timestamp":{
"type":"DateTime","value":"2023-06-13T10:20:04.00Z","metadata":{}},"ticket":{"type":"Text","value":"T","metadata":{}},"total":{"type":"Number","value":19,"metadata":{}}},{"id":"9e301e124918c8
278d1abc109e0b569cca47acd1","type":"BiciData","address":{"type":"Text","value":"Guillem de Anglesola - Av. Puerto","metadata":{}},"available":{"type":"Number","value":1,"metadata":{}},"free":
{"type":"Number","value":24,"metadata":{}},"location":{"type":"geo:json","value":{"type":"Point","coordinates":[-0.346261292,39.464271285]},"metadata":{}},"number":{"type":"Number","value":66
,"metadata":{}},"open":{"type":"Text","value":"T","metadata":{}},"record_timestamp":{"type":"DateTime","value":"2023-06-13T10:20:04.00Z","metadata":{}},"ticket":{"type":"Text","value":"F","me
tadata":{}},"total":{"type":"Number","value":25,"metadata":{}}},{"id":"f2992a19b91407cb1df4341394a9938192ebb354","type":"BiciData","address":{"type":"Text","value":"Blasco Ibañez - Aragón","m
etadata":{}},"available":{"type":"Number","value":20,"metadata":{}},"free":{"type":"Number","value":0,"metadata":{}},"location":{"type":"geo:json","value":{"type":"Point","coordinates":[-0.35
5968316,39.475860329]},"metadata":{}},"number":{"type":"Number","value":92,"metadata":{}},"open":{"type":"Text","value":"T","metadata":{}},"record_timestamp":{"type":"DateTime","value":"2023-
06-13T10:20:04.00Z","metadata":{}},"ticket":{"type":"Text","value":"T","metadata":{}},"total":{"type":"Number","value":20,"metadata":{}}},{"id":"f02b4e186c63056b224e46cb4ed0996118c903e9","typ
```


Figura 28: Datos importados

### 4.2. Actualización de datos

El código de la actualización es muy similar de la importación. Solamente es diferente en tipo de la operación CRUD y dirección de las entidades que vamos a actualizarlos.



```python
# -*- coding: utf-8 -*-
import requests
import json
import time
# Obtener datos de la API abierta
api_url = "la direccion del API"
while True:
  try:
      response = requests.get(api_url)
      data = response.json()

      for record in data["records"]:
          fields = record["fields"]
          coordinates = fields["geo_shape"]["coordinates"]
          # Transformar los datos en el formato de FIWARE
          fiware_data =  {
              "id": record["recordid"],
              "type": "TrafficData o BiciData",
              #anadir todos los attributos necesarios con sus tipos y valores
          }
          # Actualizar los datos en FIWARE usando NGSIv2
          entity_id = record["recordid"]
          fiware_url = f"http://localhost:1026/v2/entities/{entity_id}/attrs"
          headers = {
                "Content-Type": "application/json",
                "Fiware-Service": "SmartCity",
                "Fiware-ServicePath": "/trafico"
            }
          response = requests.patch(fiware_url, headers=headers, data=json.dumps(fiware_data))

          # Comprobar el resultado de la actualización en FIWARE
          if response.status_code == 204:
                print("Datos actualizados correctamente en FIWARE.")
          else:
                print("Error al actualizar datos en FIWARE:", response.text)

      time.sleep(300)

  except Exception as e:
        print("Error:", str(e))
```

Figura 29: código de actualización

También hemos utilizado otra biblioteca que es **Time** para que el programa se actualizara cada 5 minutos.

### 4.3. Eliminación de datos

Si queremos eliminar unas entidades podemos utilizar este código.

```
import requests

url = "http://localhost:1026/v2/entities?type=TrafficData"
headers = {"Fiware-Service": "SmartCity", "Fiware-ServicePath": "/trafico"}

response = requests.get(url, headers=headers)

if response.status_code == 200:
    entities = response.json()
    for entity in entities:
        entity_id = entity["id"]
        delete_url = f"http://localhost:1026/v2/entities/{entity_id}"
        delete_response = requests.delete(delete_url, headers=headers)
        if delete_response.status_code == 204:
            print(f"Entidad con el ID: {entity_id} esta eliminada")
        else:
            print(f"Error para eliminar la entidad con el ID: {entity_id}")
else:
    print("Error al recuperar entidades para su eliminación")
```

Figura 30: código de eliminación

Después de verificar que todos los datos están bien insertados en FIWARE. Ahora podemos pasar a la configuración de Wirecloud para podemos representarlos de diferentes formas.

