FRAMEWORK DE INTEROPERABILIDAD SEMÁNTICA DE OBJETOS INTELIGENTES PARA LA WEB DE LAS COSAS

Santiago Guerrero Narváez

Tesis de Maestría en Ingeniería Telemática

Director: PhD. Gustavo Adolfo Ramírez González
Asesor: PhD. Miguel Ángel Niño Zambrano

Universidad del Cauca

Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Telemática
Línea de Investigación IoT
Popayán, Julio 2021

Santiago Guerrero Narváez

FRAMEWORK DE INTEROPERABILIDAD SEMÁNTICA DE OBJETOS INTELIGENTES PARA LA WEB DE LAS COSAS

Tesis presentada a la facultad de Ingeniería Electrónica y de Telecomunicaciones de la Universidad del Cauca para la obtención del título de

Magister en
Ingeniería Telemática

Director: PhD. Gustavo Adolfo Ramírez González
Asesor: PhD. Miguel Ángel Niño Zambrano

Popayán Julio 2021

NOTA DE ACEPTACIÓN

_____

_____

_____

_____

PRESIDENTE DEL JURADO

_____

JURADO

Popayán, Julio 2021

AGRADECIMIENTOS

Quiero agradecer muy especialmente a mis padres, sin su guía, apoyo y comprensión que siempre me han dado aún en momentos difíciles no hubiera logrado esto. También quiero dar un agradecimiento muy especial a la compañera de mi vida que es quien ha vivido todas estas aventuras a mi lado y que, sin juzgarme, siempre me apoyo, pero también me corrigió cuando yo estaba equivocado, sin su apoyo, consejos paciencia y especialmente compañía no lo hubiera logrado.

TABLA DE CONTENIDO

INDICE DE TABLAS

INDICE DE FIGURAS

1 INTRODUCCIÓN

1.1 CONTEXTO GENERAL

Primitivamente los sentidos humanos eran utilizados para interpretar de forma directa el estado de algún objeto[1] que nos interesara y las primeras herramientas nos permitieron fabricar los que necesitásemos. Después, la llegada de los primeros pasos en las ciencias nos reveló nuevos secretos sobre nuestro entorno y como utilizarlo para nuestro beneficio. Con el pasar del tiempo los objetos proliferaron en la vida cotidiana, y de mano a los avances tecnológicos y científicos, fuimos capaces de conocer más a fondo las variables de nuestro entorno. Ahora, nuestra vida cotidiana funciona alrededor de muchos y diversos objetos, qué gracias a la tecnología, pueden aumentar sus capacidades, haciendo que cada uno pueda comunicar sus estados y pueda modificar su entorno. Todos los objetos con los que interactuamos pueden ser dotados de herramientas software y hardware para generar información sobre su estado, su entorno y sus relaciones, esta información puede ser entendida por el hombre de diferentes formas dando a los simples datos una dimensión interpretativa y semántica, generando a este nivel la mayor riqueza en cuanto a los servicios que un objeto nos pueda brindar, por otra parte, si consideramos que una de las ramas del desarrollo humano ha sido el internet, y la flecha apunta a que todos debemos estar conectados, resulta natural pensar que los objetos también deben estar conectados, y no solo conectados con personas, sino conectados entre sí. A esta apuesta se la ha llamado internet de las cosas (IoT).

Los objetos de todo el mundo están dejando de tener un papel inerte para convertirse en parte fundamental y activa de la calidad de vida de las personas y de las organizaciones. Los objetos cotidianos están evolucionando y gracias al avance del campo de la electrónica adquieren nuevas capacidades, como: obtener datos, procesar y compartir información del medio que los rodea. Cuando conectamos los objetos cotidianos a internet, permitiendo crear sus representaciones digitales, podemos interactuar con ellos por medio de la web y las aplicaciones contenidas ahí, a esto se le ha llamado la web de las cosas (WoT).

Para lograr una interoperabilidad entre objetos es necesario aumentar las capacidades de los distintos elementos que la soportan. De esta forma, se hace necesario que se sume nuevas aristas tecnológicas, como herramientas semánticas, para lograr una verdadera inteligencia en la interacción entre objetos, procesamiento del lenguaje natural, para manejar un lenguaje común entre usuarios y máquinas y herramientas de recuperación de la información para ordenar y parametrizar los datos que generan los recursos de los objetos. Por otra parte, es necesario definir las características básicas que deben tener los objetos para poder interoperar utilizando dichas herramientas. Para esto se utilizan los objetos inteligentes (OI) [1], los cuales contienen las características necesarias para soportar la complejidad de estas herramientas.

[1] En el proyecto se utilizará el concepto de objeto como sinónimo de cosa o su traducción al inglés “thing”.

Cada OI puede ser considerado como un pequeño servidor web, exponiendo diferentes propiedades medibles del ambiente. Si distintos OI tienen la capacidad de conectarse entre sí, basados en un contexto determinado, es posible crear un entorno de interoperabilidad, en el cual, las funcionalidades combinadas de diversos OI pueden generar nuevos servicios a los actores. Adicionando herramientas semánticas a los procesos de interoperabilidad, podemos tener un ambiente que puede ofrecer información y servicios más valiosos en la web, denominando este ecosistema como un escenario de interoperabilidad semántica entre OI para la WoT [2].

Sin embargo, el desarrollo de las tecnologías que permiten la creación de estos escenarios, son de carácter complejo, requieren de un alto conocimiento técnico y es necesario una alta inversión de esfuerzo para poder ser implementadas; por lo tanto, se hacen necesario pensar en formas de facilitar la adopción y utilización de las herramientas para generar este tipo de escenarios y así lograr obtener sistemas y servicios que puedan beneficiar de mejores formas a los usuarios y a las personas en general.

1.2 DECLARACION DEL PROBLEMA

El desarrollo de la IoT inició de manera desorganizada, generando problemas de heterogeneidad y conectividad [3, 4], impidiendo que los objetos tengan una completa "consciencia"[2] de su ambiente, desplegando funcionalidades restringidas en la interacción con los usuarios y otros objetos. La heterogeneidad se manifiesta en la gran variedad de objetos conectados a internet, los cuales son desarrollados para diferentes propósitos, por ende, no todos tienen las mismas capacidades y no todos pueden compartir de la misma forma sus datos. Y los problemas de conectividad se manifiestan, aunque en menor medida, en la variedad de protocolos y elementos de comunicación que han sido personalizados para resolver requerimientos de aplicaciones y sistemas puntuales o a la medida.

Existen diversas propuestas, las cuales solucionan en parte la heterogeneidad y conectividad, dando al OI la capacidad de definir un perfil digital y exponiendo la información que genera por medio de una plataforma. Se han creado protocolos y estándares para regular tanto la forma como se fabrican los componentes hardware, necesarios para implementar los OI; como también, para estructurar los mensajes que se envían entre dos objetos [5]. En el mismo sentido, las herramientas, aplicaciones y sistemas que se han desarrollado sobre las bases de estos protocolos y estándares, han generado una gran variedad de arquitecturas, librerías, aplicaciones, entre otras, que son el soporte para generar escenarios de interoperabilidad semántica de OI [6, 7, 8, 9, 10, 11, 12, 13]

La gran variedad de herramientas y elementos en general, que permiten el desarrollo e implementación de estos ecosistemas es un gran beneficio para las personas y organizaciones; sin embargo, para lograr una solución exitosa de este tipo de entornos se requiere una gran inversión de recursos. Las herramientas que se involucran en estos

[2] Del inglés "awareness". Tiene relación con el nivel de conocimiento que tiene un elemento con su entorno.

desarrollos, tienen una naturaleza diversa, desde las librerías para obtener los datos de recursos de los OI, pasando por elementos para realizar consultas a ontologías, hasta los módulos necesarios para manejar protocolos IoT. Todos estos elementos, requieren de una curva de aprendizaje de un tiempo considerable. Los conocimientos necesarios son de carácter complejo, como el manejo de herramientas semánticas, las cuales requieren de una gran profundización en su entendimiento. Estas conclusiones en cuanto al esfuerzo en el desarrollo de dichos entornos son rescatadas de las experiencias del trabajo [14]

A futuro se espera que los OI sean completamente interoperables entre sí, la generación de nuevos servicios se ajuste a los contextos de cada usuario y las interfaces tengan gran nivel de usabilidad, pero esto no se logrará si no se dotan a los desarrolladores de los medios para implementar este tipo de entornos de interoperabilidad semántica entre OI.

Por lo anterior, el desarrollo de entornos de interoperabilidad semántica, requieren de un alto conocimiento técnico y una alta inversión de esfuerzo (tiempo) para ser implementados. Así, surge la pregunta de investigación: ***¿Cómo reducir el esfuerzo de implementación de entornos de interoperabilidad semántica de objetos inteligentes de la WoT?*** Para la presente investigación, el enfoque de solución es la construcción de un Framework[3] de interoperabilidad semántica de OI para la WoT, utilizando el ecosistema IoT actualmente instalado y aplicando técnicas que se usan en la web semántica.

1.2.1 Hipótesis

HI: El Framework de interoperabilidad semántica de OI para la WoT, reduce el esfuerzo para la implementación de entornos de interoperabilidad semántica de OI.

1.3 ESCENARIO DE MOTIVACIÓN

Al estar inmersos en un mundo cambiante en que los objetos han tomado posesión de las rutinas diarias hasta el punto que definen, relacionan e interactúan entre sí y con las personas; es imposible no centrar la atención en los diferentes objetos que se han convertido en parte activa para mejorar la calidad de vida, controlar espacios y, entre otras cosas, aumentar el bienestar individual y grupal.

Al ir en aumento el número de objetos conectados, se ve la necesidad de ahondar en problemas específicos que contribuyan a la creación de nuevos escenarios de interoperabilidad que entreguen a los usuarios y personas, información y aplicaciones cada vez más valiosos. De acuerdo a esto, los objetos y todos los elementos que entran en juego al momento de implementar un escenario de interoperabilidad, deben evolucionar y conectarse (o ser remplazados) con nuevas herramientas de vanguardia. Sin embargo, la aceleración de la complejidad no debe afectar (o afectar en lo más mínimo posible) la

[3] Framework no es posible traducirlo con una palabra que encierre todo el sentido que el presente proyecto busca darle, este se considera como la combinación de un marco conceptual y librerías software.

factibilidad de implementación de estos escenarios, de lo contrario la población de personas y organizaciones capaces de desplegar este tipo de ecosistemas se vería reducida, haciendo que estos nuevos pasos al futuro sean contraproducentes.

En este sentido la motivación de este trabajo se centra en la construcción de un Framework de desarrollo, el cual entrega a los usuarios desarrolladores de escenarios de interoperabilidad semántica, una capa de creación de OI la cual pretende reducir la complejidad de implementar este tipo de sistemas. Para lograr definir, construir y probar el Framework se propone seguir un objetivo general y sus respectivos objetivos específicos presentados a continuación.

1.3.1 Objetivo general

Proponer un framework de interoperabilidad semántica de OI para la WoT, que permita la reducción del esfuerzo en la implementación de entornos de interoperabilidad semántica de OI, utilizando recursos IoT existentes y técnicas de la web semántica.

1.3.2 Objetivos específicos

- Crear un marco conceptual que identifique los elementos más importantes para la implementación de entornos de interoperabilidad semántica de OI para la WoT.

- Implementar un conjunto de librerías en un lenguaje de programación especifico y basado en el marco conceptual, que apoyen el desarrollo de entornos de interoperabilidad semántica de OI para la WoT.

- Realizar una prueba de concepto para el caso de estudio en domótica, mediante la implementación de un entorno IoT, con al menos tres OI utilizando el Framework propuesto.

- Evaluar el esfuerzo invertido para la implementación de entornos de interoperabilidad semántica de OI para WoT, mediante la comparación del desarrollo utilizando el Framework y sin él.

1.4 CONTRIBUCIONES

1.4.1 Nivel investigativo

- Fortalecimiento teórico en las áreas del internet de las cosas, la web semántica, objetos inteligentes. A partir del estudio y el análisis de diferentes investigaciones a través de revisión bibliográfica y análisis bibliométricos y en adición a la presente propuesta; se fortalece la base teórica perteneciente a la institución académica en lo referente al internet de las cosas, los objetos inteligentes, la interoperabilidad y la web semántica.

Esto permite establecer un soporte para futuras investigaciones en dichas áreas. El producto de este punto consta de un artículo publicado sobre el análisis bibliométrico.

- Marco conceptual de interoperabilidad semántica de objetos inteligentes en la web de las cosas. Utilizando el panorama recogido en el estudio de los principales campos: internet de las cosas, web semántica y objetos inteligentes, se construye un marco conceptual, el cual soporta la construcción del Framework, en cuanto a la selección de herramientas, protocolos, y modelos. El producto de este punto consta de una definición a nivel conceptual del Framework.

