REVISTA CUBANA
DE TRANSFORMACIÓN DIGITAL

RNPS 2487 • ISSN 2708-3411
Vol. 3 • Nro. 1 • enero-marzo 2022 • e132

ARTÍCULO ORIGINAL

API para el desarrollo de aplicaciones IoT personalizadas usando FIWARE

API for the development of custom IoT applications using FIWARE

Dalia María Berbes Villalón
dmberbes@gmail.com • https://orcid.org/0000-0002-0693-3819

Laura Sánchez Jiménez
laurasanchez0614@gmail.com • https://orcid.org/0000-0003-3486-9292

María Elena Díaz Aguirre
mdaguirre@gmail.com • https://orcid.org/0000-0003-2975-3404

Tatiana Delgado Fernández
tatiana.delgado@uic.cu • https://orcid.com/0000-0002-4323-9674

UNIVERSIDAD TECNOLÓGICA DE LA HABANA JOSÉ ANTONIO HECHEVERRÍA (CUJAE), CUBA
UNIÓN DE INFORMÁTICOS DE CUBA

Recibido: 2021-11-08 • Aceptado: 2022-03-31

RESUMEN

En los últimos años, el surgimiento del concepto de Internet de las cosas ha impulsado la creación y el desarrollo de plataformas que permiten procesar grandes volúmenes de información, producida por redes de sensores que monitorean espacios o ambientes inteligentes. FIWARE es una plataforma de internet de las cosas de código abierto que impulsa la creación de estándares para el desarrollo de aplicaciones y servicios inteligentes de diferentes dominios. En este trabajo se presenta el desarrollo de una biblioteca de clases en Java, que hace uso de los estándares, componentes genéricos y servicios Rest APIs que ofrece la plataforma FIWARE, y que permite facilitar el uso de estos en el desarrollo de aplicaciones de Internet de las cosas (IoT por sus siglas en inglés) personalizadas. Para comprobar el correcto funcionamiento de la biblioteca de clases fue desarrollada una aplicación Web que permite la creación y configuración de un entorno de trabajo basado en la plataforma FIWARE, así como el uso y análisis de los datos almacenados en el contexto de la plataforma.

Palabras clave: Biblioteca de clases Java, Componentes IoT, FIWARE, IoT, API.

ABSTRACT

In recent years, the emergence of the concept of the Internet of Things has prompted the creation and development of platforms that allow the processing of large volumes of context information, produced by sensor networks that monitor smart spaces or environments. FIWARE is an open-source Internet of Things platform that promotes the creation of standards for the development of intelligent applications and services from different domains. In this work, the development of a Java class library is presented, which makes use of the standards, generic components and APIs offered by the FIWARE platform, and which facilitates the use of these in the development of personalized IoT applications. To verify the correct functioning of the class library, a Web application was developed through which the solution developed is used and that allows the creation and configuration of a work environment based on the FIWARE platform, as well as the use and analysis of the data stored in the context of the platform.

Keywords: *Java class library, IoT Components, FIWARE, IoT, API.*

INTRODUCCIÓN

La visión del IoT describe un futuro en el que muchos de los objetos están interconectados a través de una red global. Recopilan y comparten datos de ellos mismos y de su entorno para permitir un seguimiento, análisis, optimización y control de estos. Hasta hace poco esto era simplemente una visión, pero en los últimos tiempos, se ha convertido lentamente en una realidad (Firouzi, Farahani, Weinberger, DePace, & Aliee, 2020).

El surgimiento y creciente auge del Internet de las cosas ha impulsado la creación y el desarrollo de plataformas especializadas. Estas cuentan con estándares y protocolos para el manejo y procesamiento de los datos captados. Estas características ocasionan que haya un mayor número de desarrolladores enfocados en llevar a cabo esta idea para crear soluciones inteligentes, y un incremento también en las empresas que contribuyen con herramientas, servicios y plataformas (Cubillas-Hernández, Anías-Calderón, & Delgado-Fernández, 2021).

FIWARE es una de las plataformas IoT más usadas, la cual surge como una iniciativa de la Unión Europea que, desde el año 2011, impulsa como una plataforma de Código Abierto para el desarrollo de aplicaciones inteligentes. FIWARE apuesta por el desarrollo colaborativo de soluciones inteligentes (*smart-solutions)* y por tecnologías como el IoT, el *Cloud Computing* y el *Open Data* (FIWARE-Foundation, 2021b).

Las soluciones inteligentes tienen como principal característica que recolectan datos del entorno en el que se desenvuelven. Captando información desde diferentes fuentes como usuarios

finales, redes de sensores, aplicaciones móviles y todo tipo de sistemas de información (FIWA-REmexico.org, 2021).

PLATAFORMA IOT FIWARE

La selección de FIWARE estuvo condicionada por un estudio previo que realizó el equipo para seleccionar una plataforma IoT que permitiera acceder a dispositivos heterogéneos como sensores, actuadores y etiquetas RFID y suministrar sus datos a la capa de aplicación, considerando los requisitos siguientes:

- Ser *OpenSource* y gratis.
- No depender de nubes públicas globales; permitir el empleo de nubes privadas y/o nodos niebla.
- Altamente escalable, contemplando el uso de contenedores.
- Habilitar interoperabilidad semántica.
- Sencilla de usar por desarrolladores.
- Factible para ciudades inteligentes y ambientes industriales.

Para recopilar y administrar información, FIWARE cuenta con un componente central llamado *Orion Context Broker* (OCB). Este componente aporta una función fundamental en cualquier solución inteligente: la necesidad de administrar la información del contexto. El OCB está rodeado por una suite de componentes, que permiten suministrar datos de contexto de diversas fuentes (Internet de las Cosas, robots y sistemas de terceros) y brindan soporte para el procesamiento, análisis y visualización de datos, así como para control de acceso a datos, publicación o monetización (FIWARE-Foundation, 2021 #8). Además, a través del OCB se interactúa con otras plataformas o aplicaciones utilizando la API RESTfull FIWARE NGSI. Esta API es abierta y estándar, por lo que ofrece a los desarrolladores, la capacidad de portar sus aplicaciones con FIWARE (“*Powered by* FIWARE”) y les brinda un marco estable para futuros desarrollos (FIWARE-Foundation, 2021b).

Las soluciones IoT cuentan con un sistema compuesto por dispositivos, servidores de red, protocolos de comunicación, aplicaciones para el procesamiento de datos y aplicaciones donde los usuarios finales utilizan la información obtenida. Esta información se recoge de diferentes fuentes de datos cómo son dispositivos (sensores ambientales, actuadores, electrodomésticos conectados, rastreadores de vehículos o incluso máquinas de línea de montaje.), robots o sistemas de terceros. Los datos capturados por los dispositivos IoT se producen en una combinación de formatos de datos, los cuales pueden ser estructurados, semiestructurados y no estructurados; también pueden incluir señales analógicas, lecturas discretas de sensores, metadatos de estado del dispositivo o archivos grandes para imágenes o videos. Por tanto, homogenizar la comunicación y los datos es un reto. Por lo que haciendo uso de FIWARE, estas dificultades pueden ser solucionadas usando los diferentes componentes y herramientas que rodean al OCB. Desde el conector (Agente IoT), que soluciona las dificultades de heterogeneidad del ambiente donde, desde los dispositivos con diferentes protocolos de comunicación, se unifica la información a un mismo formato (NGSI); hasta los conectores cuyo propósito es

mejorar las capacidades de almacenamiento de la información (herramientas de seguridad, almacenamiento de datos avanzados, retornar información histórica, vincular con aplicación de terceros).

La naturaleza basada en componentes de una solución basada en FIWARE permite extender la arquitectura de una solución, a medida que ésta evoluciona de acuerdo a las necesidades de la solución inteligente (Figura 1) (FIWARE-Foundation, 2021b).

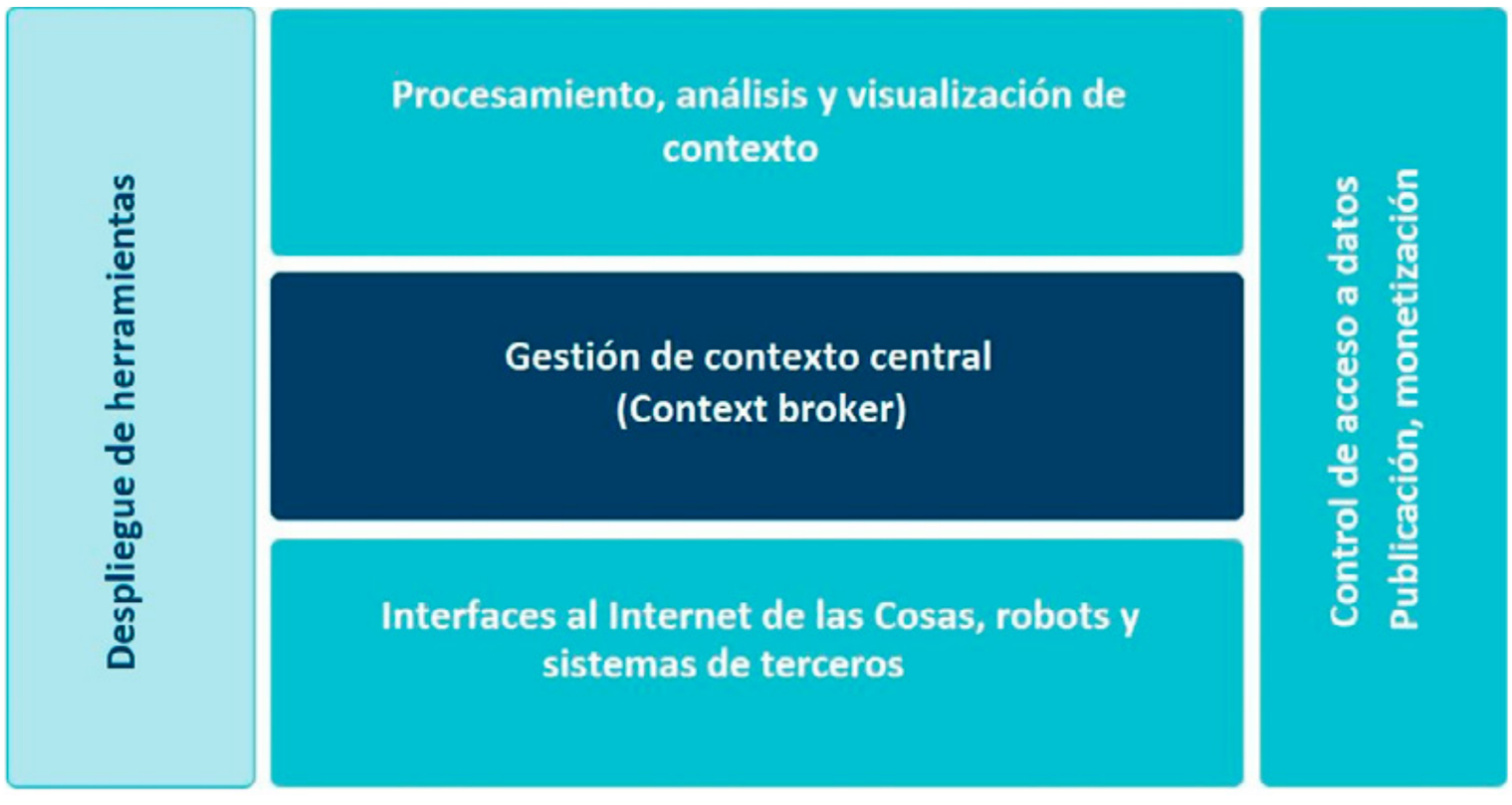


Figura 1: Componentes de plataforma de FIWARE. Tomado de (FIWARE-Foundation, 2021b)