### 4.4. Visualización de datos

Para la visualización de los datos de FIWARE vamos a utilizar varias formas como tabla, mapa y gráfico.

#### 4.4.1. Visualización de datos en forma de tabla

En este parte utilizaremos el widget **NGSI Browser** para ver los datos en forma de tabla. Hay que que configurar el widget para poder verlos.

empezamos con la configuración de widget para el trafico.

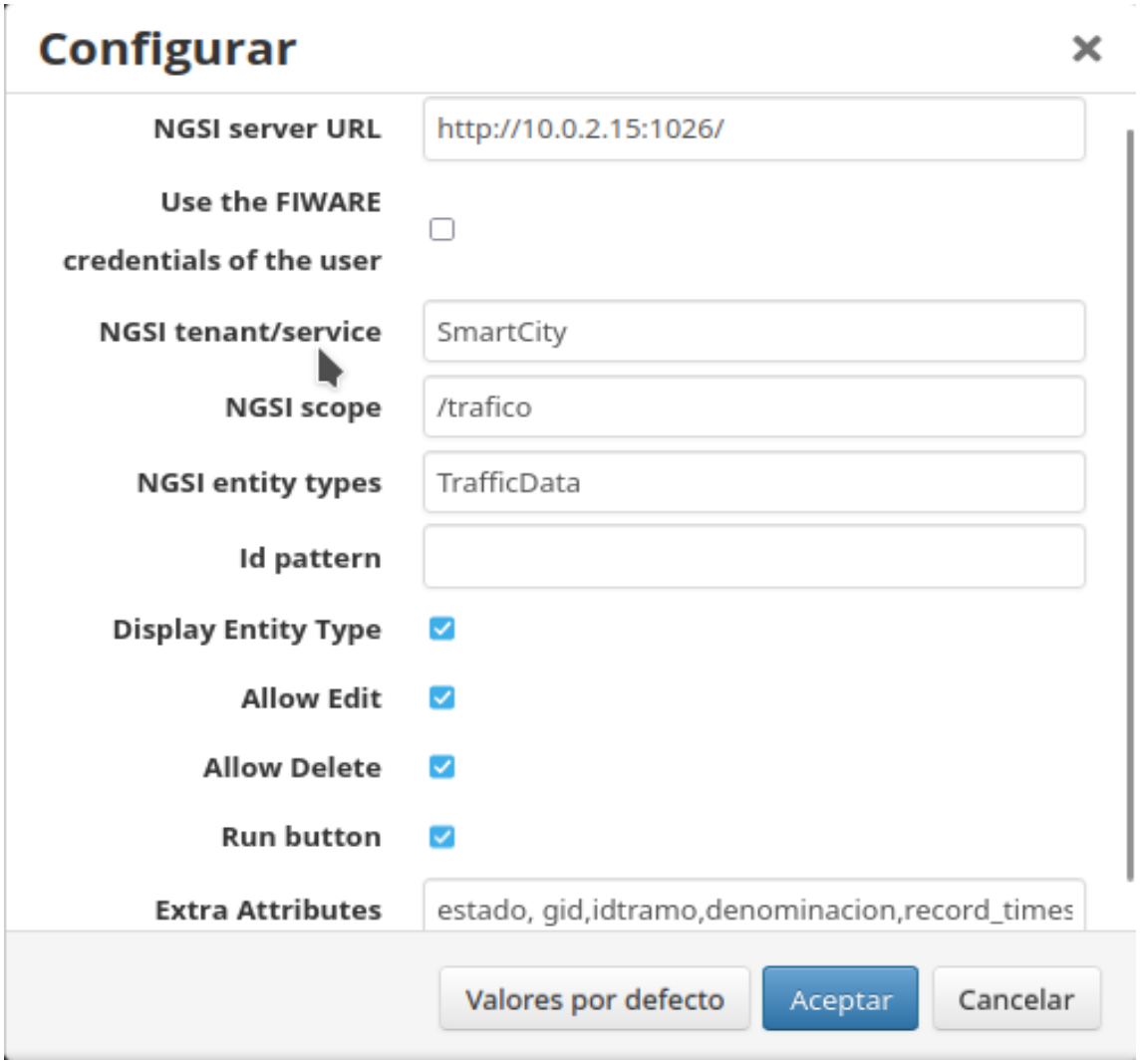



Figura 31: Configuración de NGSI Browser para el trafico

Las partes obligatorias de configuración son :

**NGSI server URL:** debe que tener la dirección de la servidor NGSI de FIWARE.

**NGSI entity types:** debe que tener tipo de entidades que vamos a utilizarlas.

Y las partes opcionales son :

**NGSI tenant/service:** Si hemos definido Fiware-Service en la importación debe que poner el nombre del servicio.

**NGSI scope:** Si también hemos definido Fiware-ServicePath en la importación debe que poner el nombre de la dirección del servicio.