- Monografía del Trabajo de Grado. Corresponde al presente documento, donde se describe el proceso utilizado en el desarrollo del proyecto, los problemas que se presentaron, las respectivas soluciones, los principales aportes, las conclusiones y recomendaciones para el desarrollo de futuras investigaciones. Este documento se divide en 8 capítulos diferentes, dentro de los cuales el primero a la introducción y una contextualización especial del proyecto, el segundo capítulo define el marco teórico que reúne conceptos que se utilizaron para definir el trabajo, el tercer capítulo contiene la arquitectura del escenario, donde se encontró los distintos elementos, herramientas, protocolos, etc, que soporta el escenario; el cuarto capítulo trata la construcción e implementación del escenario, en el quinto capítulo se puede encontrar las distintas pruebas realizadas al escenario, el sexto capítulo explica cómo se cumplieron los objetivos del trabajo propuesto, el séptimo capítulo muestra las conclusiones y los trabajos futuros y el octavo capítulo lista los referentes bibliográficos. El producto de este punto es el actual documento.

1.4.2 Nivel técnico

- Framework para la construcción de OI. A partir del marco conceptual es entregado un Framework que comprende una serie de librerías y utilidades en Python como lenguaje de programación, estas herramientas facilitan la construcción de objetos inteligentes, los cuales contienen la capacidad de interoperar a nivel semántico y crear nuevos servicios con los recursos de otros objetos inteligentes dentro de un escenario establecido. El producto de este punto es un repositorio de código abierto donde se encuentra alojado el Framework.

- Escenario de interoperabilidad semántica de objetos inteligentes en la web de las cosas. Utilizando aplicaciones de dispositivos embebidos para la gestión de los objetos inteligentes, se desarrolla la primera versión de un escenario de interoperabilidad semántica de objetos inteligentes para la web de las cosas construido a partir del Framework propuesto en el presente trabajo. El producto de este punto es un artículo publicado sobre la implementación de objetos inteligentes.

- Plantilla experimental para evaluar el esfuerzo de un Framework de interoperabilidad semántica de objetos inteligentes para la web de las cosas. Para la evaluación del

esfuerzo del Framework fue necesario desarrollar una plantilla en la cual se considera los pasos necesarios para lograr una correcta evaluación de este tipo de tecnologías, considerando los diferentes requerimientos y limitaciones que puede tener este tipo de experimentos. El producto de este punto es un documento con los detalles de cada paso seguido en el experimento.

2 MARCO TEORICO

En este capítulo se puede encontrar los principales conceptos teóricos sobre los cuales se basa el presente trabajo, además se presentan algunas comparaciones entre diferentes tecnologías. Los principales temas giran alrededor de IoT, WoT y web semántica, los cuales constituyen el núcleo conceptual principal necesario para el desarrollo del Framework.

Para mayor detalle sobre el marco teórico y los conceptos que no se presentan en los siguientes apartados consultar el Anexo B.

2.1 IoT

Se define como una red de redes, es decir, una red que interconecta redes de computadoras con el fin de compartir recursos [15]. Acorde a lo anterior IoT se puede describir como la conexión inteligente de los objetos cotidianos como teléfonos inteligentes, televisores, sensores y actuadores a internet, permitiendo nuevas formas de comunicación entre los objetos y las personas [16]. Es importante entender la arquitectura de la IoT para poder aprovecharla adecuadamente, además identificar las capas y sus funciones con el fin de diferenciar los posibles problemas y las herramientas que pueden ser aplicadas para resolverlos.

2.1.1 Arquitectura IoT

Hay muchas propuestas arquitectónicas de la IoT, pero todas en general desembocan en proponer un enfoque por capas, por lo cual se presenta la Figura 1 la cual generaliza de una mejor forma las arquitecturas IoT. Esta propuesta tiene dos divisiones distintas con una capa intermedia que comunica a las otras. Las dos capas inferiores contribuyen a la captura de datos, mientras que las dos capas en la parte superior son responsables de utilización de los datos en las aplicaciones [17].

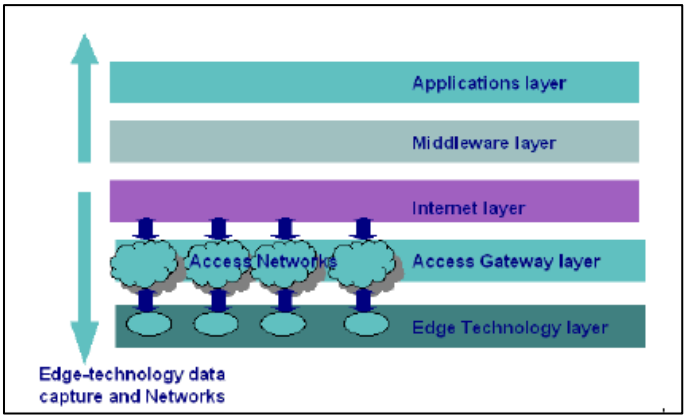


Figura 1. Arquitectura en capas de Internet de las Cosas. Fuente: [17]

- Capa hardware. Constituida por redes de sensores, sistemas embebidos, etiquetas identificación por radio frecuencia "radio frequency identification" - RFID. Estas entidades son los sensores de datos primarios desplegados en el mundo físico

- Capa de acceso. Es la primera etapa del tratamiento de datos. Se ocupa de enrutamiento de mensajes, publicación y suscripción y también lleva a cabo la comunicación entre plataformas

- Capa de aplicación. Entrega las aplicaciones a diferentes usuarios en la IoT.

- Capa de middleware. Opera en modo bidireccional. Actúa como una interfaz entre la capa de hardware y la capa de aplicación. Es responsable de las funciones críticas tales como administración de dispositivos y gestión de la información y también se ocupa de cuestiones como el filtrado de datos, la agregación de datos, el análisis semántico, control de acceso, el descubrimiento de información, como el servicio de información a través del código electrónico de producto y el servicio de nombre de objetos.

2.1.2 Middleware

El middleware [18] se soporta sobre una plataforma software que permite integrar los objetos inteligentes en la red de información. Su principal objetivo consiste en abordar la interoperabilidad entre dispositivos heterogéneos que se encuentran en diversos dominios de aplicaciones. Otro objetivo del middleware, con respecto a los objetos inteligentes, son: adaptación, descubrimiento, administración, escalabilidad, gestión de grandes cantidades de datos y aspectos de seguridad.

- Interoperabilidad. Comparte información y utiliza la misma a través de diversos dominios de las aplicaciones que utilizan diversas interfaces de comunicación. Existen tres categorías. Primero, red; define los protocolos para el intercambio de información entre los diversos objetos a través de diferentes redes de comunicación, sin tener en cuenta el contenido de la información. Segundo, semántica; define los protocolos para el intercambio de información entre las diversas cosas a través de diferentes redes de

comunicación, sin tener en cuenta el contenido de la información. Y tercero, sintácticas; hace referencia al formato y la estructura de la codificación de la información intercambiada entre los objetos.

- Detección de contextos. Recoge datos e identifica los factores que tienen un impacto significativo en la respuesta, además de esto es preciso procesar el contexto y realiza la toma de decisiones en base a ello.

- Detección del dispositivo. Permite a cualquier dispositivo de la red, detectar todos sus dispositivos vecinos y hacer notar su presencia a cada vecino en la red.

- Seguridad y privacidad. Son responsables de la confidencialidad y autenticidad.

- Administración de volúmenes de datos. Los datos pueden ser: identificación, posición, ambientales, históricos y descriptivos.

La Tabla 1 es una tabla comparativa de las principales opciones de middleware enfocados en la IoT

Tabla comparativa de herramientas middleware para IoT

Nombre	Tipo Servidor	Protocolo web*	API	Soporte	
				Comunidad	Tutorial
Oracle	Nube	R	Java	Bueno	Bueno
Swarm	Nube	R	Python	Malo	Bueno
Axeda	Nube	R, C, M	Java, C	Malo	Bueno
OpenRemote	Nube	R	Java	Bueno	Bueno
Etherios	Nube	H	Android, Java, Python	Bueno	Medio
ioBridge	Nube	R	Arduino	Bueno	Bueno
Zatar	Nube	R, C	Java	Medio	Medio
Ayla	Nube	R	Base Linux	Malo	Malo
Echelon	Nube	R	C, C++ , Python	Bueno	Bueno
EVERYTHING	Nube	R, C, M	Java, Jacascript	Malo	Medio
Exosite	Nube	H, C	Arduino, Python, C/C++, Java	Medio	Bueno
Xively	Nube	R, M	Arduino, Python, C, Java	Bueno	Bueno
Carriots	Nube	R, M	Arduino, Python	Bueno	Bueno
GroveStreams	Nube	R	Arduino, Python, Java	Bueno	Bueno
Lab of Thing	Local	R	Java, Arduino	Bueno	Bueno
Paraimpu	Nube	R	Arduino	Medio	Medio
SensorCloud	Nube	R	Python, Java, C#	Malo	Medio
ThingSepak	Nube	R	Arduino, C, Python	Bueno	Bueno
XOBXOB	Nube	R	Arduino	Medio	Medio

Lab of Thing	Local	NO	Arduino	Bueno	Bueno
2Lemetry	Nube	M	Arduino, Python, C	Medio	Medio
M2MLabs	Local	R	Arduino, Python	Malo	Bueno
OpenHAB	Local	H, M	No especifica	Medio	Medio
Node-RED	Local	H, M	Node.JS, Arduino	Bueno	Bueno
The Thing System	Local	Propio	Arduino	Bueno	Medio
DeviceHive	Local	R	C++, Python, C, Arduino	Malo	Medio
Mosquitto	Broker	M	Python, Java	Meio	Medio
Paho	Broker	M	Python, C++, Java, JavaScript	Bueno	Bueno
ThingMQ	Broker	R, C, M	Python, C++, Java, JavaScript	Medio	Medio

**R: REST, C:CoAP, M:MQTT, H:HTTP*

Tabla 1. Características herramientas middleware IoT. Fuente: Propia

La Tabla 1 compara diferentes herramientas middleware ofrecidas en la web, encontradas en distintos foros y sitios web que comparten información sobre temas de IoT. Las herramientas que se presentan son de acceso gratuito, algunas cuentan con un modo de prueba limitado. Se relaciona la herramienta con el tipo de servidor en la cual está soportado, encontrando tres principales: servidor alojado en la nube (nube), servidor que debe alojarse de forma local (local) y servidor de tipo Broker. Se relaciona la herramienta con los tipos de protocolos que soporta, diferenciando los siguientes: Rest (R), CoAP (C), MQTT (M), HTTP (H). Se relaciona la herramienta con las API's y librerías ofrecidas por la plataforma o por la comunidad en distintos lenguajes de programación. Y, por último, se evalúa los soportes ofrecidos por la comunidad y por la plataforma (tutoriales) en tres niveles, bueno, medio y malo.

2.1.3 Dispositivos IoT

Para el presente proyecto se hace necesario seleccionar una herramienta que permita manipular de una forma profunda el comportamiento del objeto inteligente, por eso quedan descartados objetos de código fuente cerrado o que no provean API´s u otras herramientas que permitan su desarrollo. Se opta por herramientas de hardware libre, específicamente placas de desarrollo. Existen muchas opciones en el mercado de placas de desarrollo que presentan diferentes capacidades, entre ellas podemos destacar las siguientes:

- Arduino [19], es una placa de desarrollo electrónica de código abierto basada en hardware y software fácil de usar. Está pensada para que la pueda utilizar cualquier persona que haga proyectos interactivos

- Intel Galileo [20], un ordenador de placa reducida de bajo coste desarrollado por Intel con el objetivo de enfocar el producto en desarrollos IoT

- BeagleBone [21] es una placa de hardware libre de bajo consumo producida por Texas Instruments en asociación con DigiKey y Newark. Fue diseñado con el desarrollo de software de código abierto.

Después de comparar las distintas posibilidades en cuestión de placas de desarrollo se opta por seleccionar la placa Intel Galileo, principalmente porque la universidad del Cauca tiene la posibilidad de proporcional estas placas para el desarrollo del presente proyecto.

- Raspberrypi [22], tarjeta de desarrollo basado en la arquitectura de microprocesadores, lanzada por la empresa Raspberry Pi Fundation en el año 2012 con fines educativos. Cuenta con un gran procesamiento de bajo consumo y coste soportado en diversos procesadores según la versión. Está diseñado para ser compatible, con sistemas operativos como Linux y Windows, aunque se cuenta con una distribución especial construida para estas placas Raspbian [23]. Cuenta con los pines analógico – digital, permitiendo realizar lecturas de elementos como sensores. También tiene una sección dedicada a realizar la regulación a 5 voltios de fuentes hasta de 12 voltios, facilitando el manejo de alimentación. Finalmente, mediante la interfaz SD es posible almacenar los distintos sistemas operativos que puede soportar la placa.

2.2 WoT

La web de objetos [24] hace referencia a un entorno donde los objetos cotidianos son identificables, legibles, reconocibles, direccionables y controlables a través de Internet. El ámbito contextual de la WoT supera al de web actual, permitiendo que los objetos físicos se puedan manejar a través de motores web.

En la actualidad se carece de una buena accesibilidad a los objetos a través de interfaces comunes, siendo esta característica esencial para la construcción de aplicaciones que exploten las capacidades en un contexto. Además, la heterogeneidad de los sistemas de computación ubicua plantea un problema importante: no hay ninguna especificación clara de las características comunes de los procesos para controlarlos o consultarlos. Los objetos físicos pueden ser dinámicos en el espacio y el tiempo, lo que implica nuevos enfoques para gestionar sus estados.