Como se muestra en la Figura 2 el componente central y obligatorio de FIWARE es el *Orion Context Broker* (OCB), este es el principal y único componente obligatorio de cualquier solución. *Orion* es una implementación en *C++* de la API REST NGSIv2. Es la pieza fundamental de la plataforma FIWARE. Proporciona la API FIWARE NGSIv2, la cual es una API *Restful* simple y muy potente que permite realizar actualizaciones, consultas o suscripciones a cambios en la información de contexto. El OCB mantiene la información del contexto actual (FIWARE-Foundation, 2021b), (FIWAREmexico.org, 2021).

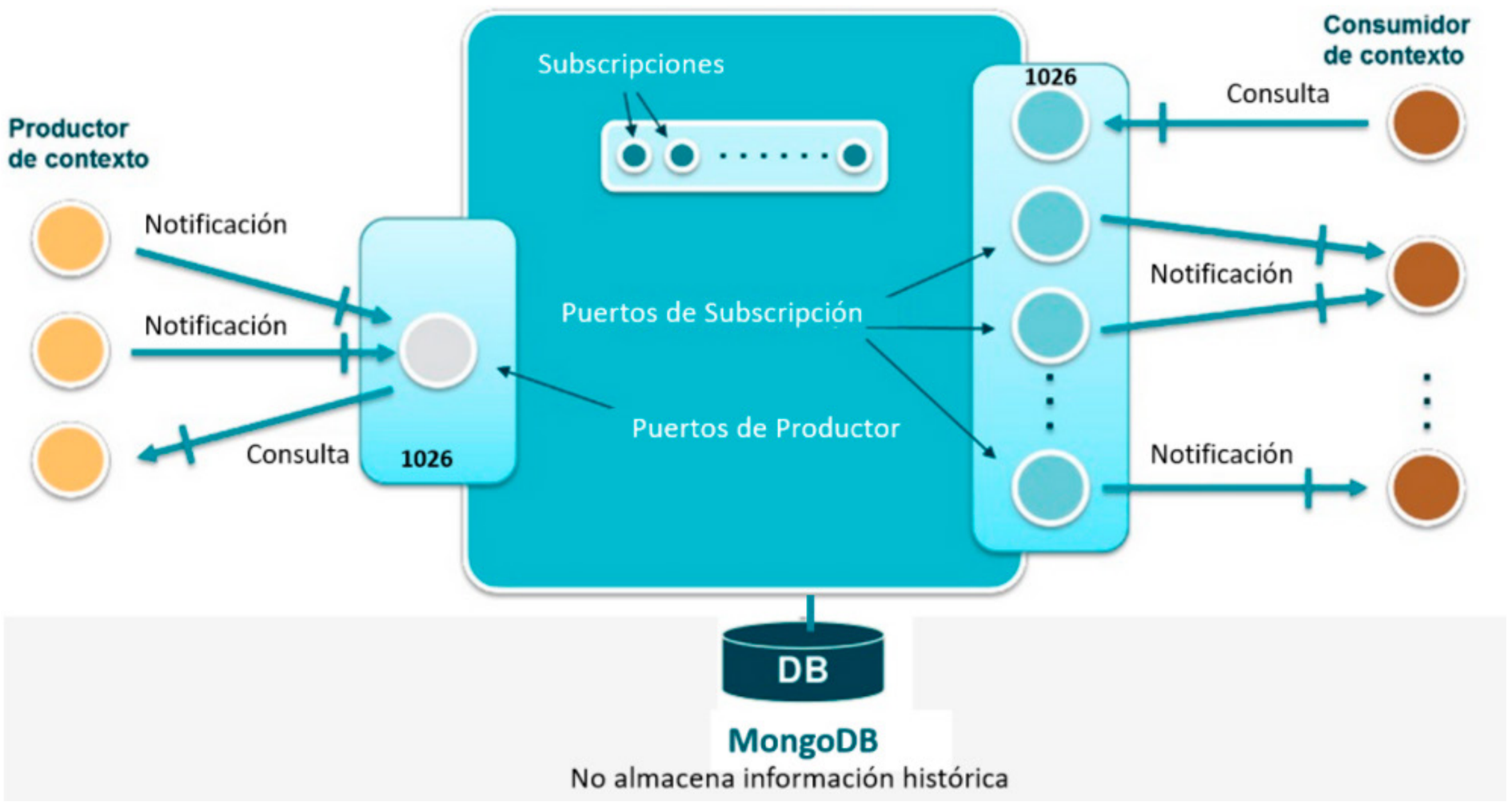


Figura 2: Funcionamiento del OCB de FIWARE (Fuente: (FIWAREmexico.org, 2021))

Un principio fundamental soportado por el OCB es el de lograr una disociación total entre productores y consumidores de contexto. Es decir, los productores de contexto publican datos sin saber qué, dónde y cuándo los consumidores de contexto consumirán los datos publicados; por lo tanto, no necesitan estar conectados a ellos. Por otro lado, los consumidores de contexto consumen información de contexto de su interés, sin que conozcan al productor de contexto que publica un evento en particular. Están interesados en el evento en sí, y no en quien lo generó (FIWARE-Foundation, 2021b).

Usando el API Rest de FIWARE

Las plataformas digitales IoT brindan el soporte para la homologación de la comunicación y la unificación entre los datos captados desde diferentes dispositivos (Zeinab & Elmustafa, 2017). En este vital proceso el uso de un API Rest es fundamental para el intercambio de información entre los dispositivos que se encuentran conectados en la web.

La API Rest estándar que propone FIWARE está basada en la especificación OMA NGSI y permite recopilar, gestionar, publicar e informar sobre cambios en la información de contexto. Esta se denomina FIWARE NGSI (*Next Generation Service Interface,* por sus siglas en inglés) v2.

El estándar FIWARE NGSI está diseñado para describir la administración de todo el ciclo de vida de la información de contexto, incluidas actualizaciones, consultas, registros y suscripciones.

NGISV2 define:

- Un modelo de datos para información de contexto, basado en un modelo de información simple que utiliza la noción de entidades de contexto.
- Una interfaz de datos de contexto para intercambiar información mediante operaciones de consulta, suscripción y actualización.
- Una interfaz de disponibilidad de contexto para intercambiar información sobre cómo obtener información de contexto.

Toda la comunicación entre los distintos componentes de la arquitectura de alto nivel del OCB se realiza a través de la especificación de la API estándar: FIWARE NGSI v2. La información de contexto en FIWARE está representada a través de estructuras de datos genéricos referidos como elementos de contexto. (FIWARE-Foundation, 2021a).

Los modelos de datos se han armonizado según la experiencia trabajando con diferentes servicios en Ciudades Inteligentes y otros dominios. No es un requerimiento usar exactamente todo lo definido en el modelo, solo las partes del modelo de datos que una aplicación realmente necesita. Esto permite reducir la cantidad de información intercambiada, a solamente la que necesita (FIWARE-Foundation, 2021 #11).

FIWARE NGSI V2 describe tres conceptos principales de los modelos de datos NGSI: Entidades, Atributos y Metadatos de Contexto (FIWARE-Foundation, 2021 #8):

Entidades de contexto

- Las entidades de contexto, o simplemente las entidades, son el centro de gravedad en el modelo de información FIWARE NGSI.

- Una entidad representa una cosa, es decir, cualquier objeto físico o lógico (por ejemplo, un sensor, una persona, una habitación, un problema en un sistema de tickets, etc.).
- Cada entidad tiene un identificador.
- FIWARE NGSI permite que las entidades tengan un tipo de entidad. Los tipos de entidad son tipos semánticos; pretenden describir el tipo de cosa representada por la entidad.

Atributos de Contexto

- Los atributos de contexto son propiedades de las entidades de contexto.
- Los atributos tienen un nombre de atributo, un tipo de atributo, un valor de atributo y metadatos.
- El nombre de atributo describe qué tipo de propiedad que representa el valor del atributo de la entidad.
- El tipo de atributo representa el tipo de valor NGSI del valor del atributo.
- El valor de atributo finalmente contiene: el dato actual y opcionalmente los metadatos que describen las propiedades del valor de atributo.

Metadatos de Contexto

De manera similar a los atributos, cada parte de los metadatos tiene:

- Un nombre de metadato
- Un tipo de metadatos, que describe el tipo de valor NGSI del valor de metadatos.