**Extra Attributes:** por defecto solamente se muestra el id y tipo pero si queremos que mostrar otras atributos podemos ponerlos en este parte.

Para la configuración de widget de la disponibilidad de bicicletas esta la misma que el trafico.

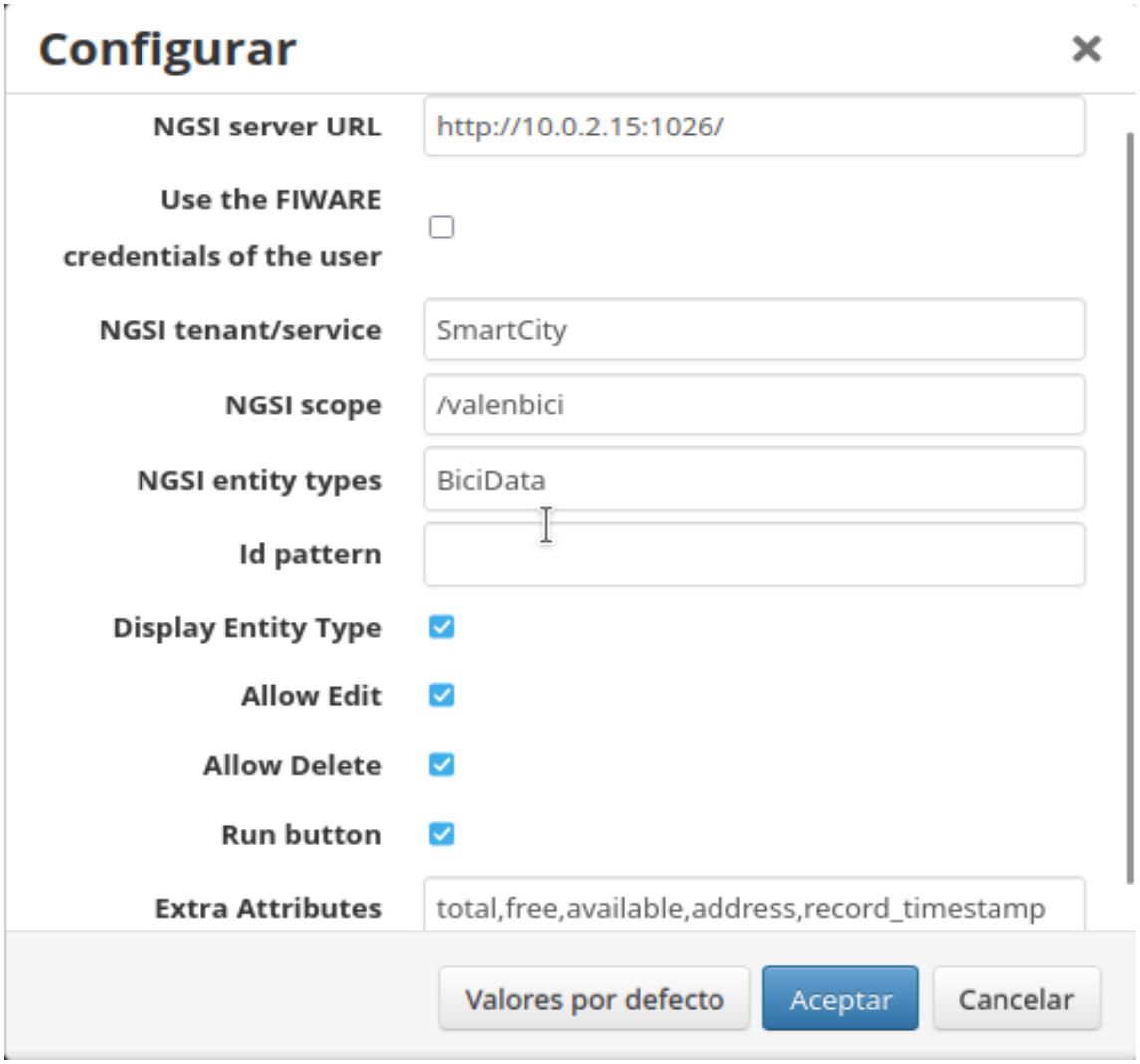



Figura 32: Configuración de NGSI Browser para la disponibilidad de bicicletas

Finalmente el resultado va a salir como muestra la figura 33 para el trafico y el 34 para la disponibilidad de bicicletas.