Los avances sobre el acceso virtual y el control de un gran número de objetos físicos en una plataforma común, está ganando un gran interés y su investigación se ha visto impulsada en los últimos años. Se cuenta con la especificación de los tipos de objetos que deben integrarse a través de la web, clasificándolos utilizando una estructura ontológica que identifica sus diversas propiedades. Este es un primer paso para la creación de un modelo común de gestión.

2.2.1 Objeto inteligente

Los objetos inteligentes, son todos los entes físicos o virtuales, que pueden compartir cualquier tipo de información sobre sí mismos. Estos son la base fundamental dentro del paradigma del IoT; y representan, básicamente, las necesidades de información de los usuarios.

A los objetos IoT, se les puede atribuir el desempeño de una función o rol [25], dependiendo de sus capacidades informáticas tales como: conectividad de red, energía disponible, escenarios de tiempo y espacio, entre otras. Estas características se pueden tener en cuenta al momento de clasificar los sensores de acuerdo a las capacidades que adquieren. Dichas características las podemos estructurar como:

- Básicas, un objeto puede pertenecer al mundo real o al mundo virtual.

- Generales, un objeto puede utilizar un servicio como interfaz para comunicarse con otro objeto. El objeto contiene sensores y actuadores para interactuar con el medio ambiente.

- Sociales, un objeto puede interactuar con otro objeto y con personas generando una información semántica de dicha relación.

- Autonomía de funciones, un objeto realiza tareas autónomas y tiene capacidad de razonamiento, capacidad de interactuar y aprender de otros objetos y el medio ambiente.

- Auto replicación y control de funciones, red de información con capacidades de detección. Establece relaciones entre los humanos y los sistemas físicos.

Los objetos tienen características especiales como: diferentes tipos de datos o señales [26] capacidad de almacenamiento, procesamiento, comunicación, autonomía energética, localización, origen, estado, utilización y una identificación única [27]. Es necesario definir una taxonomía sobre las características y propiedades de los objetos actuales, con el fin de fundar las bases para la conceptualización de los mismos. Así el trabajo [1], permite caracterizar los objetos basándose en tres dimensiones: comunicación, procesamiento y almacenamiento; además de una identidad. Las cuatro características proponen una clasificación de objetos de la IoT de acuerdo a sus capacidades intrínsecas

Capacidades		Capacidades Intrínsecas				Ejemplos
		I	P	C	A	
Core		√				RFID o códigos de barras.
Primitive	Fuzzy	√	√			Lavadora, hornos microondas.
	Plug	√		√		Altavoces, auriculares.
	Fat	√			√	CD, DVD.
Complex	Social	√	√	√		Control remoto, teléfonos fijos.
	Sticky	√		√	√	USB stick, RFID Tags.
	Gizmo	√	√		√	Calculadora, juegos de mano.
Smart Things		√	√	√	√	PDAs, PCs.

Tabla 2. Clasificación de los objetos del IoT. Fuente: Propia

En la Tabla 2 se distinguen cuatro características: identificación (I), procesamiento (P), comunicación (C) y almacenamiento (A).

Podemos definir los objetos inteligentes [28], como objetos de la IoT que se puede identificar claramente y exhiben capacidades de procesamiento, comunicación y almacenamiento. Estos objetos cuentan con una representación digital, la cual permite interactuar con las aplicaciones y los usuarios en la web. Debido a sus características son los elegidos para crear de manera más intuitiva servicios web semánticos y exponer su funcionalidad en la WoT. Los otros tipos de objetos necesitan desarrollar interfaces superiores que los administre.

2.2.2 Contexto

El contexto según [13, 29], hace referencia a cualquier información que se puede utilizar para caracterizar la situación de una entidad. Una entidad es una persona, lugar u objeto que se considera relevante para la interacción entre un usuario y una aplicación, incluyendo el usuario y las propias aplicaciones.

En el marco de la IoT, el contexto es un concepto subjetivo que se define por la entidad que lo percibe. Puede ser descrito generalmente como el subconjunto de estados físicos y conceptuales de interés para una entidad en particular. Se ha identificado cuatro capacidades contextuales genéricas: detección, adaptación, descubrimiento de recursos e incremento.

Según el procesamiento que se realice de la información, el contexto puede clasificarse en dos niveles importantes. Contexto primario, consta de elementos como ubicación, identidad, actividad y tiempo, que son fundamentales para caracterizar una entidad en particular. Contexto secundario, el cual se genera a partir de la fusión de datos y la transferencia del contexto primario por medio de cualquier función. Por ejemplo, la información de ubicación de un sensor GPS corresponde a un dato de contexto primario, mientras que la información resultado de realizar el cálculo de la distancia entre mi ubicación y cualquier punto, corresponde al contexto secundario

A continuación, se presentan las fases del que pueden seguir aplicaciones enfocadas en la construcción de contexto [13]

Adquisición

En esta fase, los datos deben ser adquiridos de diversas fuentes. Algunos factores que deben tenerse en cuenta para el desarrollo de soluciones son:

- Responsabilidad, la adquisición de los datos puede ser lograda mediante el método “push” donde el componente software hace consultas periódicas al hardware del sensor; o mediante el método “pull”, donde el sensor entrega los datos al componente software de forma periódica o instantánea, sin necesidad de realizar ningún tipo de consulta

- Frecuencia, cuando se genera el contexto se puede realizar en dos tipos de eventos. Eventos instantáneos, los cuales ocurren solo en un instante de tiempo, como abrir o cerrar una puerta o encender o apagar la luz. O eventos de intervalo, en este caso, los eventos que tienen una duración de un periodo de tiempo

- Fuente, el contexto puede ser adquirido de tres formas principales. Primero, de forma directa; donde el contexto se adquiere directamente del sensor. Segundo, las aplicaciones pueden recuperar los datos mediante las soluciones middleware. Tercero, el contexto puede ser adquirido en servidores de contexto, tales como las bases de datos o los servidores web.

- Tipos de sensor, se pueden definir tres tipos de sensores. Primero, físicos, son los más comunes, estos generan los datos por sí mismos. Segundo, virtuales; encargados de recuperar los datos de muchas fuentes publicarlos como datos propios. Y, por último, se encuentran los lógicos; consisten en una combinación entre sensores físicos y virtuales.

- Proceso de adquisición, existen tres formas de adquirir el contexto. Primero, la detección; los datos se detectan a través de los sensores directamente. Segundo, la derivación, los datos se generan mediante operaciones (simples o complejas) de los datos de los sensores. Y tercero, manualmente; los usuarios proporcionan información de contexto mediante ajustes predefinidos, como preferencias.

Modelado

Corresponde a la segunda fase, donde se define como se representarán los datos del contexto. No existe estándar para especificar qué tipo de información irá en el modelado del contexto. El modelado basado en ontologías se considera la técnica más popular para modelar y gestionar el contexto, ya que las ontologías ofrecen un lenguaje expresivo para representar las relaciones y el contexto, además existe una gran comunidad que la soporta.

Razonamiento

Tercera fase del ciclo del contexto. Es un método es utilizado para deducir nuevos conocimientos a partir de los contextos existentes. El razonamiento comprende 3 fases. Primero, pre-procesamiento; limpia los datos de los sensores. Segundo, fusión de datos; método de combinación de datos de los sensores encargada de generar información más precisa. Y tercero, inferencia del contexto; genera información a partir de contextos de bajo nivel.

Técnicas de razonamiento

Una sola técnica de razonamiento no puede lograr resultados perfectos, es necesario combinar múltiples modelos de forma que aumenten las probabilidades de éxito. Se establecen las siguientes técnicas como las principales para desarrollar el razonamiento del contexto.

- Aprendizaje supervisado, se construyen ejemplos de entrenamiento, se los etiqueta según los resultados esperados y se deriva una función que puede generar los resultados esperados. Un árbol de decisión es una técnica de aprendizaje supervisado.

- Aprendizaje no supervisado, esta categoría puede encontrar estructuras ocultas en los datos no etiquetados.

- Reglas, son estructuras en formato “IF-THEN-ELSE”, que permiten generar información de contexto de alto nivel utilizando contexto de bajo nivel.

- Lógica difusa, permite un razonamiento de valores aproximados. En teoría lógica tradicional, los valores de verdad aceptables son 0 o 1, para lógica difusa se aceptan valores parciales, además permite el uso de lenguaje natural (por ejemplo, temperatura: un poco caliente, bastante frío) permitiendo que escenarios del mundo real sean representados de forma más natural.

- Basado en ontologías, se basa en descripción de la lógica como una representación formal del conocimiento. El razonamiento ontológico está apoyado por dos idiomas de la web semántica, RDF, OWL, entre otros.

- Lógica probabilística, permite la toma de decisiones sobre la base de las probabilidades de los hechos relacionados con el problema.

2.2.3 Context-aware[4]

Un sistema es context-aware si se utiliza el contexto para proporcionar información y/o servicios de interés para el usuario, donde la relevancia depende de la tarea del usuario. En esta definición se hace un especial énfasis en el usuario, sin limitar la parte "aware" solo a la interfaz de la aplicación [13, 30, 31].

También el context-aware se define como: la capacidad de un programa o dispositivo para detectar diversos estados de su entorno y en sí mismo. Esta característica permite descubrir otros recursos dentro del mismo contexto y explotarlos mientras permanezcan en el mismo contexto. La información brindada por parte del contexto permite: detectar, reaccionar e interactuar con el entorno.

Para la utilización del context-aware se requiere la disposición de los contextos de otras entidades. La comprensión del contexto permite a los diseñadores de las aplicaciones elegir cuál contexto se va a utilizar.

2.2.4 Protocolos de comunicación

A continuación, se realiza una muestra de los protocolos de comunicación WoT más relevantes para el presente caso de estudio. Luego se realiza una comparación entre estos y finalmente se definen los protocolos más adecuados para ser ejercitados.

Simple Object Access Protocol – SOAP

SOAP [32] es un Protocolo para el intercambio de información estructurada en un entorno descentralizado y distribuido. Utiliza tecnologías XML para definir un marco de mensajería extensible que proporciona una construcción de mensajes que se puedan intercambiar a través de múltiples protocolos de transporte. El marco ha sido diseñado para ser independiente de cualquier modelo de programación particular

La especificación de SOAP establece que los mensajes sean codificados en XML, presentando ventajas como: representar datos de forma independiente a las plataformas o lenguajes que soporten la aplicación, mejorando la interoperabilidad.

El mensaje SOAP se compone de dos elementos, el encabezado y el cuerpo. El encabezado proporciona un mecanismo de extensión y de enrutamiento de los mensajes. El cuerpo es el elemento obligatorio, donde se encuentra la carga útil del mensaje.

State Representational Transfer – REST

Se define como una arquitectura de servicio para desarrollo apoyada totalmente en el estándar HTTP. REST transfiere el estado del servidor al cliente, asignando una representación que es transferida al cliente por una petición, la cual puede variar con

[4] Context-aware no es posible traducirlo con una palabra que encierre todo el sentido que el presente. Conciencia que tiene una entidad sobre su contexto.

diferentes operaciones. Se utilizan los métodos propios de HTTP como GET (transfiere la representación de un recurso del cliente al servidor), POST (cambia el estado de un recurso), *PUT* (cambia la representación del recurso directamente) y DELETE (borra el recurso).

Constrained Application Protocol – CoAP

CoAP [33] Es un protocolo especializado para el uso de nodos inalámbricos restringidos y limitados de baja potencia, que pueden comunicarse de forma interactiva a través de internet. COAP surge con la necesidad de suplir un protocolo web genérico que permita la comunicación en redes con necesidades especiales debido al entorno restringido, especialmente: la energía y la comunicación M2M. Este protocolo está diseñado para aplicaciones machine to machine – M2M. CoAP proporciona un modelo de interacción cliente/servidor similar al de HTTP, esto para facilitar el intercambio de información entre los dos protocolos.

Extensible messaging and presence protocol – XMPP

XMPP [34] Es un protocolo de la capa de aplicación diseñado originalmente por Jeremie Miller en 1998 para la mensajería instantánea (originalmente llamado Jabber). El protocolo consiste en un sistema de mensajería presencial, el cual usa datagramas en XML para implementar las funcionalidades necesarias. Es un protocolo libre y abierto. Fue creado con la intención de unificar los servicios de mensajería instantánea, dado que, para su época, no existía un protocolo que pudiese satisfacer dicha necesidad, lo que llevaba a situaciones incómodas, como el uso de múltiples clientes (uno por cada servidor/servicio) proceso altamente ineficiente.

MQ Telemetry Transport – MQTT

MQTT [5] es un protocolo cliente/servidor diseñado para intercambiar mensajes entre pequeños dispositivos con un reducido ancho de banda, basado en el protocolo de transporte TCP y Web Services - WS.

La arquitectura utilizada por MQTT sigue una topología de estrella, con un nodo central que hace de servidor o mediador, encargado de gestionar la red y transmitir los mensajes entre los extremos. La comunicación puede ser cifrada en el mediador para el manejo de seguridad.