Los datos que componen el JSON de las entidades del modelo de datos NGSI que se envía en la petición HTTP REST son:

- El identificador de la entidad se especifica mediante la propiedad “id” del objeto, cuyo valor es una cadena que contiene el identificador de la identidad.
- El tipo de entidad se especifica por la propiedad “type” del objeto, cuyo valor es una cadena que contiene el nombre del tipo de entidad.
- Los atributos de la entidad se especifican mediante propiedades adicionales:
  - » El nombre del atributo se especifica mediante la propiedad “name”, cuyo valor es una cadena que contiene el nombre del atributo.
  - » El valor del atributo se especifica mediante la propiedad “value”, cuyo valor puede ser cualquier tipo JSON.
  - » El tipo de atributo se especifica por la propiedad “type”, cuyo valor es una cadena que contiene el tipo NGSI.
  - » El atributo metadata se especifica en la propiedad “metadata”. Su valor es otro objeto JSON, que contiene una propiedad por elementos de metadatos definidos (el nombre de la propiedad es el nombre del elemento de metadatos).

En la Figura 3 se puede observar la entidad que representa el dispositivo que mide la geo-localización, así como en la se muestra una entidad que representa el dispositivo que mide los niveles de temperatura y humedad relativa.

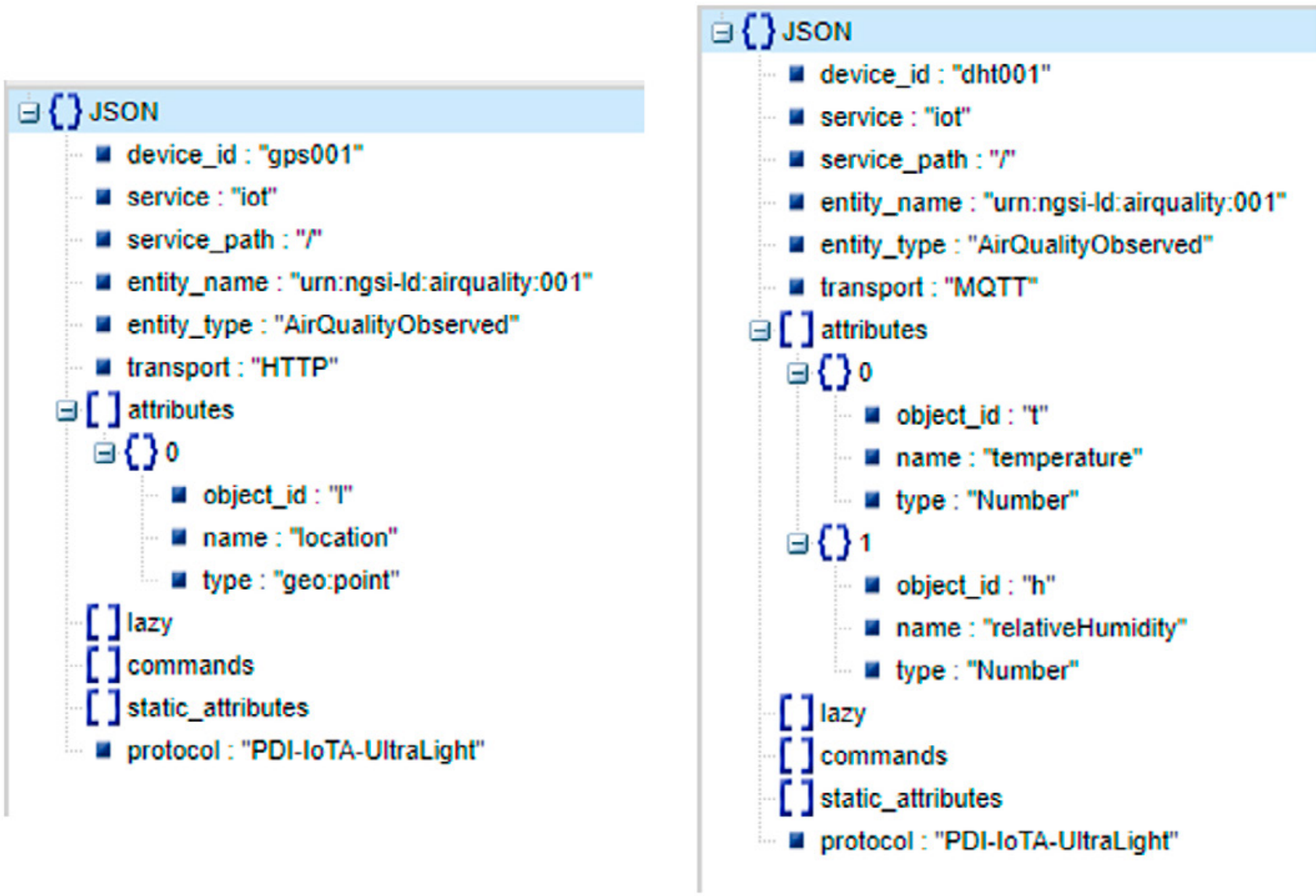


Figura 3. Entidades que representan dispositivos.

Para apoyar la creación de aplicaciones inteligentes en el entorno de desarrollo *Java* se propone la creación de una biblioteca de clases, que permita mejorar el trabajo de integración entre el desarrollo de aplicaciones inteligentes y el API Rest FIWARE NGSI. El uso de esta biblioteca permitirá a los programadores, especializados en esta plataforma de desarrollo (*Java)* y el modelo de programación orientado a objetos, disminuir la carga de trabajo en la programación del *back-end.* Al contar con un conjunto de herramientas que los aísle del trabajo del tratamiento de los datos del contexto, la carga de todas las entidades representadas, obtener la relación entre las entidades y los dispositivos o sensores que forman parte del sistema, minimizar el trabajo del parseo de los objetos JSON que se envían y se reciben, el programador puede centrarse en el trabajo de la solución IoT a desarrollar. Otra de las funcionalidades que favorece el uso de la biblioteca de clases es la carga de los valores históricos de un atributo de una entidad determinada. Estos valores históricos pueden ser ploteados sobre una gráfica o tener representación sobre un mapa. La figura 4 muestra la definición de los objetos que representan una entidad del contexto y el atributo.

Otra facilidad que aporta es que ya no sería un requerimiento el conocimiento técnico de la especificación de la API estándar FIWARE NGSI, abstrayéndose del trabajo de conexión de los end-points o el conocimiento de la estructura de los elementos del contexto.

Además que se incorporan nuevas funcionalidades, que constituyen una extensión de los servicios del API Rest de FIWARE como es el acceso directo a diferentes gestores de base de

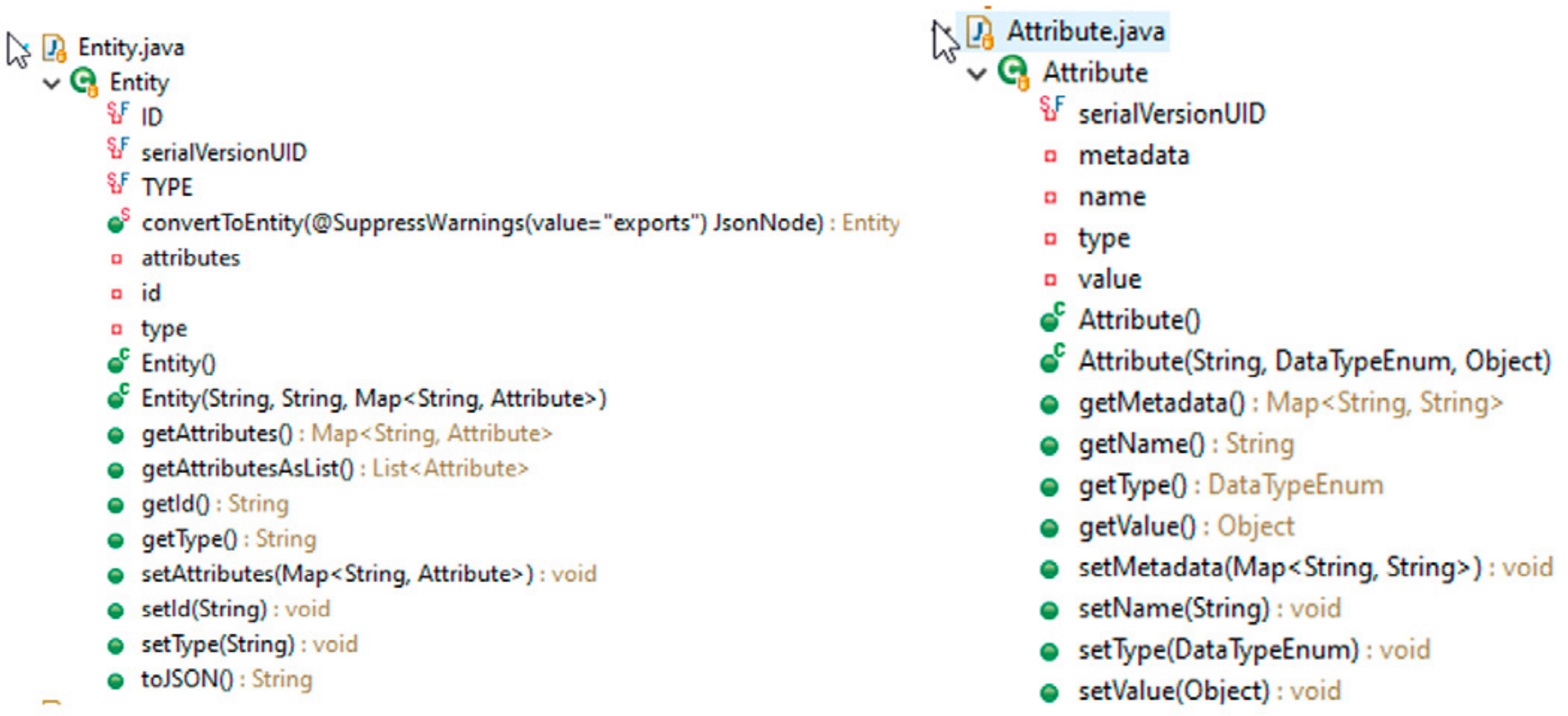


Figura 4. Entidades del contexto representadas en la biblioteca de clases.

datos utilizados para persistir la información captada del entorno para obtener los registros históricos de las mediciones obtenidas. Esto permite obtener y gestionar el histórico de los datos dando la posibilidad de realizar análisis y su monitorización, ya sea a través de representación gráfica o mapas. La figura 5 muestra la definición de clases dentro de la biblioteca que permiten la conexión directa a la base de datos donde se almacenan los valores históricos coleccionados de los dispositivos.