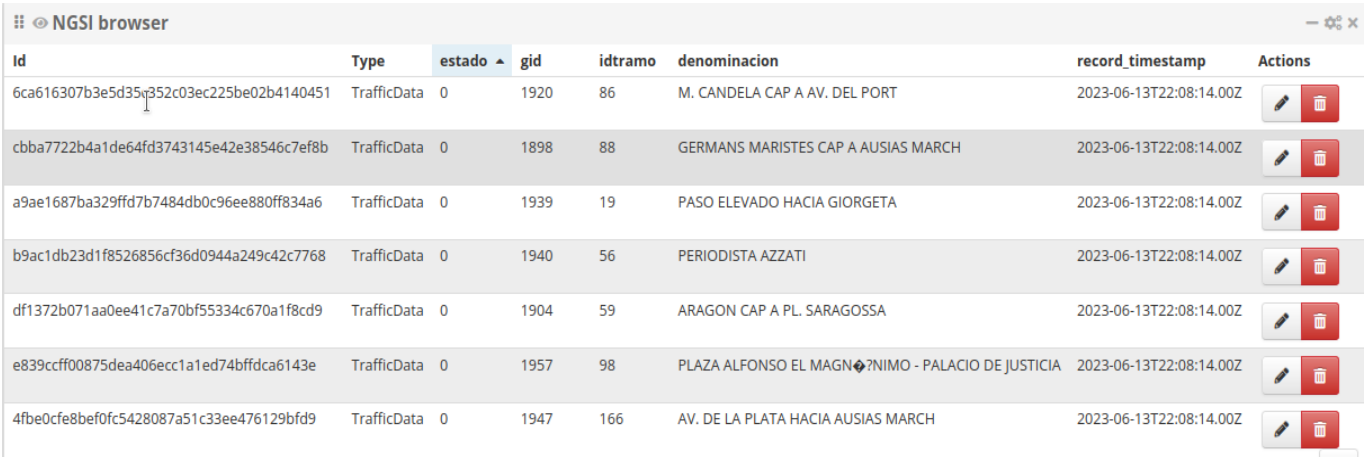



Figura 33: Resultado final del trafico en forma de tabla

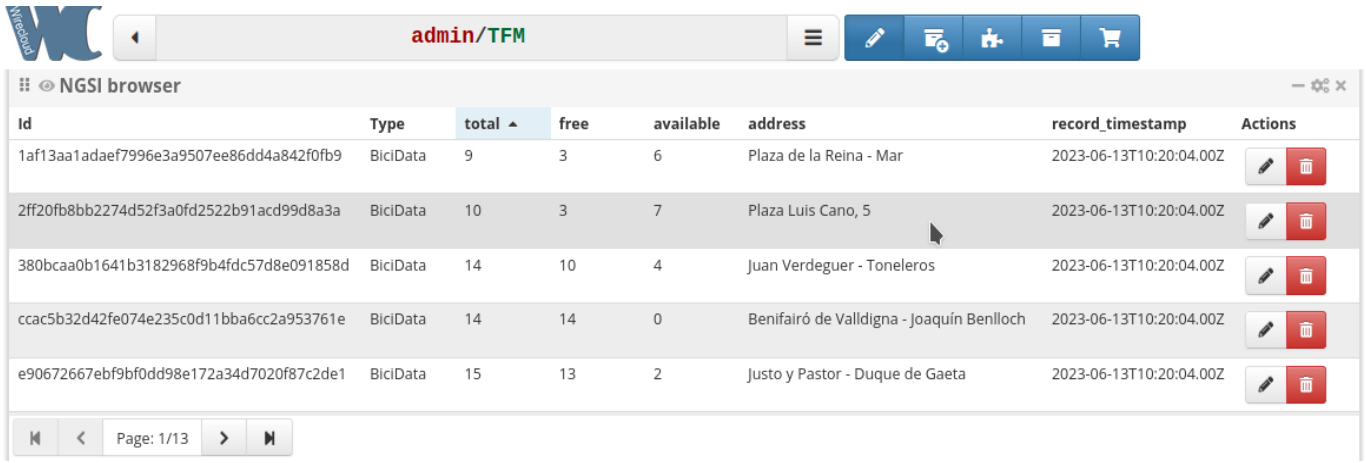



Figura 34: Resultado final de la disponibilidad de bicicletas en forma de tabla

### 4.4.2. Visualización de datos en forma de mapa

En este parte utilizaremos el widget **OpenLayres Map** y los operadores **NGSI Source** y **NGSI Entity To POI** para ver los datos en mapa, entonces debe que configurar el widget y los operadores para poder verlos.