Para poder realizar la comunicación es preciso que los clientes realicen una suscripción al servidor creando un canal o tópico, el cual se representa mediante una cadena de caracteres y tiene una estructura jerárquica que separa sus elementos por una barra inclinada (/). Por otro lado, el usuario receptor debe suscribirse a este canal y esperar la publicación de los mensajes.

El protocolo MQTT es idóneo para aplicaciones IoT, en las cuales se envían cantidades pequeñas de información y por tanto no se necesita un gran ancho de banda.

2.2.5 Comparación entre los protocolos

A continuación, se presenta la Tabla 3, que compara los protocolos para realizar el intercambio de información entre objetos. El protocolo SOAP no se encuentra dentro de esta comparación debido a que este será utilizado por el proyecto en el intercambio de información entre distintos módulos, este tema entra con mayor detalle en el capítulo donde se define la arquitectura.

	CoAP	**MQTT**	**XMPP**
Abierto	X	X	X
Transporte	Corre sobre UDP	Basado en TCP	Corre sobre TCP
Mensajería	Cliente servidor	publish/subscribe	Publish / Subscribe
Negociación de contenido	Si.	No.	No.
Topología	Malla	Estrella	Malla
Seguridad integrada	Medio – opcional	Medio – opcional	Alta – obligatoria
Recursos computacionales	10Ks/RAM Flash	10Ks/RAM Flash	10Ks/RAM Flash
Bajo consumo y pérdidas (1000's)	Excelente	Bueno	Justa

Tabla 3. Tabla comparativa de los protocolos IoT. Fuente: Propia

- Abierto, hace referencia a la estandarización del protocolo.

- Transporte, protocolo de transporte utilizado.

- Mensajería, forma de comunicación entre los nodos del protocolo.

- Negociación del contenido, posibilidad de realizar la comunicación de forma directa o si es necesario conocer con anterioridad el tipo de dato a recibir.

- Topología, presenta la disposición de la red.

- Seguridad integrada, seguridad de la transmisión de datos a través.

- Recursos computacionales, requisitos mínimos para el funcionamiento del protocolo en un dispositivo.

A partir del análisis de los diferentes protocolos utilizaos en la IoT se concluye que para el proyecto es factible utilizar MQTT como el protocolo mediador entre los objetos del escenario. Aunque los protocolos presentaban igualdad de ventajas para la implementación de la interacción entre los objetos del escenario, se ha seleccionado MQTT, debido principalmente a la facilidad y familiaridad del equipo en el desarrollo con este protocolo. Esta conclusión se llegó después de realizar un ejercicio simple de desarrollo, se intentó establecer una conexión simple (utilizando los tres protocolos) entre dos computadores y

enviar una cadena de texto y se pudo ver que el protocolo MQTT fue el más fácil de manejar y el que mejor se comportaba para los propósitos del presente proyecto.

2.2.6 Recomendación para descripción de objetos

La recomendación *WoT Thing Description* [35] es definida por la W3C[5] para establecer los parámetros que definan la forma en como es descrito un objeto de la WoT a partir de sus metadatos, esta recomendación propone representar la información para describir al objeto utilizando el formato JSON y un sistema de etiquetas jerárquicas, con las cuales es posible organizar las descripciones de diferentes componentes que puedan tener, además, busca que la información pueda ser entendida e intercambiada de manera universal entre los distintos sistemas que gestionan datos de la WoT, superando los problemas de interoperabilidad a nivel sintáctico.

Cada etiqueta puede ser diferenciada en tres grupos. Primero, las etiquetas tipo atributo, estas representan un valor puntual y no contiene otras etiquetas dentro, en formato JSON se puede entender como una clave-valor. Segundo, etiquetas tipo colección, estas representan varios conceptos con una misma estructura, cada concepto es una etiqueta que conforma a la colección, en formato JSON se puede entender como una clave-objeto. Y tercero, etiquetas tipo lista, estas representan una lista del mismo concepto, en formato JSON se puede entender como un lista-objeto.

WoT Thing Description [35] es utilizada en el proyecto para generar los elementos de mensajería, construir el documento de los metadatos que describe a cada objeto inteligente y construir el documento utilizado para la interoperabilidad entre objetos (explicados más a profundidad en el capítulo de arquitectura) Los archivos que se intercambian entre los OI del escenario, generados por el framework, contienen los metadatos de todos los elementos físicos y virtuales asociados con el OI. Las principales etiquetas de esta recomendación se describen a continuación.

Etiqueta Thing

La etiqueta *Thing* pertenece al nivel principal de la recomendación. Representa la abstracción física o virtual de un objeto o la fusión de varios objetos. Para el presente caso limitaremos esta definición a la abstracción física de un solo objeto. Dentro de este elemento encontramos alrededor de 17 etiquetas definidas por la recomendación, pero para el presente trabajo se limitaron a 7, siendo estas consideradas más relevantes para describir los metadatos de la primera versión del Framework. El esquema se puede ver en Figura 2.

- Etiqueta *id*. Atributo, representa el identificador del objeto. Este debe ser único y global, o al menos único para la aplicación especifica el dominio o la red de las organizaciones involucradas.

[5] World Wide Web Consortium

- Etiqueta *created.* Atributo, representa la marca de tiempo de la creación del objeto.

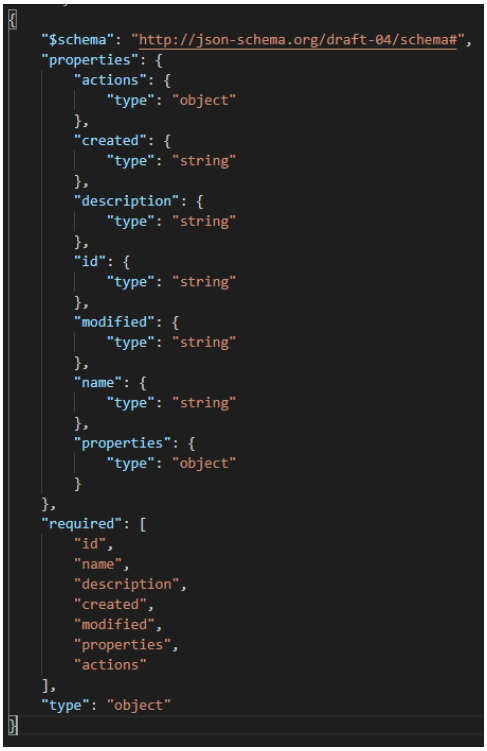
- Etiqueta *modified.* Atributo, representa la marca de tiempo de la modificación del objeto.

- Etiqueta *description*. Atributo, representa la descripción del objeto en formato de texto libre.

- Etiqueta *name*. Atributo, representa el nombre del objeto.

- Etiqueta *properties*. Colección, explicada en la siguiente sección.

- Etiqueta *actions*. Colección, explicada en la siguiente sección.



```
{
    "$schema": "http://json-schema.org/draft-04/schema#",
    "properties": {
        "actions": {
            "type": "object"
        },
        "created": {
            "type": "string"
        },
        "description": {
            "type": "string"
        },
        "id": {
            "type": "string"
        },
        "modified": {
            "type": "string"
        },
        "name": {
            "type": "string"
        },
        "properties": {
            "type": "object"
        }
    },
    "required": [
        "id",
        "name",
        "description",
        "created",
        "modified",
        "properties",
        "actions"
    ],
    "type": "object"
}
```


Figura 2. Esquema JSON etiqueta Thing. Fuente: Propia

Colección de etiquetas *Properties*

La colección de etiquetas *properties* pertenece a un segundo nivel de la recomendación. Esta colección puede tener dentro una a muchas etiquetas y cada una representa un estado específico del objeto. Como la definición entregada por la recomendación de esta etiqueta resulta muy amplia, se decide tomar para el presente trabajo, este elemento como una colección de recursos del objeto. Cada recurso será descrito en los metadatos como una etiqueta, siendo el valor que entrega el recurso el estado del objeto definido por la recomendación.

Cada etiqueta dentro de la colección es nombrada según el identificador del recurso y contiene alrededor de 13 etiquetas definidas por la recomendación, pero para el presente trabajo se limitaron a 6, siendo estas consideradas más relevantes para describir los metadatos de la primera versión del Framework

- Etiqueta *description.* Atributo, representa la descripción del recurso en formato de texto libre.

- Etiqueta *name.* Atributo, representa el nombre del recurso.

- Etiqueta *type.* Atributo, representa el formato de valor que retorna o acepta el recurso.

- Etiqueta *unit.* Atributo, representa la base de medición del valor retornado o aceptado por el recurso.

- Etiqueta *const.* Atributo, representa un valor inmutable utilizado por el recurso.

- Etiqueta *forms.* Lista, explicada en la siguiente sección.

Lista *Forms*

La lista *Forms* pertenece a un tercer nivel de la recomendación. Esta lista puede tener dentro uno a muchos ítems y cada uno representa un método de comunicación que puede exponer un recurso a la red.

Cada ítem dentro de la lista contiene alrededor de 8 etiquetas definidas por la recomendación, pero para el presente trabajo se limitaron a 4, siendo estas consideradas más relevantes para describir los metadatos de la primera versión del Framework. El esquema se lo puede ver en la Figura 3.

- Etiqueta *href.* Atributo, representa la dirección web o url donde puede ser consumido un recurso

- Etiqueta *methodName.* Atributo, representa el método con el cual puede ser consumido un recurso. Para el presente trabajo solo son soportados métodos REST y CoAP

- Etiqueta *op.* Atributo, representa la operación que es posible realizar sobre la dirección web compartida en la etiqueta *href*

- Etiqueta *protocol.* Atributo, representa el protocolo soportado para este ítem de comunicación

Figura 3. Esquema JSON etiqueta Properties. Fuente: Propia

Colección de etiquetas *Actions*

La colección de etiquetas *actions* pertenece a un segundo nivel de la recomendación. Esta colección puede tener dentro una a muchas etiquetas y cada una representa una función que modifica uno o varios estados del objeto. Para el presente trabajo, se limita esta colección a una sola etiqueta, esto para simplificar la construcción de los documentos que contienen la interoperabilidad del objeto.

La etiqueta dentro de la colección tiene un nombre prestablecido y contiene alrededor de 4 etiquetas definidas por la recomendación, pero para el presente trabajo se limitaron a 2, siendo estas consideradas más relevantes para describir los metadatos de la primera versión del Framework

- Etiqueta *input.* Colección, representa los datos de entrada para ejecutar la función.

- Etiqueta *output.* Colección, representa los datos de salida entregados por la función.

2.3 WEB SEMÁNTICA

La web semántica nace en respuesta a la demanda en nuevos servicios y funcionalidades y al rápido avance y crecimiento de la web. Actualmente en la web existen varias facilidades para que el usuario humano pueda encontrar información, pero no es posible que una máquina, como un agente autónomo, pueda realizar búsquedas similares con el mismo éxito, esto, debido a que la información de la web no está bien estructurada, es decir, no se define formalmente la información. Esta falta de formalidad supone diversos problemas para las herramientas de la web actual [4].

En [36] se ha denominado web semántica a la web donde las herramientas como: aplicaciones y servicios, serán inteligentes. Estas tendrán la capacidad de procesar

información de forma más profunda, “entendiendo” el contenido de los documentos y las páginas web, comparando información entre varias fuentes y realizando inferencias y deducciones lógicas para poder recuperar información más cercana a las necesidades del usuario.

2.3.1 Metadatos

La manera que se ha propuesto para codificar los significados de la información que contiene la web, es por medio de etiquetas las cuales especifiquen objetivamente el contenido, dándole a la información una interpretación más correcta. Sin embargo, existe cierta incertidumbre sobre qué etiquetas son relevantes y en qué formatos deben estar especificadas, debido a que las etiquetas son entregadas por los usuarios humanos. Para esto se ha propuesto reunir y normalizar las etiquetas en un solo documento o recurso llamado metadatos.

Los metadatos corresponden a un conjunto de propiedades y atributos necesarios para etiquetar, catalogar, describir y clasificar información de una fuente o recurso [37] presentan información del contenido, especificaciones formales (tamaño, formato, idioma), derechos de autor, autenticación y finalmente el contexto. Estos se han convertido en una base fuerte sobre la cual se construye la web semántica, haciendo posible que las aplicaciones de la web semántica, puedan obtener el significado de los recursos que recuperan.

Para lograr aplicaciones y servicios que contengan más profundidad semántica, es necesario organizar los valores y propiedades de metadatos en repositorios especiales, estos deben permitir que los significados puedan relacionarse dependiendo del dominio de conocimiento (jerarquías, etc.) posibilitando que las aplicaciones realicen operaciones de razonamiento lógico entre los metadatos.

2.3.2 Servicios y aplicaciones de la web semántica

A continuación, se presentan algunas aplicaciones y servicios que pueden ser implementados utilizando herramientas de la web semántica.

- Recuperación de la información, a diferencia de los buscadores actuales, que realizan la búsqueda basándose en palabras clave; los buscadores semánticos realizan la búsqueda basados en el significado de las palabras, entregando al usuario documentos o páginas que se relacionen con los significados de la búsqueda.

- Publicación de la información, los usuarios de la web semántica visualizarán simplemente la información que se relacione con sus necesidades, filtrando la información de los documentos que no sean relevantes.

- Interfaces inteligentes, las herramientas semánticas permiten la creación de aplicaciones con interfaces inteligentes, como las basadas en lenguaje natural.