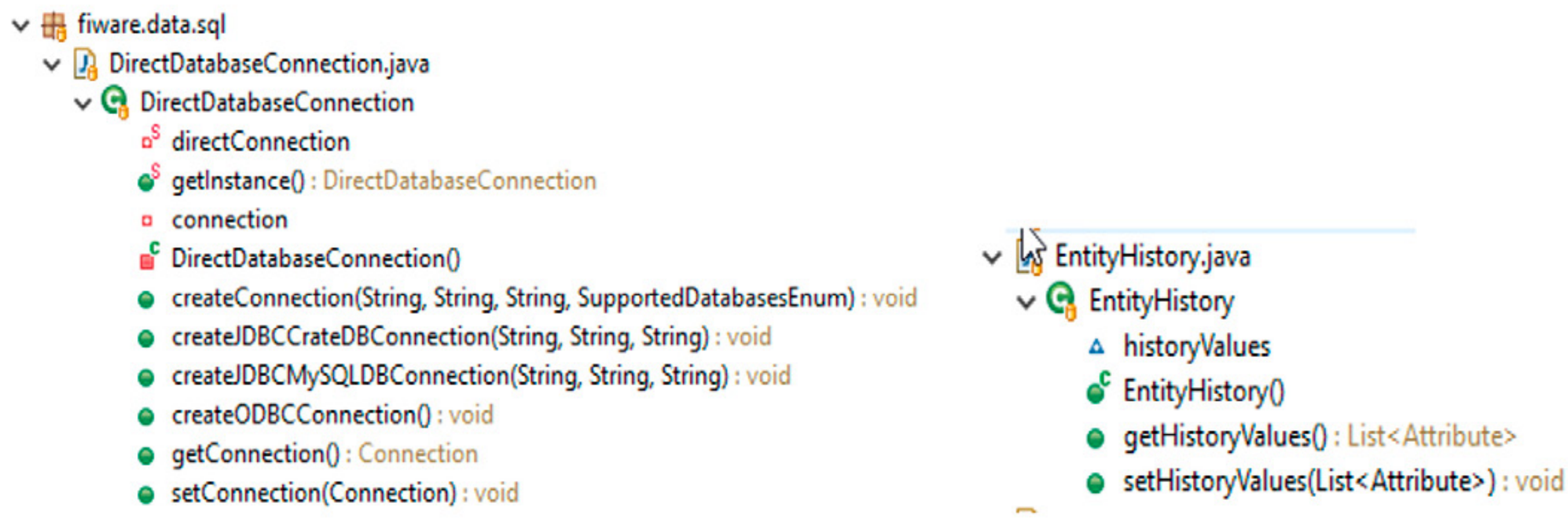


Figura 5. Entidades del contexto representadas en la biblioteca de clases.

RESULTADOS: DESCRIPCIÓN DE LA API DESARROLLADA

La solución propuesta en este trabajo es una biblioteca de clases, creada para desarrolladores *Java*. Las bibliotecas en *Java,* como en la mayoría de los lenguajes de programación, son un recurso que consiste en un conjunto de clases, métodos y atributos que la componen y que facilitan operaciones al brindarle al programador funcionalidades listas para ser utilizadas a través de una Interfaz de Programación para que satisfagan requerimientos específicos durante el proceso de desarrollo en un determinado proyecto. La clave radica en el ahorro de

tiempo que aporta a los implementadores al dedicar su esfuerzo y tiempo en desarrollar las funcionalidades principales de su aplicación (Qiu, Li, & Leung, 2016).

El propósito de esta biblioteca, es contener las funcionalidades que permiten gestionar los elementos de contexto de la plataforma FIWARE permitiendo así que los desarrolladores de soluciones inteligentes se abstraigan del trabajo con la conexión y el tratamiento de las estructuras de los datos en el proceso de intercambio de información entre la aplicación IoT personalizada y la plataforma FIWARE. Permite también agrupar un conjunto de funcionalidades que engloban las principales operaciones realizadas sobre la información del contexto del OCB. De esta forma, el acceso a los servicios proporcionados por este se realiza con mayor comodidad, al transformar todo el trabajo de las peticiones http y el parseo del formato JSON de cada uno de los mensajes de respuesta de la plataforma, hacia el objeto *Java*. Esto permite que el desarrollo de nuevas aplicaciones web y móviles haciendo uso de la tecnología *Java,* sea de forma más natural.

Las principales funcionalidades de la solución propuesta consisten en la gestión de las entidades, así como sus relaciones, gestión de los diversos dispositivos registrados, gestión de las suscripciones, obtención de los valores de los atributos, aplicación de filtros básicos y avanzados, obtención de entidades georreferenciadas, entre otras.

La solución aprovecha los generic enablers (componentes FIWARE) que permiten adicionar nuevas funcionalidades a la plataforma los cuales favorecen el desarrollo de aplicaciones más complejas. Los componentes (*generic enablers)* utilizados en la solución son (Figura 6):

- *Orion Context Broker* (OCB), principal y único componente obligatorio de cualquier solución basada en FIWARE.
- Como el OCB solo almacena la información de contexto actual se utilizó *QuantumLeap* (FIWARE.org, 2021b) para lograr almacenar los valores en una base de datos externa y así poder obtener un histórico, este componente que permite interceptar un cambio en el modelo de datos y lo transforma en registros de una base de datos.
- Para almacenar el histórico de la información se utilizó *CrateDB* (Crate.io, 2021), esta una base de datos SQL distribuida y construida sobre una plataforma NoSQL que controla el almacenamiento, indexación e interconectividad, facilitando una mayor rapidez, flexibilidad y escalabilidad. Debido a estas características, *Quan-*

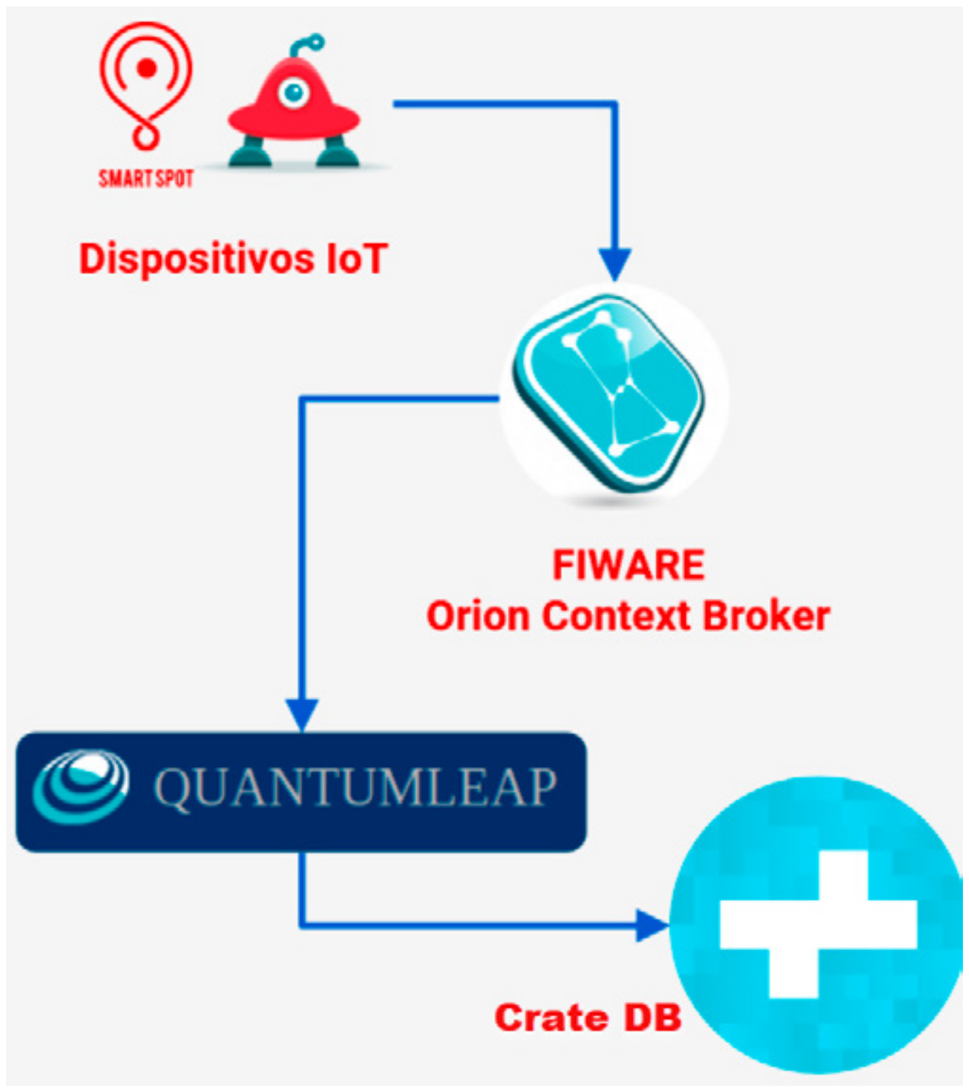


Figura 6: Componentes de la plataforma FIWARE utilizados en la solución.

tumLeap utiliza *CrateDB* como base de datos para almacenar información de series de tiempo, sirviendo como puente entre el mecanismo de notificaciones del OCB y el registro.

- Agente IoT, que es un componente que media entre un conjunto de dispositivos físicos o virtuales que utilizan sus propios protocolos nativos y un proveedor de contexto compatible con NGSI, para la comunicación por MQTT (o sea desde el dispositivo a la plataforma) se necesita de otro componente de software, un mediador, en este caso se utilizó Mosquitto (FIWARE.org, 2021a).

Descripción de la arquitectura de la nueva biblioteca de funcionalidades

La arquitectura de la solución (como se muestra en la Figura 7 consta de 3 capas fundamentalmente, donde:

1. En la capa cliente se encuentran las aplicaciones IoT de propósito general o específico desarrolladas para un cliente.
2. En la capa de aplicación se encuentra el paquete de funcionalidades: *Java API File* que brinda las funcionalidades implementadas en Java que permiten la interacción con la plataforma FIWARE haciendo uso del API de FIWARE. Garantiza una capa de abstracción al desarrollador sobre el uso de la plataforma FIWARE y sus funcionalidades.
3. En la capa de datos se encuentran las fuentes de datos que alimentan la solución cliente, en este caso contamos con la plataforma FIWARE que nos brinda la información en tiempo real de los dispositivos y entidades registradas en ella. También se cuenta con una base de datos series de tiempo en la cual se almacena la información histórica de los valores de los sensores, en este caso el SGBD que se usa es *CrateDB*.