Empezamos con la configuración de los operadores:

> **NGSI Source:** tiene la misma configuración que el NGSI Browser solamente tiene el **NGSI Proxy URL** mas, que va a tener la dirección del proxy.


Figura 35: Configuración del operador NGSI Source del trafico

Y para la disponibilidad del bicicletas solamente cambiamos el Fiware-Service con **BiciData** y Fiware-ServicePath con /**valenbici**.

**NGSI Entity To POI:** este operador va a tener el atributo que tiene las coordenadas.


Figura 36: Configuración del operador NGSI Entity To POI del trafico

Y para las coordenadas de las bicicletas están definidas en el atributo **location**.

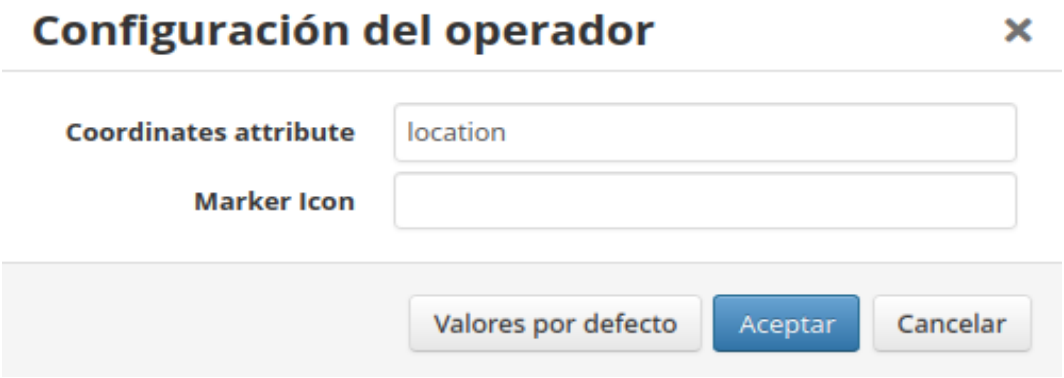



Figura 37: Configuración del operador NGSI Entity To POI de la disponibilidad de bicicletas

La configuración del widget de mapa:

**OpenLayers Map:** como que nuestros datos están localizados en Valencia entonces debe que inicializar la localización en las coordinadas de valencia que son **-0.3763, 39.4699**.


Figura 38: Configuración del widget OpenLayers Map

Finalmente, después de este configuración podemos ver el estado de trafico y la disponibilidad de bicicletas en Valencia en mapa.

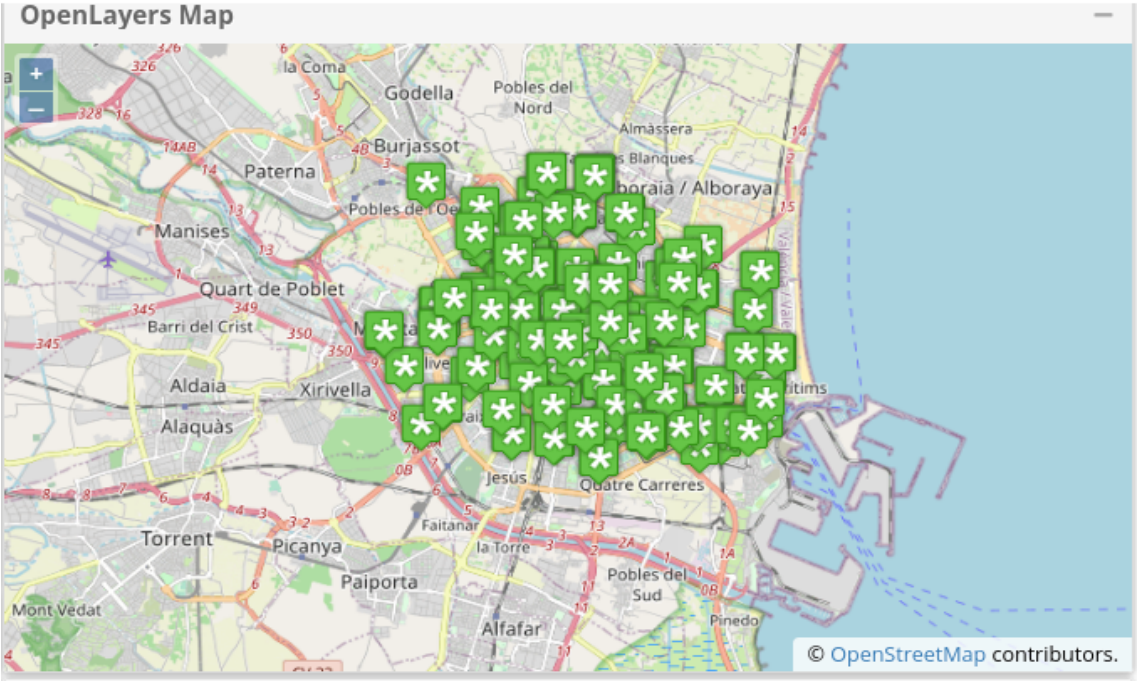



Figura 39: Resultado final de la visualización de datos en forma de mapa

En cada punto va a mostrar todos las informaciones de cada entidad.