- Sistemas de inferencia, es posible que se presente información útil al usuario a partir de inferencias y deducciones lógicas.

- Intercambio de información, aumentar la compatibilidad entre sistemas heterogéneos, normalizando la información de diferentes aplicaciones con formatos y organizaciones independientes.

La web semántica permite también la creación de servicios, los cuales pretenden resolver las necesidades de las aplicaciones mediante la integración de diversas soluciones web. En primera instancia, los servicios web semánticos deben ser descubiertos, descritos e invocados por las aplicaciones, esto se logra gracias a la estandarización de protocolos como WSDL, UDDI y SOAP [38]. Por otra parte, estos servicios deben permitir que las aplicaciones puedan entender su alcance y posibilidades, con el fin de que puedan decidir su utilidad para lograr los objetivos de procesamiento e integración automática [39].

2.3.3 Indexación semántica

La indexación es el mecanismo para representar el contenido de un documento mediante un conjunto de términos o conceptos. Un documento es cualquier fuente de información, en este caso una fuente de información digital. Se entiende como concepto a una unidad de pensamiento, el cual se puede expresar como una combinación de otros conceptos.

Para crear un índice, se debe hacer un proceso de representación de las fuentes documentales mediante un conjunto de términos o conceptos, de tal forma que se pueda registrar ordenadamente los temas de los que trata dicho documento con el fin de permitir una clasificación y consulta rápida.

La indexación se hace semántica cuando a las herramientas y métodos de indexación se le suman elementos que permiten aumentar el significado de los conceptos, permitiendo consultas con un mayor grado de relevancia para los usuarios. Este tipo de indexación se enfoca en la asociación de los conceptos con el fin de buscar mayor precisión en los significados.

Según [40] las características de un índice semántico son:

- Multidimensionalidad, los elementos de indexación son valores de atributos que pueden estar basados en complejas descripciones.

- Adaptabilidad, el índice es altamente adaptable a las necesidades de cada proyecto. Los conceptos de indexación pueden ser añadidos o eliminados como se desee, lo cual los hace muy densos y precisos con respecto al interés de un grupo de personas.

- Disponibilidad, dado que el índice es en realidad un conjunto de descripciones parciales de los objetos indexados, mucha información se puede extraer directamente del índice.

Para realizar un índice semántico es preciso utilizar un tesauro o una ontología, para saber cuál se debe escoger es preciso tener en cuenta que un tesauro agrupa un conjunto de palabras, lo que nos sirve para trabajar con un idioma en particular; mientras que la ontología presenta un nivel semántico más alto, describiendo mejor el objeto, sus prioridades y relaciones. Si se desea, se puede utilizar las dos herramientas para adaptar mejor los dominios.

2.3.4 Ontologías

La definición más aceptada sobre las ontologías fue dada por Thomas Gruber, y reza: "Una Ontología es una especificación formal y explícita de una conceptualización compartida" [41] donde el término "especificación formal" sugiere que los elementos deben estar expresados siguiendo formalidades idénticas, permitiendo que la ontología pueda ser reutilizada y leída por cualquier máquina, cualquier plataforma o lenguaje. El término "especificación explícita" refiere que todos los elementos no deben dar nada por supuesto o por obvio. La palabra "conceptualización" aclara que las ontologías desarrollan modelos abstractos del mundo real. La palabra "compartida" es el elemento de la afirmación más importante, ya que una ontología, lo es, cuando sus conceptualizaciones sobre el mundo real y su formalización del conocimiento, es aceptado y acogido por todos los usuarios y posibles usuarios de dicha ontología.

Básicamente las ontologías funcionan como cualquier modelo conceptual existente [42], sin embargo, estas se diferencian de otros modelos en tres aspectos principales. Primero, las ontologías buscan la integración de información entre aplicaciones. Segundo, permiten que los significados sean descritos sin ambigüedades. Y tercero, permiten realizar razonamiento utilizando cálculos en tiempos de ejecución.

Algunos de los lenguajes mayormente utilizados para las ontologías son: XOL [43], RDF [44], OIL [45], DAML [46] y OWL [47], los cuales están basados en el lenguaje XML para el intercambio y estandarización de la información.

2.3.5 Distancia semántica

También puede ser encontrada como su opuesta similitud semántica. Ambas reflejan una medida entre dos palabras o conceptos y la diferencia entre la información compartida, además, partiendo que cada concepto contiene un significado en un contexto compartido es posible calcular y transformar en un valor objetivo, cuanto difiere el significado de dos conceptos dados.

Existen varias técnicas para lograr obtener este valor de diferencia, sin embargo, bajo el marco de las herramientas de la web semántica, es comúnmente utilizada la fórmula de Wu & Palmer [48] la cual define qué, dada una ontología de referencia que tiene un conjunto de nodos como conceptos, tomamos s1 y s2 como dos conceptos de entrada que se

encuentren en la ontología, donde lcs() corresponde a una función para calcular los pasos para llegar de s1 a s2 y depth() corresponde a una función para calcular los pasos para llegar del nodo principal a s1 o s2, obtenemos un número entre 0 y 1, donde 0 significa que s1 es totalmente opuesto a s2 y 1 significa que s1 es igual a s2. La fórmula se puede ver en la Figura 4.

$$\mathbf{Wu - Palmer} = 2 * \frac{\mathbf{depth}\ (\mathbf{lcs}(\mathbf{s1}, \mathbf{s2}))}{(\mathbf{depth}\ (\mathbf{s1}) + \mathbf{depth}\ (\mathbf{s2}))}$$

Figura 4. Formula calculo similitud semántica de Wu & Palmer. Fuente: Propia

La distancia semántica o similitud semántica es utilizada usuamente para conocer la diferencia entre dos términos a nivel de su significado, sin importar su nivel sitáctico, por ejemplo: la el término “hogar” y el término “casa” no tienen a nivel sitntáctico una gran equivalencia, sin embargo dado un contexto su significado puede ser el mismo. El el presente proyecto esta formula es utilizada con el fin antes expuesto dado un contexto soportado por una ongología de dominio.

3 ESTADO DEL ARTE

Este capítulo presenta los estudios de revisión relacionados con los núcleos temáticos principales, IoT, web semántica, interoperabilidad. En primer lugar, se realizó un análisis de mapeo científico con la ayuda de las herramientas CiteSpace [49] y SciMAT [50], con el fin de obtener los aspectos estructurales y dinámicos del campo de investigación para luego ser incluidos como insumos para la revisión sistemática.

Con las conclusiones obtenidas en el análisis del mapeo científico, se realizó una revisión de la literatura en diversas bases de datos. Como metodología de apoyo se utiliza el modelo de investigación documental [51, 52], que permiten abordar, desde una perspectiva científica, la construcción de una visión global del conocimiento en un área temática determinada.

El modelo sugiere la construcción de fichas bibliográficas, las cuales contienen un resumen e interpretación de la literatura encontrada, estas se pueden encontrar en el anexo A.

3.1 MAPEO CIENTÍFICO

El mapeo científico fue realizado sobre el término “semantic web” el cual es considerado el principal concepto y que puede correlacionarse fácilmente con los otros núcleos temáticos propuestos en el proceso del estado del arte.

Este ejercicio es utilizado principalmente para detallar la evolución del concepto y como es impactado por diferentes disciplinas y cuál es su relevancia en el ámbito científico. Para analizar esta evolución es necesario implementar una revisión cienciométrica [53]. Todos los artículos usados para este estudio han sido publicados en la colección principal de *Web of Science* (WOS) y *Scopus* la metodología de búsqueda en estas herramientas fue aportada por los estudios de [54, 55]. Las herramientas de soporte usadas son CiteSpace y SciMAT

3.1.1 Conjunto de datos para análisis

El conjunto de datos bajo el cual se realiza el análisis es de 21.039 registros para fuentes para Scopus y 12.891 para WOS, el período de búsqueda se realizó entre 1996 a 2020Q1[6]. Los registros consultados son documentos relacionados con artículos de revistas y actas de conferencias. Los criterios de exclusión se enfocaron en filtrar los documentos que no cumplieran con la definición completa (ejemplo, que tengan web o semántica de forma separada) y que no se encuentren en las áreas afines a la informática.

La Figura 5 proporciona una visión a largo plazo de los resultados por tiempo, incluidos los resultados de 1995 al primer trimestre de 2020 para Scopus y de 1996 a 2017 para la colección principal de WOS. Utilizando esta visión por tiempo podemos ver la evolución en cuanto a número de publicaciones.

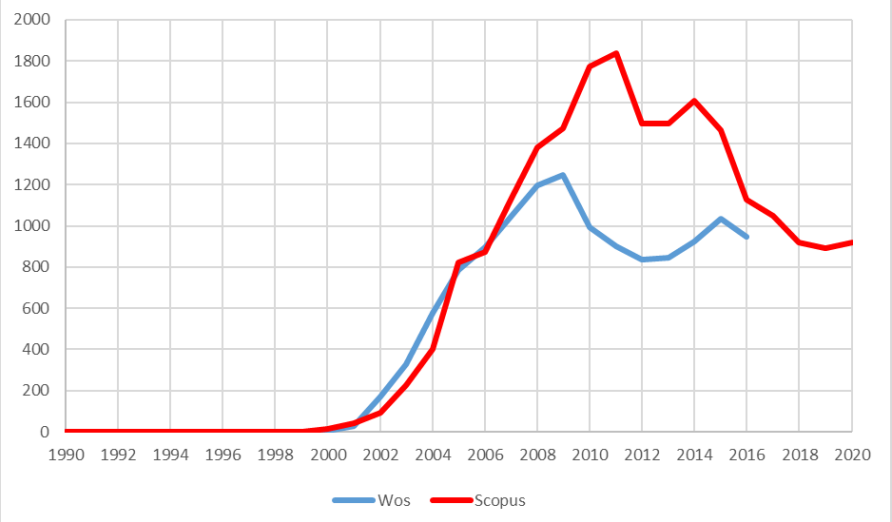


Figura 5. Distribución de documentos a lo largo del tiempo para Scopus y WOS. Eje X: años, Eje Y: cantidad de documentos. Fuente: Propia

3.1.2 Análisis con CiteSpace

CiteSpace se utiliza para aplicar tres tipos de técnicas bibliométricas [56]. El primero es el análisis de coautores, que toma los nombres de los autores, los países de afiliación y las afiliaciones institucionales como unidades de análisis y examina las coincidencias de

[6] Primer cuartil del año. Primeros meses del año enero, febrero, marzo

autores, las coincidencias de países y las coincidencias de instituciones. El segundo es el análisis de co-palabras, basada en palabras clave. Y, por último, el análisis de co-citaciones que identifica a los autores y a las revistas citadas conjuntamente, basándose en referencias, artículos, autores y revistas.

La productividad de los autores en un campo se puede representar aplicando análisis de coautoría. La red de coautoría resultante, que se aprecia en la Figura 6, se compone de 706 nodos y 674 enlaces, que contiene todos los autores principales del campo. Cada nodo representa un autor, y los enlaces entre los autores representan la coautoría de los artículos. Los diferentes nodos representan el número de publicaciones proporcionalmente. Está claro que Enrico Motta (Motta E), profesor de la Open University en el Reino Unido, es el autor más citado en el campo de la Web Semántica, seguido de Antoniou G., Bassiliades N. y Decker S. El ancho de las líneas y círculos indica la proporción de las relaciones cooperativas y el número de publicaciones.

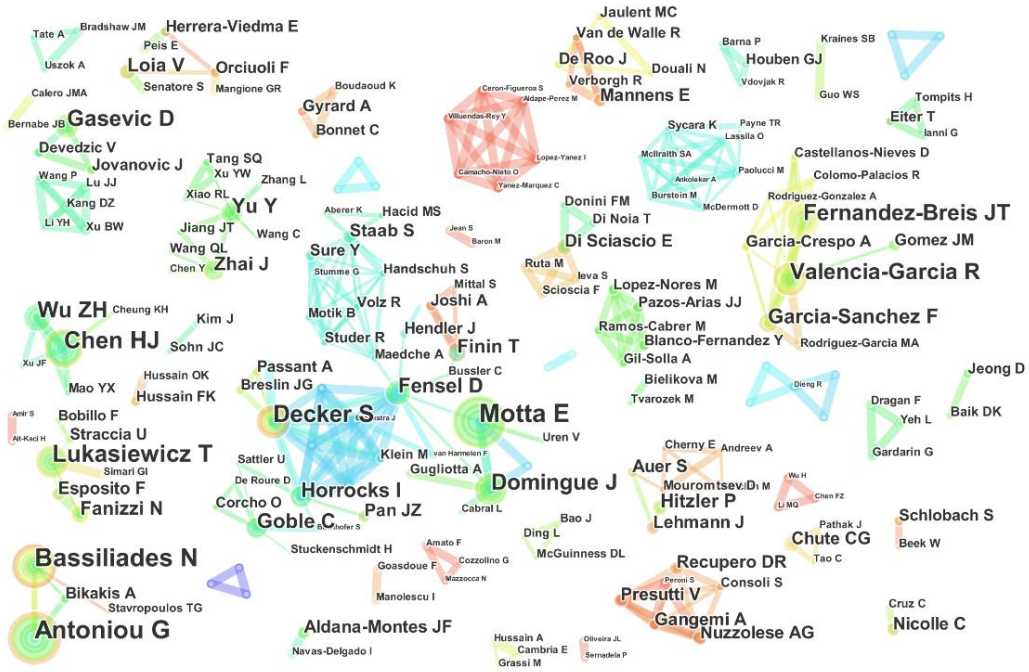


Figura 6. Red de coautoría, con 706 nodos y 674 enlaces. Fuente: Propia

La red de palabras clave concurrentes resultante contiene 292 nodos y 1139 enlaces, se muestra en la Figura 7. Cada nodo de palabras clave está compuesto por varios anillos, y los anillos y enlaces están representados en un conjunto de colores correspondientes a los años de ocurrencia. Para facilitar la lectura, la Figura es centrada solo en la parte más poblada de los clústeres. Las palabras clave más relevantes por conteo son *Ontology* con recuentos de citas de 2836, seguida por *System* con 621 y *Web* con 618.