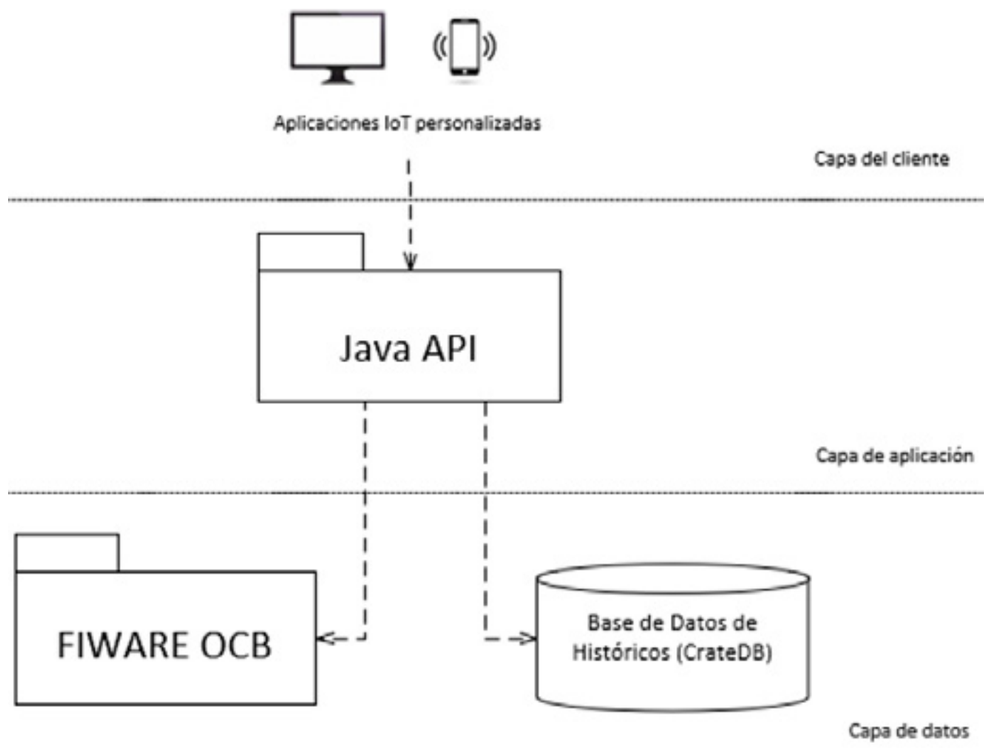


Figura 7: Arquitectura en capas de la solución.

Modelo de datos de la solución

En la biblioteca de clases se modelaron los principales artefactos con los que se trabaja en la plataforma IoT FIWARE y los enablers. Las principales entidades del modelo de datos (Figura 8) de la solución consisten en:

- *Entity:* representa el artefacto entidad dentro del API NGSI
- *Device:* dispositivo que actúa como un sensor que mide los cambios del entorno o un actuador.
- *Subscription:* las subscripciones registradas para la intercepción de las modificaciones sobre los valores de los atributos de las entidades.

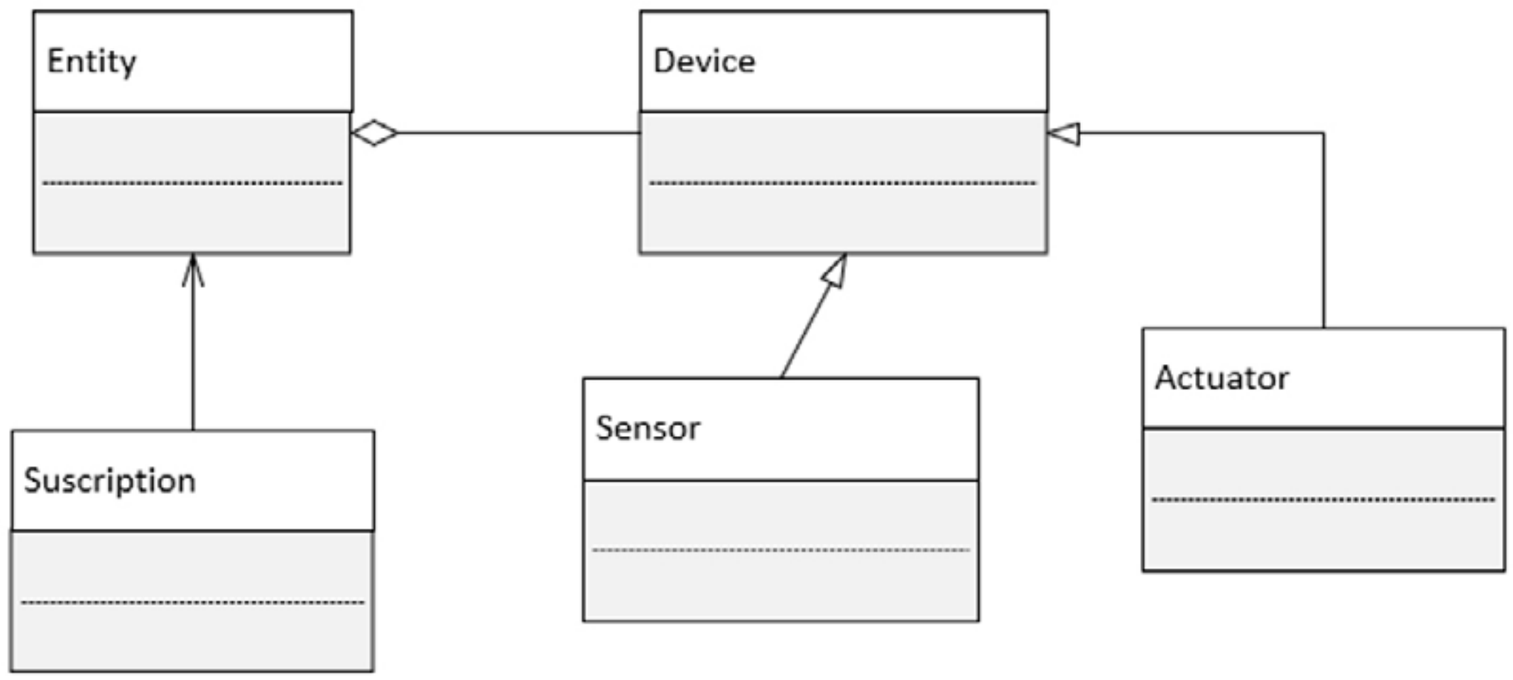


Figura 8:
Modelo de datos de la solución propuesta.

Modelo de despliegue

El procedimiento recomendado para instalar la plataforma FIWARE es usando el contenedor de *Docker* oficial de *Orion* en disponible en *dockerhub*. *Docker* es una plataforma de software que permite crear, probar e implementar aplicaciones rápidamente. Empaqueta el software en unidades estandarizadas llamadas contenedores que incluyen todo lo necesario para que el software se ejecute, incluidas bibliotecas, herramientas de sistema, código y tiempo de ejecución. Con *Docker,* se puede implementar y ajustar la escala de aplicaciones rápidamente en cualquier entorno con la certeza de saber que su código se ejecutará.

Por lo que en la Figura 9 se muestra un diagrama de despliegue donde los contenedores se corresponden a los componentes de la plataforma FIWARE utilizados en la solución propuesta, los cuales se mostrarán como nodos.

Descripción de los nodos del diagrama:

- FIWARE OCB: nodo principal e imprescindible para una solución basada en FIWARE.
- Iot Agente: *IoT Ultralight* es el componente para establecer la comunicación entre los dispositivos y la plataforma; este agente permite tanto la comunicación desde los dispositivos hacia la plataforma enviando mediciones (*Northbound Traffic*), como la comunicación desde la plataforma hacia los dispositivos actuadores que reciben un comando y cambian su estado (*Southbound Traffic).* Para esta solución se implementa la comunicación *Northbound Traffic,* donde el agente tiene la capacidad de reconocer dos protocolos de comunicación: HTTP y MQTT y *Southbound Traffic* el actuador tiene la capacidad de reconocer la comunicación: HTTP.
- *Quantum Leap:* permite el almacenamiento de datos de la API FIWARE NGSIv2 a una base de datos de series de tiempo, conocida como *ngsi-tsdb.* Cuenta con un traductor de *CrateDB.*
- *CrateDB:* base de datos para persistir los datos a lo largo del tiempo.
- *MongoDB:* el OCB se basa en la tecnología de código abierto MongoDB para mantener la persistencia de los datos del contexto.
- *Mosquitto:* es el mediador usado para la comunicación por MQTT entre el dispositivo y el agente IoT.