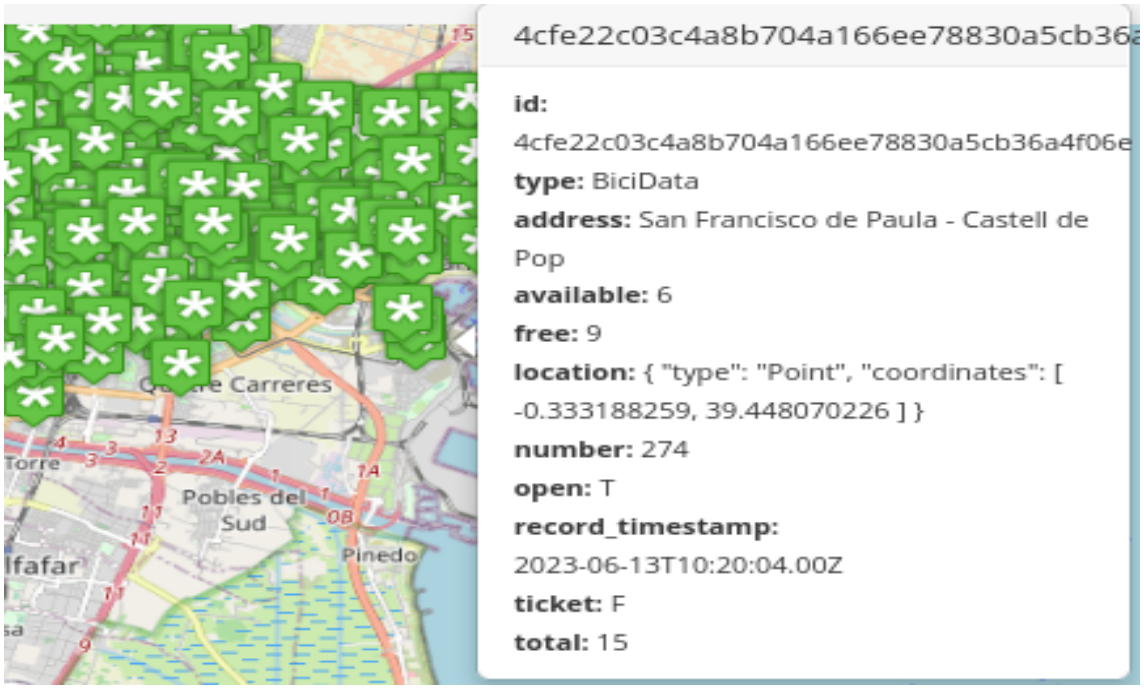



Figura 40: Información de un punto

### 4.4.3. Visualización de datos en forma de gráfico

Para la visualización de datos en forma de gráfico vamos a utilizar 4 operadores (NGSI Source, Value List Filter,Labels to dataseries operator, Column chart generator) y un widget (Highcharts). Empezamos con la configuración de los operadores :

**NGSI Source:** tiene la misma configuración que la visualización en forma de tabla y mapa.

**Value List Filte:** la configuración este operador tiene el atributo con el que queremos crear el gráfico. En este gráfico vamos a crear la gráfico con el estado del trafico en tiempo real.


Figura 41: Configuración del operador Value List Filte

**Labels to dataseries operator:** Este operador no tiene configuración porque convierte por defecto un conjunto de etiquetas en un conjunto de números.