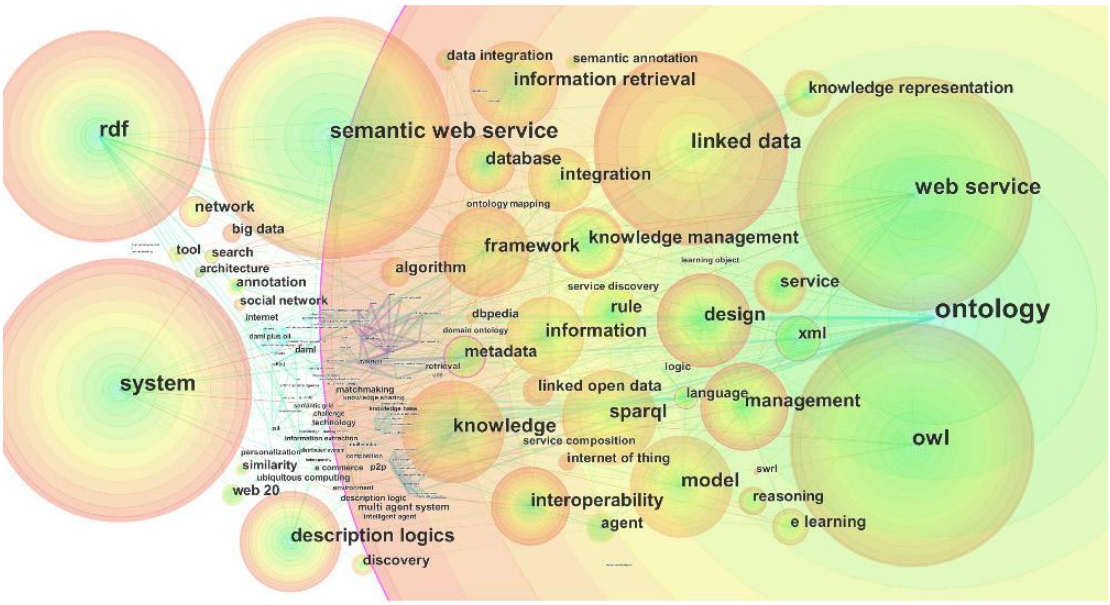


Figura 7. Red de palabras clave concurrentes, con 292 nodos y 1139 enlaces. Fuente: Propia

3.1.3 Análisis con SciMAT

Para el análisis en SciMAT los registros fueron organizados en 5 periodos diferentes, el criterio de separación fue intentar homogeneizar el número de documentos por periodo. Por otro lado, el análisis nos propone dividir los conceptos recuperados de los registros en 4 posibles áreas. Cada concepto contiene valores de densidad y centralidad, los cuales ubican al término en una de estas áreas del desarrollo de la investigación.

- Conceptos motores (con alta densidad y alta centralidad). Altamente desarrollados e importantes para la construcción del campo científico y con una alta adopción (poco asilados)

- Conceptos transversales (con baja densidad y alta centralidad). Importantes para el desarrollo del campo científico, estables y una alta adopción (poco asilados), pero con poco desarrollo

- Conceptos emergentes o en declive (con baja densidad y baja centralidad). Poco desarrollados con una evolución al alza o en retroceso

- Conceptos periféricos (con alta densidad y baja centralidad). Altamente desarrollados internamente, pero que se encuentran aislados, tienen un papel marginal en el desarrollo del campo científico

Primer periodo 1996-2001

Este periodo fue donde empieza el desarrollo de la investigación en Web Semántica. Para este periodo podemos encontrar como concepto principal *Internet*, densidad (13.24), centralidad (40.55) Este concepto tiene una alta centralidad y alta densidad. Esto indica que

es un concepto motor y está altamente desarrollado y compartido por los documentos científicos.

Segundo periodo 2002-2006

En este periodo se aumentó el número de documentos. Para este periodo podemos encontrar como conceptos principales para el análisis los siguientes:

- *Owl*, con valores: densidad (0.43), centralidad (4.71). Estos valores lo ponen como un concepto emergente, que, aunque no tiene un alto desarrollo, está en crecimiento y adopción

- *Context*, con valores: densidad (0.36), centralidad (4.27). Estos valores lo ponen como un concepto emergente, que, aunque no tiene un alto desarrollo, está en crecimiento y adopción

- *Framework,* con valores: densidad (1.57), centralidad (3.4). Estos valores lo ponen como un concepto básico y transversal, el cual no ha tenido mucho desarrollo, pero es considerado como importante y está medianamente adoptado

- *Application-software*, con valores: densidad (2.91) y centralidad (7.08). Estos valores lo ponen como un concepto motor.

Tercer periodo 2007-2011

Para este periodo podemos encontrar como conceptos principales para el análisis los siguientes:

- *Information-retrieval*, con valores: densidad (0.06), centralidad (3.14). Estos valores lo ponen como un concepto básico y transversal, que tiene poco desarrollo, es considerado como importante y está medianamente adoptado

- *Jena,* con valores: densidad (0.26), centralidad (3.24). Estos valores lo ponen como un concepto básico y transversal, que tiene poco desarrollo, es considerado como importante y está medianamente adoptado

- *Semantic-discovery*, con valores: densidad (1.01), centralidad (3.5). Estos valores lo ponen como un concepto motor, que tiene un mediano desarrollo y una mediana adopción

Cuarto periodo 2012-2017

Durante este período se aumentó la cantidad de documentos. Para este periodo podemos encontrar los conceptos principales separados por área de desarrollo fueron

- Los conceptos motores fueron: *SWS*, *Middleware* y *Tag-recomendation*.

- Los conceptos periféricos fueron *Hbase*, *Query-expansion* y *Metrics*.

- Los conceptos emergentes fueron *RDF*, *Service-composition* y *Classification*

- Los conceptos transversales fueron: *Entity-resolution*, *Linked-data* y *Ambient-intelligence*

Además, los conceptos con más documentos asociados o con más recuento fueron: *Ontologies* con 880 documentos, *Service-composition* con 298 documentos y *Linked-data* con 259 documentos

Quinto periodo 2018-2020Q1
Para este periodo podemos encontrar como conceptos principales para el análisis los siguientes:

- *SPARQL,* con valores: densidad (0.33), centralidad (2.29). Estos valores lo ponen como un concepto transversal, el cual tiene poco desarrollo, pero tiene una mediana adopción e importancia

- *Embedding*, con valores: densidad (0,48), centralidad (1,52). Estos valores lo ponen como un concepto emergente, que, aunque no tiene un alto desarrollo, está en crecimiento y adopción

- *Ontology*, con valores: densidad (0.43), centralidad (2.23). Estos valores lo ponen como un concepto en declive, tiene un bajo desarrollo, y como en periodos anteriores estaba altamente posicionado, la adopción de este tema está en bajada

- *Interoperability,* con valores: densidad (0.53), centralidad (2.06). Estos valores lo ponen como un concepto emergente, que, aunque no tiene un alto desarrollo, está en crecimiento y adopción

3.1.4 Conclusión del análisis

La investigación en Web Semántica está bien definida en las fuentes seleccionadas (WOS y Scopus), esto significa que ambos tienen registros similares y su evolución en ambas fuentes es similar. Los resultados para el análisis de palabras clave en ambas herramientas son similares.

Los resultados para SciMAT son más explícitos según períodos y podemos organizar de una mejor forma los resultados, además de filtrar, corregir y unir, conceptos con significados similares. Para CiteSpace tenemos una mayor riqueza visual y podemos hacer un análisis más global, debido a los gráficos con alto grado de información y versatilidad que entrega la herramienta.

Aunque en los últimos años la investigación en esta área ha bajado considerablemente, no se ha perdido el propósito de ahondar en herramientas donde las computadoras puedan tener mayor inteligencia, esto se puede ver en este documento cuando aparecen conceptos nuevos como *IoT*, *Natural Language Processing* o *Big Data*.

Finalmente se puede ver que existen esfuerzos en reducir complejidad al desarrollo de herramientas de la web semántica, debido a que, desde el segundo periodo ya podemos ver el concepto *Framework* como un concepto básico y transversal.

3.2 TABAJOS RELACIONADOS

3.2.1 Descripción trabajos relacionados

Se pueden encontrar proyectos que generan entornos específicos de interacción de objetos IoT en contextos bien definidos como: educación [57] y asistencia médica [58] mediados por servidores middleware hechos a medida. Los servidores están encargados de gestionar la información de los objetos, efectuando procedimientos que calculan y despliegan las tareas y acciones sobre los datos presentados a los usuarios, la definición de la gestión es responsabilidad directa de su propietario y desarrollador de la aplicación. Así, es posible construir entornos de interacción de objetos IoT que facilitan la vida y los procesos cotidianos de las personas, sin embargo, estos proyectos no utilizan OI, ni tampoco herramientas semánticas, relegando muchas configuraciones y decisiones del sistema al usuario, a otros actores humanos o aplicaciones terceras. Por lo tanto, aunque las soluciones son eficientes, no son fácilmente replicables, ya que se debe construir casi desde cero las nuevas interacciones, funciones e incluso la construcción de nuevo hardware. Finalmente, estos trabajos se enfocan principalmente a la interacción entre objetos con personas, a diferencia del presente proyecto que se basa en la interacción entre OI.

Uno de los trabajos importantes para la presente investigación es el desarrollado en [31] en cual se presenta un análisis profundo sobre el entendimiento del contexto en la IoT, además del trabajo presentado en [30], el cual se enfoca en reducir las tareas que el usuario debe realizar para conectarse con los servicios que proveen los objetos. El presente trabajo utiliza los conceptos de contexto introducidos por Perera, con el fin de hacer un mejor manejo de las interacciones entre objetos. Otro aporte importante, lo podemos resumir, en el descubrimiento y la configuración automática de los objetos, con lo cual es posible utilizar los servicios de los objetos descubiertos en cualquier entidad del sistema. El modelo presentado (CADDOT) y la herramienta desarrollada (SmartLink) permiten utilizar técnicas semánticas de anotación de información en la ontología SSN-XG y su configuración en un middleware propio, posibilitando su consumo posterior por usuarios y aplicaciones. Los puntos débiles de la solución propuesta son:

- No todos los objetos vienen con la posibilidad de ser consultados vía protocolos estándares como CoAP o MQTT

- No se explota la potencialidad de cada objeto en cuanto a su capacidad de procesamiento y almacenamiento. Estas características se delegan al middleware.

- El proyecto no aborda la interoperabilidad entre OI, solo ofrece ideas de cómo sería un dialogo de identificación entre un objeto y un ente inteligente centralizado.

El proyecto [28] presenta un resumen de tipos de OI caracterizados en tres dimensiones: conciencia, representación e interactividad. La dimensión de conciencia se refiera a la habilidad de un OI de entender los eventos de su entorno y las actividades humanas. Los OI en esta dimensión pueden tener tres niveles de conciencia: enfocada en la actividad, enfocada en las reglas y enfocada en los procesos. La dimensión de representación se refiere al modelo de aplicación y de programación del OI. La dimensión de interacción denota la habilidad del OI de conversar con el usuario. La actual propuesta utiliza OI donde la dimensión de conciencia entra en el nivel: enfocado a las reglas, debido a que la interacción con el usuario es básica y está basada en reglas para establecer la configuración inicial de los servicios.

El proyecto [59], define cómo se pueden solucionar los problemas de interacción en la IoT, utilizando las tecnologías de la web semántica. Para esto, definen las características de un objeto y su despliegue en un espacio inteligente. Se utilizaron ontologías para modelar el razonamiento de la información compartida entre los objetos y los usuarios, concluyendo que: las tecnologías semánticas pueden ser aplicadas en el dominio de IoT para generar interacciones más enriquecidas. El proyecto permite establecer de primera mano, los elementos básicos que un entorno de objetos debe tener en cuenta para su despliegue, como: el descubrimiento de servicios, mensajería común para todos los participantes y notificación de eventos. Estos elementos solucionan, en parte, los problemas de interacción centrada entre el usuario y el objeto. Lamentablemente, el artículo no es muy claro en la arquitectura utilizada y en la solución completa, ya que muestra en anexos apartes de la ontología, ejemplos de consulta e ilustraciones de despliegue. Además, la evaluación de la solución propuesta es más descriptiva que técnica.