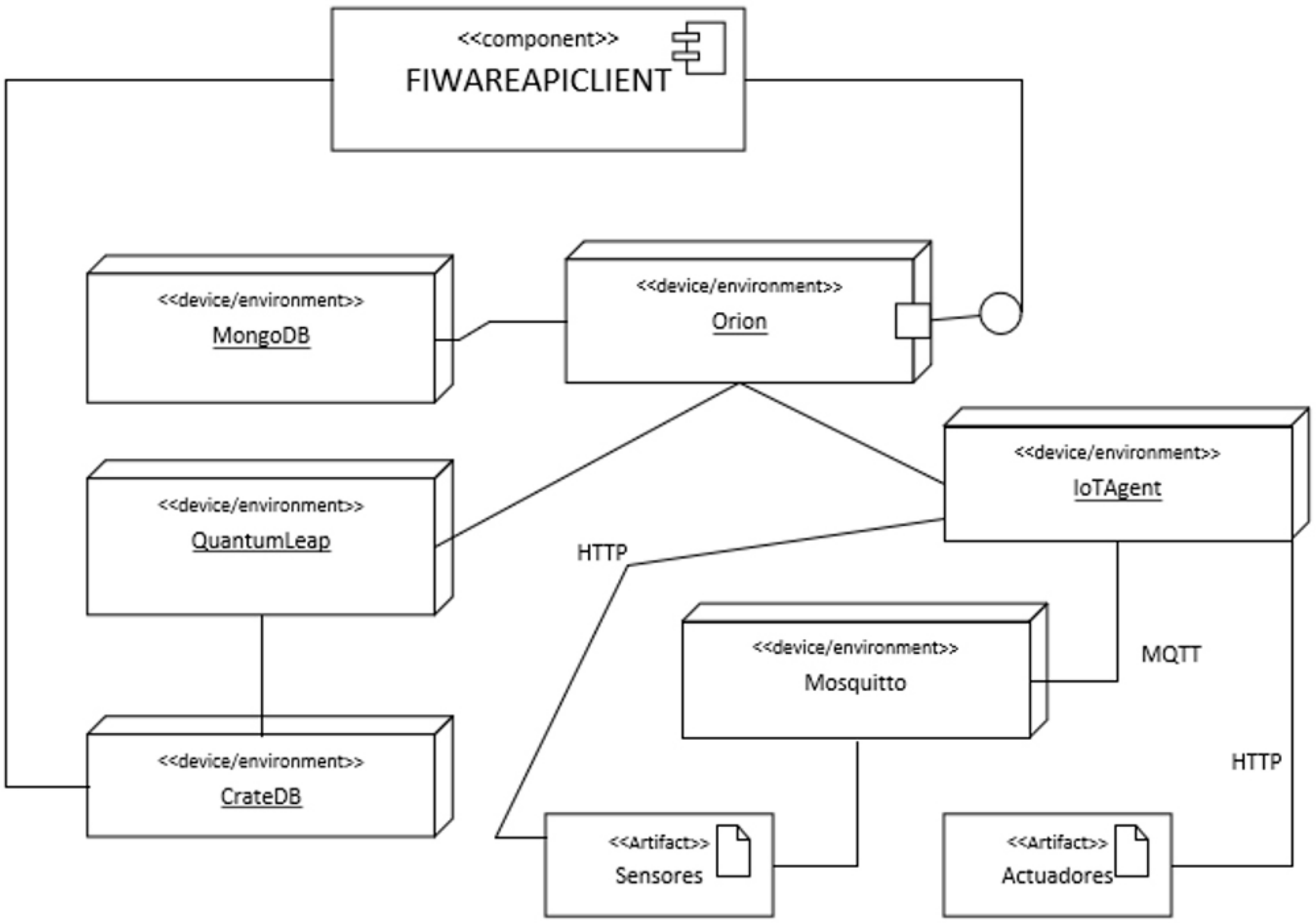


Figura 9: Modelo de Despliegue de la solución.

VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

La comprobación de la biblioteca de clases *Java* desarrollada consta de dos etapas, primero se diseña e implementa una aplicación web, en el entorno de desarrollo *Java* que usa la biblioteca de clases y posteriormente se selecciona un conjunto de datos de prueba para analizar los resultados que se obtienen a partir de las distintas consultas realizadas a la plataforma IoT FIWARE. Esta aplicación visualiza el formato de las peticiones y las respuestas (es decir, interacción con la plataforma FIWARE) que se realizan a la plataforma a través de la biblioteca desarrollada. En las operaciones realizadas se describe:

1. Cuál petición se debe realizar para cada operación.
2. Qué tipo de operación se espera.
3. Los encabezados necesarios.
4. El formato JSON enviado y recibido.

La aplicación de prueba desarrollada también muestra un ejemplo de cómo se podría estar visualizando en un mapa la información que se almacena en *Orion* en tiempo real y que cuenta con una componente espacial, así como la información almacenada en una base de datos externa con los históricos de mediciones anteriores.

Por otro lado, con el objetivo de comprobar el correcto funcionamiento de la biblioteca de funcionalidades creada también se desarrollaron un set de pruebas unitarias en *Java* usando *JUnit,* mediante la cual se pudo asegurar que cada unidad funciona correctamente y eficientemente por separado.

Diseño e implementación de las pruebas

Aplicación Web Cliente

Para mostrar el uso y la utilidad de la propuesta descrita se desarrolló de una aplicación web que muestra mediante gráficas y tablas los datos de contexto que se gestionan en la plataforma (Figura 10).

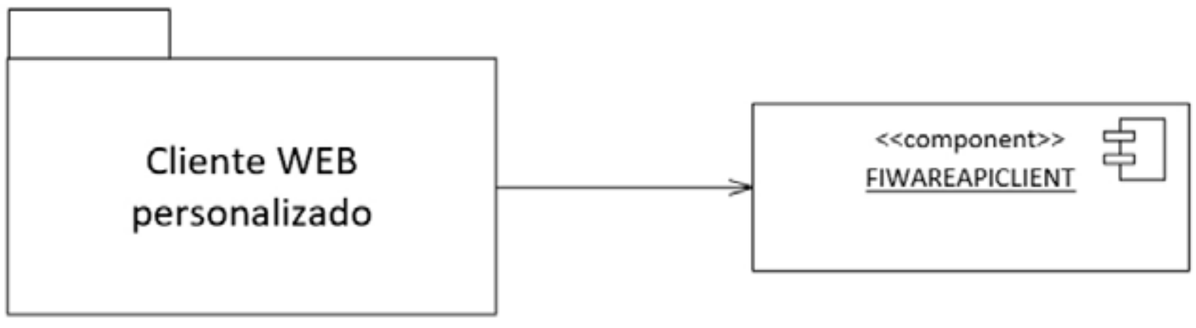


Figura 10:
Aplicación web cliente.

Esta aplicación estará formada por componentes que, gracias a la estructura establecida y al componente creado, serán capaces de mostrar en todo momento los datos reales de los dispositivos, así como permitir visualizar los datos históricos almacenados en una base de datos externa al contexto.

Esta aplicación permite conocer cómo es la interacción con la plataforma ya que en todo momento se tiene la posibilidad de visualizar las peticiones que se realizan a la plataforma, además que permite configurar toda la información almacenada en esta.

Uso del componente desarrollado

Para poder hacer uso del componente desarrollado luego de creada la aplicación cliente se adiciona el fichero ".jar" que contiene la solución tanto en el proyecto, como en el servidor de aplicaciones utilizado (véase Figura 11 A y B respectivamente). De esta manera se compilará el proyecto con el componente desarrollado y permitirá el uso de todas las funcionalidades que contiene, mediante las cuales se puede gestionar la información que se almacena en la plataforma, así como acceder a una base de datos externa donde se almacena la información histórica del contexto

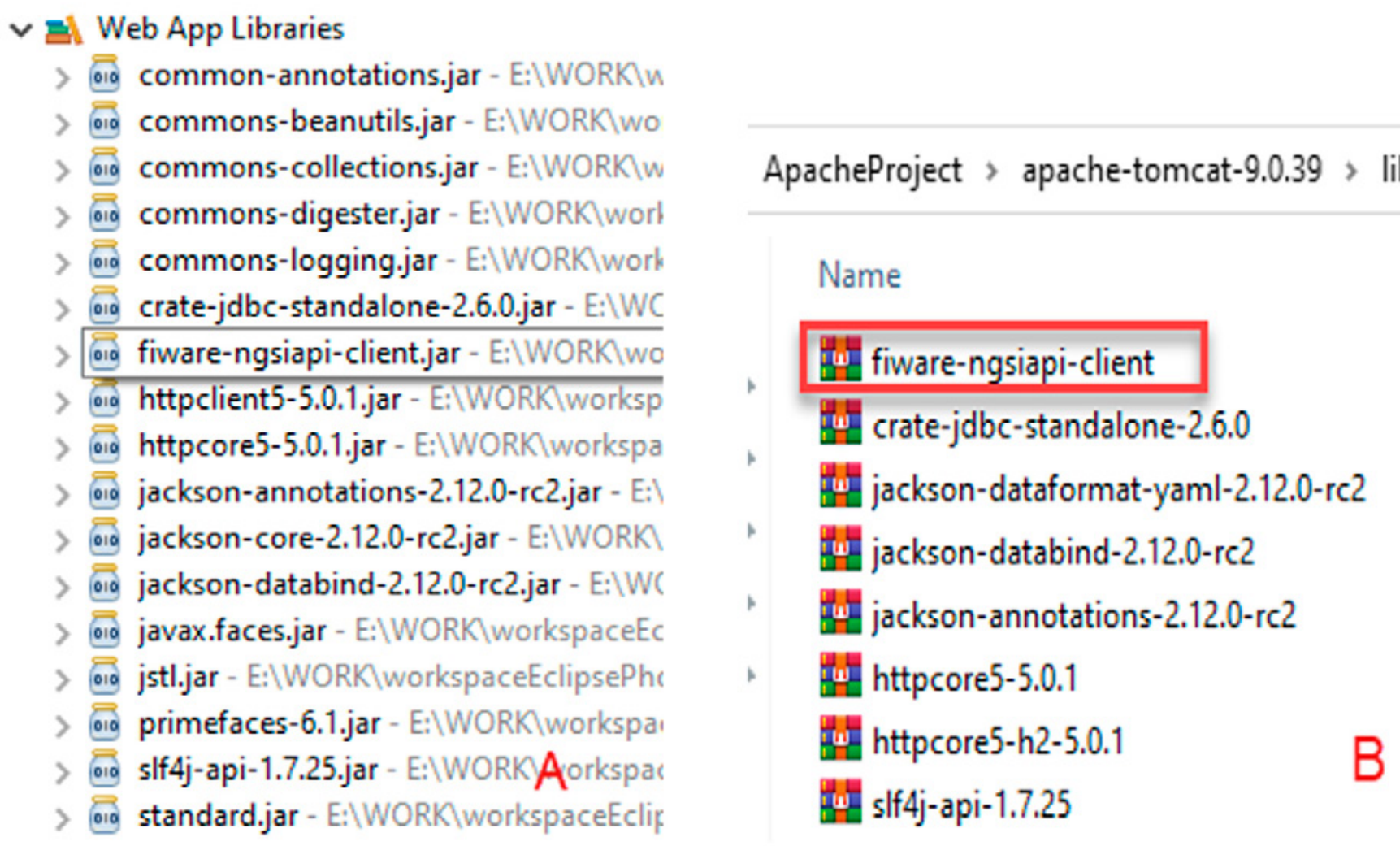


Figura 11: Vista que evidencia el uso del componente desarrollado.

Aplicación de pruebas unitarias

Para realizar pruebas unitarias al componente desarrollado y demostrar su correcto funcionamiento se ha realizado el desarrollo de un proyecto en el cual se implementaron un set de pruebas a cada una de las funcionalidades de la solución propuesta en este trabajo (Figura 12).