**Column chart generator:** Este operador tiene la configuración del titulo del gráfico.

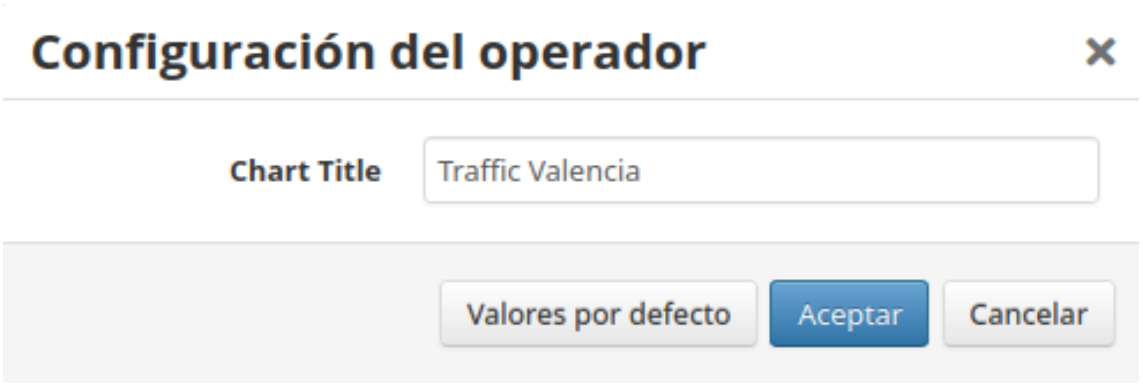



Figura 42: Configuración del operador Column chart generator

Después de la configuración podemos ver el resultado final como muestra la figura 43.

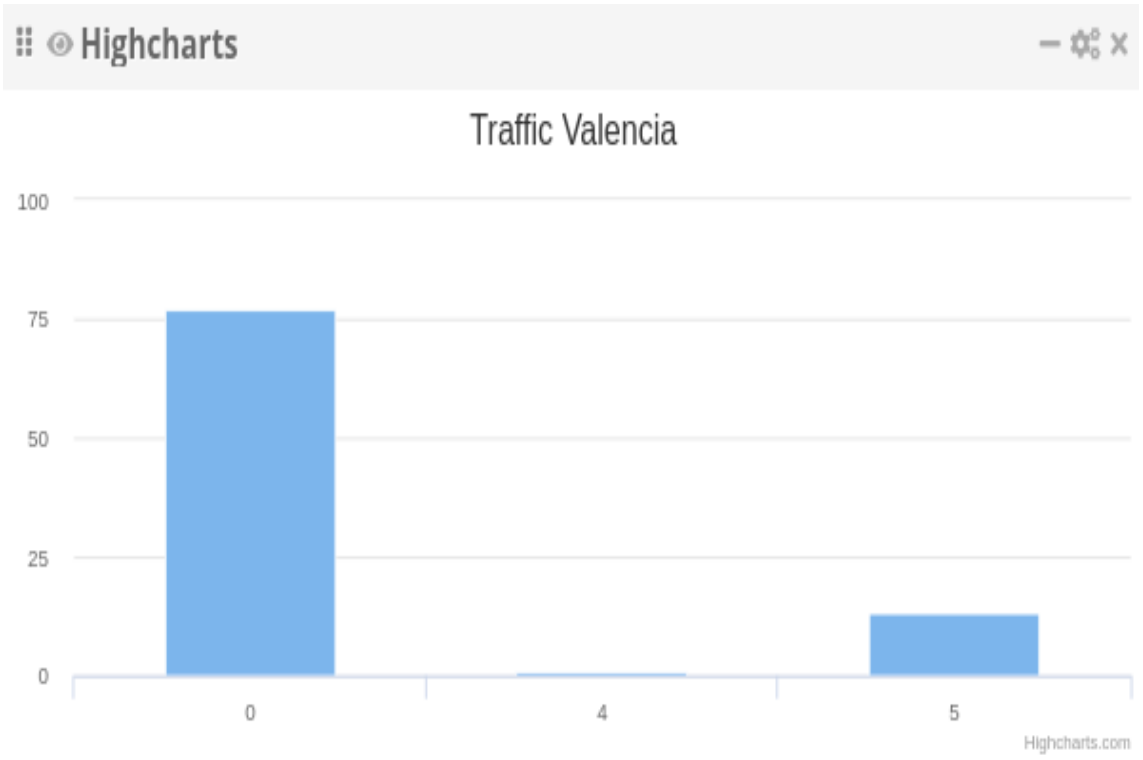



Figura 43: Resultado final de la visualización de datos en forma de gráfico

# 5. Conclusión

Nuestro objetivo en este trabajo fin de máster fue implementar y evaluar la capacidad de FIWARE para interactuar con otras plataformas existentes y poder solicitar y proporcionar información desde ellas.