El proyecto [60] propone, una plataforma para ofrecer servicios semánticos integrados (ISSP). Este enfoque resuelve tres principales problemas: el descubrimiento de dominios IoT distribuidos, representación semántica dinámica entre una gran cantidad de recursos IoT en tiempo real y un repositorio semántico de datos para archivar grandes cantidades de datos recogidos por dispositivos IoT. ISSP maneja varios servicios de conocimiento de dominio utilizando ontologías para cada dominio. Estas ontologías son creadas por una herramienta web y manejadas por un servicio de integración de ontologías, el cual administra las ontologías según el dominio. Este proyecto tiene un fuerte vínculo con la actual propuesta, sin embargo, la principal diferencia recae en la ubicación y el manejo de consultas a la ontología. El proyecto de Ryu centraliza estas características en un sistema basado en la nube y el presente proyecto distribuye estos aspectos en cada OI.

El proyecto de [61], plantea la vigilancia a distancia sobre los estados de objetos provistos de sensores. Para ello, se implementó una arquitectura donde se propone la descentralización de la inteligencia, la cual debe estar repartida en la capa física de sensores, la capa de coordinación y la capa de supervisión. La inteligencia del objeto se encuentra descentralizada, por un lado, en el objeto mismo y por otro lado un elemento coordinador. Este enfoque facilita el desarrollo de servicios y la utilización de diversos objetos con distintos propósitos, sin tener que depender totalmente de las configuraciones y capacidades de otros elementos del sistema. A diferencia del trabajo de Piyare, la actual propuesta supone el uso de diversas plataformas y tipos de OI y no necesariamente una red de sensores inalámbricos, aunque su prueba de concepto pone al descubierto los problemas que se deben tener en cuenta para implementar una red de sensores, como: un protocolo eficiente y común de comunicación y un manejo adecuado de la energía utilizada por los objetos.

El proyecto [62] se enfoca principalmente en aportar en un sistema de comunicación estándar entre todos los objetos, implementar un controlador de eventos y generar nuevas aplicaciones y servicios para administrar las funciones de los objetos del hogar. A pesar de que el proyecto de Rosen tiene una similitud al actual proyecto, se presentan dos inconvenientes principales:

- Almacena los objetos en un servidor centralizado restringiendo el contexto a un montaje específico y a medida.

- Las aplicaciones tienen parámetros predefinidos (ejemplo, apagar temperatura, prender luz) limitando las posibilidades que se puedan generar entre la interacción entre los dispositivos.

En el trabajo [63] se presenta un Framework para desarrollar servidores basados en semántica, los cuales funcionan como intermediarios entre objetos. El Framework propone que terceras partes (desarrolladores) puedan realizar aplicaciones genéricas para ambientes donde hay varios objetos de diferentes fabricantes. Estas aplicaciones serán consumidas a través de un servidor central, el cual provee capacidades semánticas a las aplicaciones. El servidor intermediario entre las interacciones de los objetos de un entorno pretende aumentar las capacidades de interoperabilidad del sistema; además, la ontología central que se soporta en el servidor será una herramienta mediadora para alinear ontologías de dominio de los distintos fabricantes. Este trabajo tiene una fuerte relación con el presente proyecto ya que desarrolla principalmente un Framework dirigido a desarrolladores de la IoT. Sin embargo, la lógica del sistema está centralizada en un servidor, desaprovechando las capacidades que los objetos finales pueden aportar para su interacción.

El proyecto [64] se enfoca en reducir el esfuerzo para los desarrolladores IoT quitando las barreras que genera la adaptación a las herramientas semánticas. Para lograr esto el proyecto propone una metodología para modelos dirigidos y un generador de librerías de

ontologías (OLGA). La metodología divide el trabajo en tres actores: experto en ontologías, experto del dominio y el desarrollador IoT. Los actores expertos seleccionan o diseñan las ontologías necesarias para la solución y el actor desarrollador utiliza OLGA para generar las librerías (en un lenguaje de programación determinado del desarrollador) que manejan las consultas a las ontologías. Utilizando OLGA, el manejo de las ontologías es transparente para el desarrollador, sin embargo, en la implementación de entornos de interoperabilidad semántica, existen, además del trabajo con ontologías, otros elementos que aumentan el esfuerzo de los desarrolladores IoT.

El proyecto [65] propone un Framework semántico para construir aplicaciones IoT, en donde cada dispositivo IoT contiene su propia herramienta de anotación y razonamiento semántico. Esta propuesta se divide en tres capas:

- Capa física los dispositivos sensores y actuadores son conectados a una pasarela (*Gateway)*, donde son anotados semánticamente y el documento de anotación es guardado localmente.

- Capa de nube. La información que llega desde la capa física es procesada para ser utilizada en los distintos módulos. Primero el módulo de almacenamiento en el cual se guarda la información en bases de datos y también en ontologías. Esta información es enriquecida por un módulo razonador. Existe un módulo de consumo de esta información para que pueda ser desplegada a los usuarios finales

- Capa de Aplicación. Se despliega la información deseada a los usuarios finales

Aunque este proyecto propone un Framework de desarrollo que incluye herramientas semánticas de forma distribuida, no contempla la generación o gestión de servicios de interoperabilidad y se limita a la construcción de aplicaciones para dispositivos que funcionan de manera separada o con una interoperabilidad a la medida.

El proyecto [66] Propone un modelo con enfoque de computación en la niebla, utilizando herramientas semánticas, como ontologías y razonamiento sobre dispositivos IoT, con el fin de lograr interoperabilidad entre ambientes IoT heterogéneos. El modelo separa las capacidades semánticas, una parte se procesa en el dispositivo IoT (*fog_device*) y otra en la nube, esto con el fin de lograr mayor eficiencia en la ejecución de aplicaciones que necesiten análisis en tiempo real. Separa los dispositivos IoT en dos capas, la capa de dispositivos IoT de adquisición de datos, donde se obtienen los datos de los sensores y actuadores y la capa de dispositivos IoT semánticos, los cuales gestionan herramientas semánticas para la interoperabilidad.

Aunque el proyecto propone un modelo para construir dispositivos IoT con herramientas semánticas, de forma distribuida, los cuales manejan de forma independiente la interoperabilidad con otros dispositivos, los dispositivos que manejan estas capacidades no obtienen directamente los datos de los recursos, siendo estos un dispositivo tipo pasarela

(gateway) o middleware, además separan parte de las herramientas semánticas en servicios de la nube, aunque esto se hace con el fin de manejar análisis de aplicaciones en tiempo real, reduce la capacidad de los dispositivos IoT los cuales dependen de una entidad central.

El proyecto [67] propone un Framework para facilitar el desarrollo de sistemas IoT complejos. Teniendo en cuenta que los sistemas IoT tienen elementos delicados (robots, sistemas de salud, sistemas de carga) sobre los cuales no es aceptable errores en su desarrollo, el Framework busca entregar a los desarrolladores las funcionalidades principales para reducir estos riesgos. Esta solución se enmarca en un modelo que tiene las siguientes características:

- herramientas semánticas libres para mejor las operaciones de interoperabilidad e intercambiar datos y características de control entre dispositivos.

- herramientas de desarrollo para implementar soluciones que faciliten prototipos y la integración con otros sistemas IoT

Aunque esta propuesta es una de la más cercana al presente proyecto, centra la capacidad de las aplicaciones en estandarizar flujos de desarrollo para reducir posibles errores en sistemas IoT. Aunque el modelo propuesto utiliza herramientas semánticas para compartir información entre dispositivos IoT, la finalidad no es la generación de nuevos servicios entre los dispositivos.

También existen soluciones comerciales como [68, 69, 70] donde se proveen servicios de domótica que permiten al usuario el control a distancia de los objetos del hogar utilizando aplicaciones móviles o web. Cuentan con objetos como: equipos de audio, equipos de video, iluminación, alarmas, cámaras, cortinas, sistemas de riego, entre otros. El principal inconveniente de estas propuestas es que los desarrollos de están restringidos a marcas específicas, acotando la solución solo a un tipo de objetos, a modelos específicos y a protocolos cerrados.

Finalmente, en la Universidad del Cauca, en el grupo de investigación y desarrollo en tecnologías de la Información (GTI) y el grupo de investigación en ingeniería telemática (GIT), han trabajado en la creación de escenarios de interacción semántica en la IoT [71], proponiendo una arquitectura y modelo semántico para permitir la creación de servicios de interacción y la conectividad de cualquier dispositivo de la WoT. Además, derivado del trabajo anteriormente mencionado, se han desarrollados proyectos que han desplegado un escenario de interacción semántica de OI en la WoT [14], donde se implementó el modelo propuesto a manera de prueba de concepto, demostrando la factibilidad de la solución. Aunque estos proyectos se encuentran finalizados, han surgido nuevos retos por resolver como: la formalización en la implementación de los entornos de interoperabilidad semántica entre OI de la WoT y aprovechar el potencial de las ontologías que pueden ser gestionadas directamente por los OI.

3.2.2 Agrupación trabajos relacionados por conceptos

En la Figura 8 podemos ver como se organizan los principales trabajos con respecto a tres conceptos principales

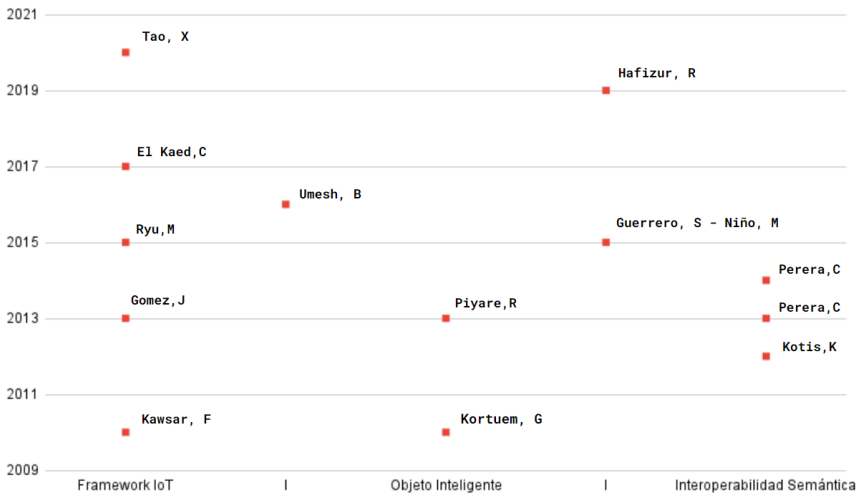


Figura 8. Relación trabajos vs. Años. Fuente: Propia

- Framework IoT. Estos artículos son las más relacionados con el actual proyecto, en general proponen herramientas para facilitar la construcción de Framework IoT, enfocándose en resolver la complejidad de algún elemento en particular, como OI o interoperabilidad. Sin embargo, podemos derivar que los principales problemas que presentan son: la falta de unificación con herramientas semánticas y la baja utilización de las capacidades de los OI. Estas brechas separan a estos proyectos de ser un framework de interoperabilidad semántica para OI

- Objeto inteligente. Estos artículos entregan elementos para definir los alcances de un OI, además muestran como descentralizar los procesos de los entornos y cargar la lógica a los OI. Sin embargo, la principal brecha de estos proyectos es que los OI no utiliza herramientas semánticas y su interoperabilidad solo es física y/o sintáctica.

- Interoperabilidad semántica. Estas propuestas manejan herramientas semánticas fundamentales, principalmente ontologías y contextos, además proponen entornos donde objetos realizan interoperabilidad semántica. Sin embargo, estas herramientas están centralizadas en servidores o en aplicaciones en la nube, esta brecha causa una baja utilización de las capacidades de los OI.

4 CONSTRUCCIÓN DEL FRAMEWORK

Este capítulo presenta todas las etapas necesarias para construir el Framework de interoperabilidad semántica de OI para la WoT. La implementación del Framework se realizó acorde a la metodología seleccionada [72], la cual nos indica los principales principios que se deben tener en cuenta para la construcción de un Framework.

De esta forma esta construcción consiste en dos etapas fundamentales. Primero, el desarrollo de un modelo que soporte la selección de herramienta y la forma en como estas interactúan, aquí podemos encontrar la definición de una arquitectura de soporte y la definición formal del Framework. Segundo, el desarrollo de una librería en un lenguaje específico, aquí podemos encontrar el repositorio de código y manuales de usuario.

4.1 ARQUITECTURA SOPORTE

La arquitectura presentada toma como base de la arquitectura propuesta por los proyectos de [71, 14]. La presente representa de forma más específica los elementos y herramientas utilizados en un escenario de interoperabilidad semántica donde encontramos OI que utilizan herramientas de la WoT.

La arquitectura presenta capas bien definidas, es orientada a servicios y guiada mediante un modelo semántico de interoperabilidad en la WoT. Además, está enfocada en la construcción de los OI y la interacción entre estos. Las principales características son:

- Distribución. El procesamiento de la información se encuentra descentralizada en los bordes de la red ("*fog computing"* [73]) reduciendo requerimientos en la comunicación y el tiempo de respuesta, como también la disponibilidad de los servicios o aplicaciones.