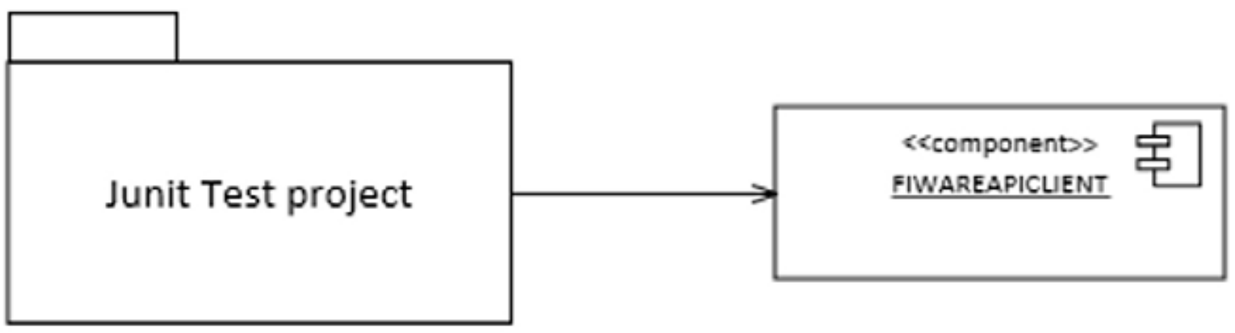


Figura 12:
Aplicación de pruebas unitarias.

Pruebas configuradas al componente desarrollado

Para poder realizar las pruebas del componente desarrollado luego de creada la aplicación de pruebas se adiciona el fichero ".jar" que contiene la solución como una referencia a una biblioteca de clases (véase Figura 13). De esta manera se compilará el proyecto con el componente desarrollado y permitirá la realización de cada una de las pruebas a las funcionalidades que contiene, mediante las cuales se puede validar el correcto funcionamiento de cada funcionalidad independiente.

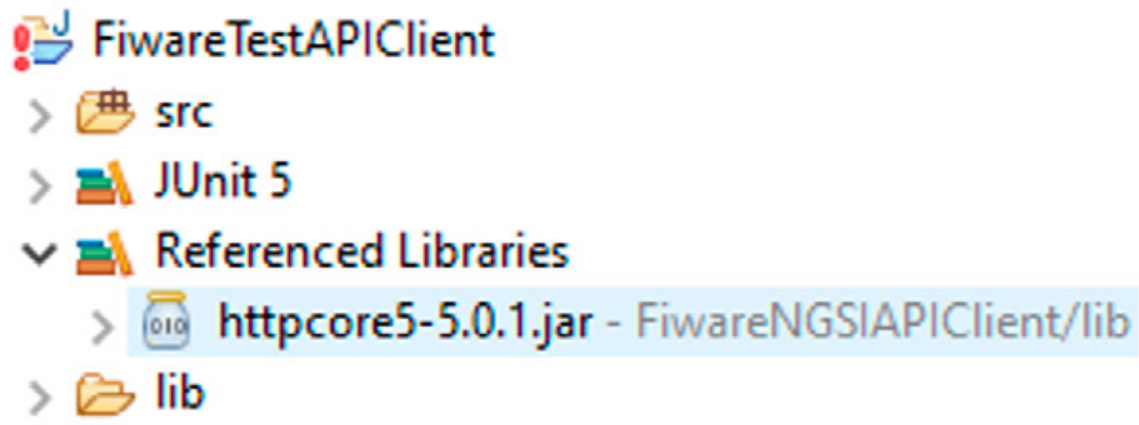


Figura 13:
Vista de implementación que evidencia las pruebas del componente desarrollado.

Evaluación del uso de la solución propuesta

Después de realizar pruebas a la biblioteca a continuación se muestran algunas de las funcionalidades evaluadas.

Funcionalidades evaluadas

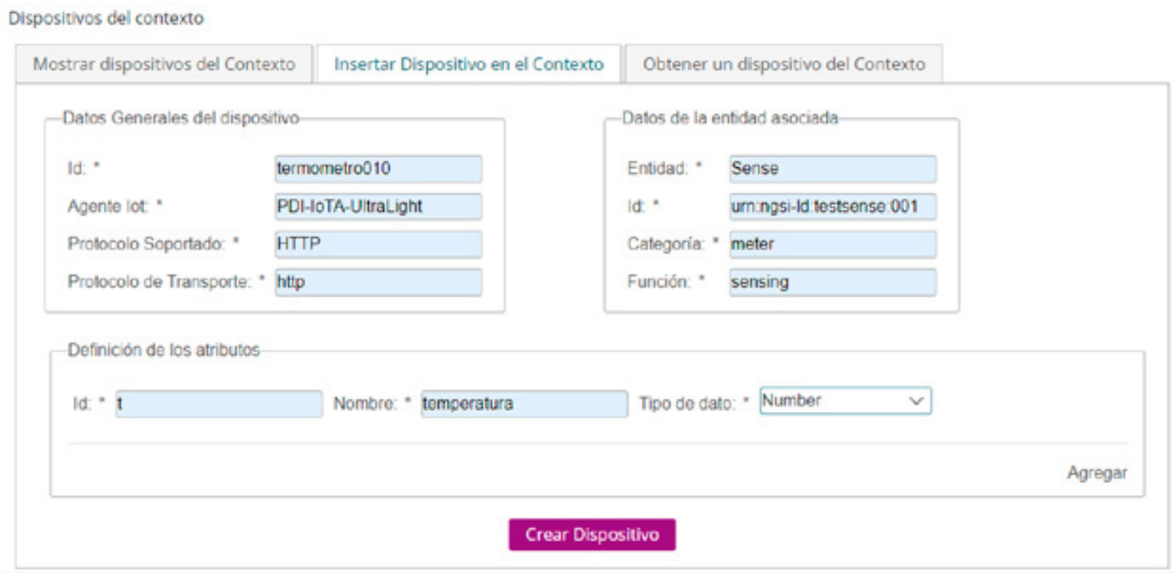
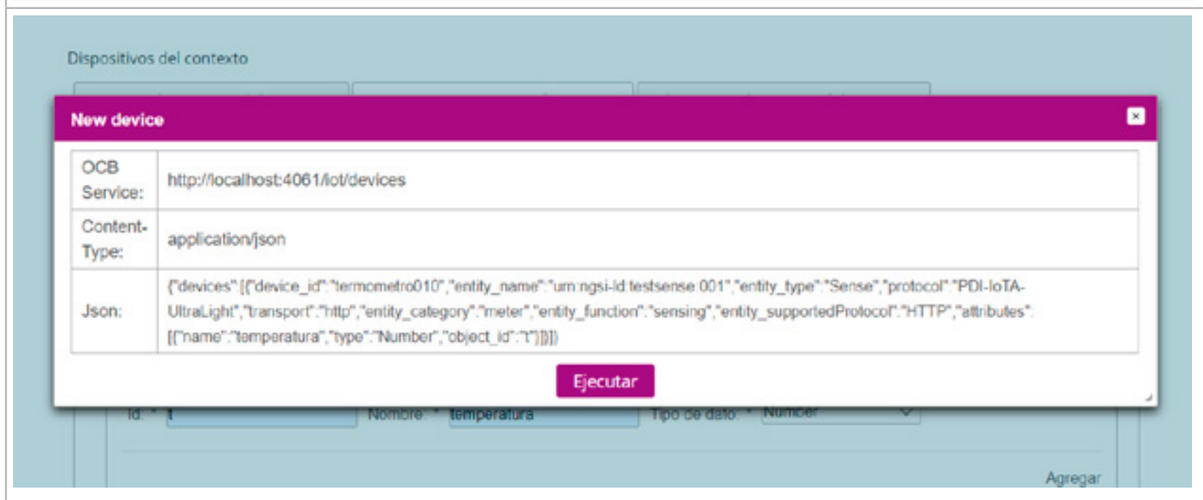
Todas las funcionalidades que se describen a continuación son peticiones realizadas al API REST de FIWARE. Estas peticiones requieren de uso de dos encabezados *FIWARE-service* y *FIWARE-servicepath,* se detallan a continuación.

- *FIWARE-service:* se define para que las entidades puedan ser guardadas en una base de datos separada de la base de datos por defecto de *MongoDB*.
- *FIWARE-servicepath:* marca una diferencia entre arreglos de dispositivos.

En una solución para una ciudad inteligente, por ejemplo, se esperaría definir diferentes *FIWARE-service* para diferentes departamentos: parques, transporte, etc. y el encabezado *FIWARE-servicepath* se referiría a un parque en específico.

A modo de ejemplo, en las tablas (de la 1 a la 6) se muestran las características de algunas funcionalidades implementadas en la solución propuesta.