Para realizar este trabajo , comenzamos con un estudio en profundidad de FIWARE, Wirecloud, las tecnologías y las herramientas que vamos a utilizar en el proyecto. Tras este estudio y comprender lo que vamos a hacer, en primer lugar comenzamos a instalar el entorno mediante Docker-compose que contiene todos los servicios necesarios del entorno.

En segundo lugar, creamos algunos programas para poder importar y actualizar los datos públicos del Ayuntamiento de Valencia que vamos a utilizar en FIWARE.

Una vez que todos los datos necesarios estaban en FIWARE, con el fin de mejorar la visualización de los datos FIWARE, hemos configurado Wirecloud, los widgets y operadores necesarios para visualizar los datos.

Para mejorar el trabajo podemos realizar un aplicación móvil, puede ser una excelente manera de mejorar la accesibilidad y la experiencia de los usuarios al interactuar con los datos.

FIWARE ha demostrado ser una plataforma sólida para el desarrollo de aplicaciones IoT . Su capacidad de interoperabilidad, gestión de datos y enfoque en estándares abiertos la posicionan como una herramienta versátil y prometedora para impulsar el avance del Internet de las Cosas.

En el futuro, se espera que FIWARE siga evolucionando para adaptarse a los avances tecnológicos y las necesidades cambiantes del Internet de las Cosas. Se espera que se integre con tecnologías emergentes como la inteligencia artificial, el aprendizaje automático y el procesamiento de datos en tiempo real para mejorar aún más la capacidad de análisis y toma de decisiones de las aplicaciones IoT.

## Referencias


[1] Definicion IoT, mayo de 2023. Disponible en: https://www.techtarget.com/iotagenda/definition/Internet-of-Things-IoT

[2] Arquitectura IoT, mayo de 2023. Disponible en:https://ieeexplore.ieee.org/document/5579493

[3] FIWARE, abril de 2023. Disponible en:https://fiware-training.readthedocs.io/es_MX/latest/ecosistemaFIWARE/plataformaFIWARE/

[4] FIWARE Hub, mayo de 2023. Disponible en:https://www.fiware.org/developers/fiware-lab/

[5] FIWARE accelerator, mayo de 2023. Disponible en:https://www.fiware.org/community/fiware-accelerator-programme/

[6] FIWARE Mundus, mayo de 2023. Disponible en:https://www.fiware.org/wp-content/uploads/2016/12/FIWARE-Mundus-Committee-The-Road-Ahead-Malaga-2016-12-15.pdf

[7] FIWARE Hub, mayo de 2023. Disponible en:https://www.fiware.org/community/fiware-ihubs/

[8] IDAS, abril de 2023. disponible en:https://www.fiware.org/catalogue/

[9] LWM2M, junio de 2023. disponible en:https://1nce.com/en-eu/resources/news-insights/blog/mqtt-coap-lwm2m

[10] JSON, junio de 2023. Disponible en:https://fiware-zone.readthedocs.io/es/latest/iot-agent-json.html

[11] Ultralight, junio de 2023. disponible en:https://fiware-zone.readthedocs.io/es/latest/iot-agent.html

[12] OPC-UA, junio de 2023. Disponible en:https://www.cognex.com/es-es/blogs/machine-vision/why-opc-ua-is-essential-for-factory-automation

[13] Sigfox, junio de 2023. Diponible en:https://sigfox.com.py/que-es-sigfox/

[14] LORAWAN, junio de 2023. Disponible en:https://blogthinkbig.com/lorawan-ventajas-usos-telefonica

[15] Orion Context Broker, abril de 2023. Disponible en:https://fiware-training.readthedocs.io/es_MX/latest/ecosistemaFIWARE/ocb/

[16] NGSIv2, Abril de 2023. Disponible en:https://www.fiware.org/wp-content/uploads/2017/01/NGSIv2-Overview-for-Developers-That-Already-Know.pdf

[17] CRUD,mayo de 2023. Disponible en:https://fiware-tutorials.readthedocs.io/en/1.0.0/crud-operations/

[18] MongoDB, abril 2023. Disponible en:https://es.wikipedia.org/wiki/MongoDB/

[19] CKAN, mayo de 2023. Disponible en:https://docs.ckan.org/en/2.10/user-guide.html#what-is-ckan

[20] Docker, abril de 2023. Disponible en:https://www.ibm.com/es-es/topics/docker

[21] Wirecloud, mayo de 2023. Disponible en:https://wirecloud.readthedocs.io/en/stable/

[22] Widget, junio de 2023. Disponible en:https://wirecloud.readthedocs.io/en/stable/widgets