Tabla 1: Funcionalidad para la creación del dispositivo IoT

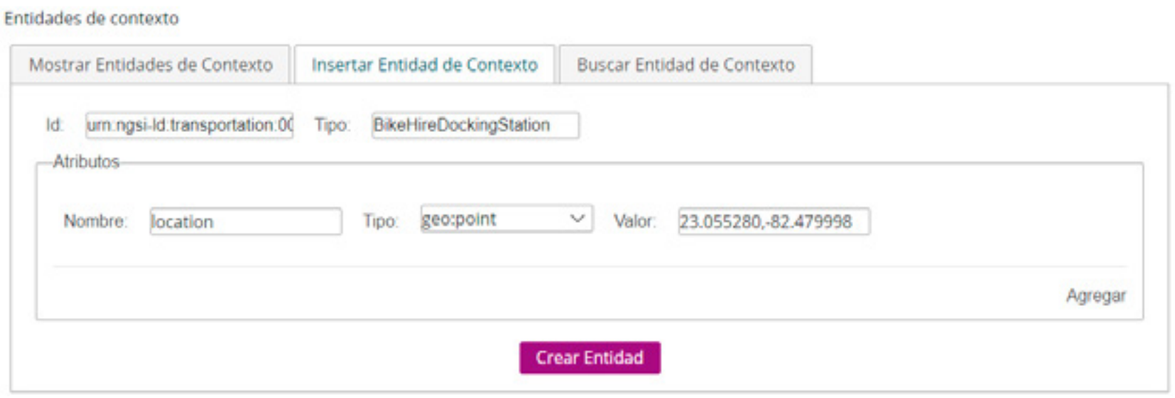
Nombre	**Crear dispositivo IoT**
Descripción	Esta funcionalidad de la solución permite crear un nuevo dispositivo IoT (sensor / actuador) en la plataforma IoT y establecer su configuración inicial.  
Utilizando directamente el API Rest de Fiware	
Header	FIWARE-service:iot, FIWARE-servicepath:/,content-Type: application/json
URL	http://[iotagenturl]/iot/devices Ex.http://localhost:4061/iot/devices
Atributos	device_id: Identificador del dispositivo entity_name: Identificador de la entidad a la cual le van hacer llegar algún valor de medición obtenido. entity_type: El tipo de la entidad a la cual le van hacer llegar algún valor de medición obtenido. protocolo: Protocolo de comunicación del agente IoT. Ex PDI-IoTA-UltraLight transport: protocolo de transporte del valor obtenido. Ex MQTT entity_category: Categoría del sensor o dispositivo. entity_function: Función del sensor o dispositivo. entity_supportedProtocolo: Protocolo de transporte soportado por la entidad. attributes: Arreglo de los atributos de la entidad que van a recibir los valores de medición enviados por el dispositivo. “attributes”: [{ “object_id”: “[letra q representa el parámetro del valor]”, “name”: “[entitytAttr]”, “type”: “[entityAttrData Type]” }] Ex. “attributes”: [{ “object_id”: “t”, “name”: temperature”, “type”: “Number” }]
Body	JSON: { “devices”:[{ “device_id”:”termometro003”, “entity_name”: “urn:ngsi-ld:testsense:001”, “entity_type”: “Sense”, “protocol”: “PDI-IoTA-UltraLight”, “transport”: “MQTT”, “entity_category”: “meter”, “entity_function”:”sensing”, “entity_supportedProtocol”:”MQTT”, “attributes”: [{ “object_id”: “t”, “name”: “temperature”, “type”: “Number” }]
Method	POST

Utilizando a través de la biblioteca de clases el API Rest de Fiware	
Clase	fiware.ngsi.v2.api.operations. NGSIDevicesManagement
Function	createDevice(String server, Map<String, String> headers, Sensor device, boolean actuator)
Code example:	NGSIResponseMessage response = NGSIDevicesManagement.createDevice(this.urlAgentIoT, getHeaders(), this.device, false);

Tabla 2: Funcionalidad para obtener dispositivo IoT por ID

Nombre	**Obtener dispositivo IoT por ID**
Descripción	Esta funcionalidad de la solución permite obtener un dispositivo IoT (sensor / actuador) creado en la plataforma IoT, dado su ID. 
Utilizando directamente el API Rest de Fiware	
Header	FIWARE-service:iot, FIWARE-servicepath:/
URL	http://localhost:4061/iot/devices/[deviceID]
Method	GET
Utilizando a través de la biblioteca de clases el API Rest de Fiware	
Clase	fiware.ngsi.v2.api.operations. NGSIDevicesManagement
Function	Sensor getDeviceById(String server, String id, Map<String,String> headers)
Code example:	Sensor foundDevice = NGSIDevicesManagement.getDeviceById(this.urlAgentIoT, this.deviceId, getHeaders());

Tabla 3: Funcionalidad para crear entidad de contexto

Nombre	**Crear entidad de contexto**
Descripción	Esta funcionalidad de la solución permite crear una entidad de contexto en el Orion Context Broker de la plataforma.  
Utilizando directamente el API Rest de Fiware	
Header	FIWARE-service:iot, FIWARE-servicepath:/, content-Type:application/json
URL	localhost:1026/v2/entities
Atributos	ID: id de la entidad Tipo: tipo de la entidad. Atributos: atributos de la entidad y su tipo (es decir si es numérico, o geográfico) Fecha: fecha de la creación o modificación de la entidad.

Body	{
	"id": "[entityid]",
	"type": "[entityType]",
	"[entityAttr]":{ "value":"[EntityAttrValue]", "type":"[EntityAttrDateType]" }}
	Ex. {
	"id": "urn:ngsi-ld:transportation:001",
	"type": "BikeHireDockingStation",
	"name":{ "value":"Transportation", "type":"Text" }}

Method	POST
Utilizando a través de la biblioteca de clases el API Rest de Fiware	
Clase	fiware.ngsi.v2.api.operations. NGSIEntityManagement
Function	NGSIResponseMessage createEntity(String server, Entity entity,Map<String, String> headers)
Code example:	NGSIResponseMessage response = NGSIEntityManagement.createEntity(url,newEntity, headersWithJsonContent)

Tabla 4: Funcionalidad para obtener entidad de contexto por ID

Nombre	**Obtener entidad de contexto por ID**
Descripción	Esta funcionalidad de la solución permite obtener los datos de una entidad de contexto dado su ID.

Utilizando directamente el API Rest de Fiware	
URL	localhost:1026/v2/entities/[entityid] Ex. localhost:1026/v2/entities/Room1
Header	FIWARE-service:iot, FIWARE-servicepath:/
Method	GET
Utilizando a través de la biblioteca de clases el API Rest de Fiware	
Clase	fiware.ngsi.v2.api.operations. NGSIEntityManagement
Function	Entity getEntityById(String server, String id, Map<String, String> headers)
Code example:	Entity entity = getEntityById(server, entityId,headerMap);

Estas y otras funcionalidades no mostradas en el ejemplo conforman el paquete completo de funcionalidades de la biblioteca API desarrollada. La API se encuentra en fase de registro de acuerdo al procedimiento establecido por el Ministerio de Comunicaciones (Mincom) para hacerla disponible a los desarrolladores como parte del despliegue de la solución.

AGRADECIMIENTOS

Este artículo forma parte de los trabajos desarrollados por el grupo de desarrollo de servicios y aplicaciones informáticas T&C de la Facultad de Ingeniería Informática de la Universidad Tecnológica de La Habana, como parte del *Proyecto Ciudad* como plataforma inteligente y colaborativa, coordinado por la Unión de Informáticos de Cuba, en el marco del Programa Nacional de Ciencia, Tecnología e Innovación (2021-2023).

CONCLUSIONES

La biblioteca de clases desarrollada permite crear y gestionar una solución inteligente, a partir de la comunicación con la plataforma FIWARE, así como, obtener los datos históricos almacenados en una base de datos externa. Utiliza las capacidades de FIWARE para desarrollar una aplicación web que permite gestionar entidades de contexto, sensores, actuadores y suscripciones.

La aplicación desarrollada permite unir componentes de propósito general de la plataforma FIWARE, con aplicaciones clientes y con dispositivos para la captura de información de contexto. Además de agrupar las principales operaciones que se realizan sobre el contexto, es posible obtener la información histórica del contexto almacenada en una base de datos externa a la plataforma.

Con vistas a su explotación, la biblioteca está siendo desplegada en la nube de la Unión de Informáticos de Cuba, junto con una aplicación demostrativa para guiar a los desarrolladores. El equipo continúa trabajando en la mejora continua de ese servicio, como parte del Proyecto titulado “Marco gobernable de ciudades inteligentes y colaborativas como plataformas” del Programa Nacional de Ciencia, Tecnología e Innovación “Telecomunicaciones e informatización”.

REFERENCIAS

Crate.io. (2021). A Single Data Hub for(FIWARE-Foundation, 2021) all Operational Data. from https://crate.io/

Cubillas-Hernández, E., Anías-Calderón, C., & Delgado-Fernández, T. (2021). Arquitectura M2M para el monitoreo ambiental en tiempo real. *ITECKNE: Innovación e Investigación en Ingeniería, 18*(1), 2-2.

Firouzi, F., Farahani, B., Weinberger, M., DePace, G., & Aliee, F. S. (2020). Iot fundamentals: Definitions, architectures, challenges, and promises *Intelligent Internet of Things* (pp. 3-50): Springer.

FIWARE-Foundation. (2021a). FIWARE COMPONENTS. from FIWARE.org/developers/catalogue/

FIWARE-Foundation. (2021b). FIWARE: The Open Source Platform for Our Smart Digital Future. from https://www.FIWARE.org/

FIWARE.org. (2021a). FIWARE IoT-Agent-UL. from https://FIWARE-iotagent-ul.readthedocs.io/en/latest/usermanual/index.html

FIWARE.org. (2021b). FIWARE QuantumLeap. from https://quantumleap.readthedocs.io/en/latest/

FIWAREmexico.org. (2021). APRENDE FIWARE EN ESPAÑOL. from https://FIWARE-training.readthedocs.io/es_MX/latest/

Guth, J., Breitenbücher, U., Falkenthal, M., Fremantle, P., Kopp, O., Leymann, F., & Reinfurt, L. (2018). A detailed analysis of IoT platform architectures: concepts, similarities, and differences *Internet of everything* (pp. 81-101): Springer.

Miorandi, D., Sicari, S., De Pellegrini, F., & Chlamtac, I. (2012). Ad hoc networks internet of ings: vision, applications and research. *Ad Hoc Networks, 10*(7), 1497-1516.

Ochoa Duarte, A., Cangrejo Aljure, L. D., & Delgado, T. (2018). Alternativa Open Source en la implementación de un sistema IoT para la medición de la calidad del aire. *Revista Cubana de Ciencias Informáticas, 12*(1), 189-204.

Qiu, D., Li, B., & Leung, H. (2016). Understanding the API usage in Java. *Information and software technology, 73*, 81-100.

Rodriguez, A. (2008). Restful web services: The basics. *IBM developerWorks,* 33, 18.

Sánchez, V. M. B. (2018). Internet de las cosas-horizonte 2050. bie3: Boletín IEEE(11), 956-969.

Zeinab, K. A. M., & Elmustafa, S. A. A. (2017). Internet of things applications, challenges and related future technologies. *World Scientific News, 2*(67), 126-148.

FIWARE-Foundation. (2021). FIWARE: The Open Source Platform for Our Smart Digital Future. Retrieved from https://www.fiware.org/

Copyright © 2022 Berbes-Villalon, D. M., Sanchez-Jimenez, L., Diaz-Aguirre, M. E., Delgado-Fernandez, T.

Este obra está bajo una licencia de Creative Commons Atribución-No Comercial 4.0 Internacional