- Reutilización. Las aplicaciones y servicios que pueden ser construidos bajo esta propuesta son diversos, dado que en definición los elementos no están acoplados a una entidad única, entorno o propósito particular y sus relaciones dependerán de los objetivos del usuario en un momento dado.

- Generación de servicios. Permiten crear redes de objetos con el fin de cooperar en la creación de nuevos servicios, participando los OI en objetivos diferentes para lo que originalmente fueron creados.

4.1.1 Conceptos

A continuación, se definen los conceptos sobre los que se fundamenta la propuesta arquitectónica

Mundo físico

Esta investigación comparte el concepto dado por [74] en la que definen el mundo real como "… *el entorno físico que es instrumentado a través de máquinas identificadas a través de*

etiquetas, sensores, actuadores y elementos de procesamiento organizados en islas de dominios específicos, con el fin de monitorear e interactuar con las entidades físicas en las que estamos interesados". A los elementos del mundo físico se les conoce como representaciones físicas.

Mundo Digital.
Se refiere a la información que se almacena digitalmente acerca del mundo físico mediado por tecnologías de la IoT. A la información sobre un elemento en el mundo digital se le conoce como representación digital.

Recursos.
Representaciones digitales de los instrumentos que poseen los objetos para interactuar con el mundo físico. Se relacionan a entidades del mundo físico con el fin de capturar información, procesarla y comunicarla, además tienen la capacidad de cambiar algún estado de dicha entidad. Un recurso puede ser:

- Sensor**,** examina el entorno del mundo físico y capta la información que entra al sistema

- Controlador**,** procesa la información extraída del mundo real y gestiona los recursos del objeto

- Actuador, efectúa una acción sobre el mundo físico, con el fin de cambiar el estado de la entidad

Entidades*.*
Son representaciones de personas, lugares y cosas del mundo real. Una entidad necesariamente está relacionada con uno o más objetos. La entidad pasa a ser una entidad de interés cuando es objeto de búsqueda o interacción por parte de los usuarios

Propiedades
Corresponde las características de las entidades que pueden describir el estado en un momento dado. La propiedad pasa a ser una propiedad de interés cuando es objeto de búsqueda o interacción por parte de los usuarios

Metadatos
Define al OI y sus recursos por medio de un archivo JSON, en donde se asocia al objeto con su nombre, identificación, fechas de creación y edición, entre otros. Cada recurso igualmente está definido en este archivo y tiene asociados información sobre su nombre, tipo de recurso, direcciones url para consumir sus servicios, entre otros. Un ejemplo del documento JSON de metadatos lo podemos ver en la Figura 9

```
{
    "id": "708637323",
    "name": "Regulador de temperatura",
    "description": "Regulador de temperatura conectado a la web",
    "created": "2018-11-14T19:10:23.824Z",
    "modified": "2019-06-01T09:12:43.124Z",
    "properties": {
        "temperature": {
            "name": "Sensor de temperatura",
            "description": "Medida de temperatura de la estación",
            "type": "number",
            "unit": "ºC",
            "forms": [
                {
                    "op": "readproperty",
                    "protocol": "mqtt",
                    "href": "mqtt://domain:port/project_name/topoic"
                },
                {
                    "op": "readproperty",
                    "protocol": "coap",
                    "href": "coap://domain:port/project_name/lightSensor",
                    "methodName": "GET"
                },
                {
                    "op": "writeproperty",
                    "protocol": "coap",
                    "href": "coap://domain:port/project_name/lightSensor",
                    "methodName": "PUT"
                }
            ]
        },
        "heater": {···
        },
        "fan": {···
        }
    }
}
```


Figura 9. Ejemplo archivo metadatos. Fuente: Propia

Objeto semántico

Un objeto semántico es en base un OI que tiene una representación digital (identificador único y mecanismo para recuperarlo), está asociado a una entidad de interés específica, está definido por metadatos y utiliza un índice semántico para almacenar el conocimiento de su contexto e interacciones con otros OI.

Servicio básico

Son los servicios expuestos al usuario que presentan los OI por defecto, están asociados al funcionamiento original con el cual el OI fue creado. Cada OI debe tener por lo menos un servicio básico asociado.

Servicio de interacción

Es generado por la interacción entre dos objetos. Está asociado a una entidad de interés, a la propiedad que desencadena una acción y a la propiedad que se desea modificar. Cada servicio de interacción está asociado a un ECA y es ejecutado por el OI que contiene el recurso actuador que puede modificar la propiedad deseada.

Usuarios

Personas o aplicaciones que interactúan con los OI definidos en una entidad de interés especifica. Los usuarios tienen una experiencia natural y transparente con el escenario.

Escenario de interoperabilidad semántica
Entorno físico mediado con herramientas semánticas donde los OI ofrecen sus recursos y funcionalidades e interactúan de forma dinámica con otros OI y los usuarios. En este espacio se pueden identificar las entidades de interés, de las cuales se quiere conocer su estado midiendo las propiedades de interés.

Coordinador
Aplicación utilizada por el usuario como interfaz para comunicarse con el escenario. Recoge la información de los distintos elementos del escenario por medio de servicios web especializados en IoT.

ECA (evento - condición - acción)
Archivo JSON el cual define la interacción entre dos OI en una misma entidad de interés permitiendo generar nuevos servicios (servicios de interacción) Se compone de tres partes

- Evento, propiedad de interés captada por un recurso. El OI asociado al recurso seleccionado para el evento es denominado OI evento

- Condición, sentencia condicional (mayor, menor o igual) que compara el estado del recurso seleccionado en el evento (propiedad de interés) con un valor ingresado por el usuario. Ambas variables se sincronizan acorde a los metadatos recuperados del OI evento

- Acción, sentencia condicional (mayor, menor o igual) que compara el estado de un recurso de tipo actuador con un valor ingresado por el usuario. El OI asociado al recurso seleccionado es denominado OI acción. Ambas variables se sincronizan acorde a los metadatos recuperados del objeto acción.

El ECA tiene asociado un nombre, un estado (prendido o apagado) y una entidad de interés, así los OI pueden tener varios ECA que ejecuten un mismo recurso, sin que esto signifique un problema de conflicto. Sin embargo, queda a disposición de la coherencia del usuario el proponer sentencias que desaten un conflicto. Un ejemplo del documento JSON que representa el ECA lo podemos encontrar en la Figura 10

```
{
    "id": 0,
    "name": "name",
    "description": "description",
    "created" : "2018-11-14T19:10:23.824Z",
    "modified" : "2019-06-01T09:12:43.124Z",
    "properties": {
        "state": {
            "type": "boolean",
            "const": 0
        }
    },
    "actions": {
        "name": {
            "input": {
                "type": "object",
                "properties": {
                    "event": {
                        "type": "number",
                        "unit": "ºC",
                        "forms": [
                            {
                                "op": "readproperty",
                                "protocol": "mqtt",
                                "href": "mqtt://domain:port/topoic"
                            },
                            {
                                "op": "readproperty",
                                "protocol": "coap",
                                "href": "coap://domain:port/lightSensor
                                "methodName": "GET"
                            }
                        ]
                    },
                    "condition": {
                        "type": "object",
                        "properties": {
                            "operator": {
                                "type": "string",
                                "const": ">"
                            },
                            "value": {
                                "const": 10
                            }
                        }
                    }
                }
            },
            "output": {
                "type": "boolean",
                "const": 1,
                "forms": [
                    {
                        "op": "writeproperty",
                        "protocol": "coap",
                        "href": "coap://domain:port/lightSen
                        "methodName": "PUT"
                    }
                ]
            }
        }
    }
}
```


Figura 10. Ejemplo archivo ECA. Fuente: Propia

4.1.2 Vista estática

A continuación, se presenta la Figura 11, que distingue las distintas capas de la arquitectura propuesta para el soporte del Framework.

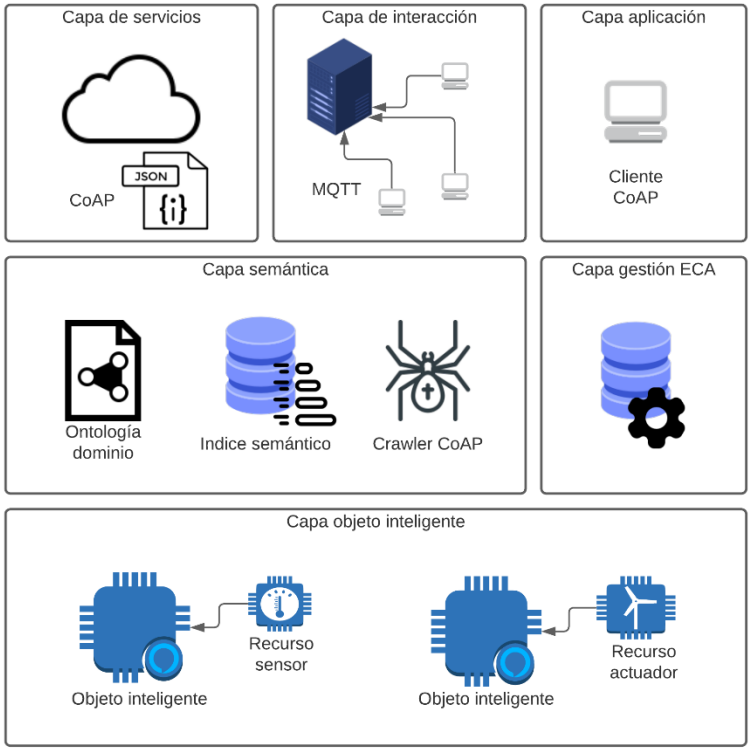


Figura 11. Vista estática de la arquitectura. Fuente: Propia

Capa objeto inteligente

Se encuentra la representación física de los OI y se definen los recursos asociados. Esta es la base de la arquitectura ya que cada OI contiene las herramientas expresadas en las capas superiores. Esta capa se conecta con la capa semántica, debido a que cada OI puede almacenar y ejecutar las herramientas semánticas, además se conecta con la capa de gestión ECA debido a que cada OI contiene la capacidad de almacenar y ejecutar las unidades de interoperabilidad ECA.

Capa semántica

Encargada de alojar las herramientas y el motor semántico de cada OI. Se diferencian 3 elementos:

- Crawler CoAP, basado en el comportamiento de un Crawler web, este elemento está dedicado a buscar información sobre cada OI en la red de un escenario, y utilizando la información recuperada aumenta el dominio de destinos de búsqueda. El protocolo sobre el cual está realizando la búsqueda es CoAP

- Ontología de dominio, utilizada por el índice para crear el contexto a partir de los conceptos y relaciones. Esta ontología contiene los conceptos referentes a un sistema, y en general, conceptualiza y recoge los elementos pertenecientes a dicho sistema

- Índice semántico, elemento que contextualiza los OI según los conceptos que tengan asociados. Para esto el índice realiza un barrido de todos los conceptos y sus relaciones de la ontología de dominio y los compara con la información recuperada en cada OI alcanzado por el Crawler. A la información obtenida de cada OI se le realiza un preprocesamiento donde se obtiene como resultado un listado de términos, cada término es filtrado utilizando la distancia semántica entre el término y los conceptos de la ontología de dominio, si la distancia resulta ser menor a un canon (filtrado semántico) este término será incluido en el índice, y si el término se repite, será sumado al peso que tiene el término en el índice. El índice al final, es una matriz en donde se relacionan los OI con los listados de términos, cuya intersección es el peso que tiene el término con respecto al OI.

Capa gestión ECA

Encargada de almacenar y ejecutar las unidades de interoperabilidad ECA. Cada ECA generado en el escenario debe ser persistido en cada OI el cual encuentre sus recursos involucrados en el servicio de interoperabilidad. Cada OI es responsable de ejecutar todos los ECA que tenga guardados y activos. Para ejecutar el ECA el OI utiliza la capa de interacción, en donde puede consumir el estado del recurso.

Capa de servicio

Encargado de almacenar, encapsularla y exponer los servicios del OI a través de servicios web, haciendo posible que las aplicaciones y otros OI puedan consumir la información expuesta por cada OI de manera interoperable y transparente. Expone los servicios

basados en CoAP y encapsula los mensajes en elementos JSON. Se define los siguientes grupos de servicios:

- Servicios para metadatos, métodos expuestos para realizar consultas y modificaciones de los metadatos de un OI.

- Servicio para ECA, métodos expuestos para realizar consultas y modificaciones sobre las unidades de interoperabilidad ECA que guarda cada OI.

- Servicio para índice semántico, métodos expuestos para consultar las listas generadas por el índice semántico.

Capa de interacción
Soporta las herramientas que permiten comunicar a los OI con otros OI. Esta capa está basada en el enfoque MQTT por el cual los OI pueden comunicar los estados de sus recursos. Las interacciones realizadas sobre esta capa son:

- OI como cliente publicador. El OI comparte de forma desatendida y periódica el estado sus recursos en diferentes tópicos (un tópico diferente por recurso) en un servidor central (Broker o Servidor MQTT)

- OI como cliente suscriptor. Cuando el OI necesita consultar el estado de otro OI, se suscribe al tópico expuesto para dicho recurso en un servidor central

Capa de aplicaciones
Permite al sistema integrar aplicaciones terceras. Esta capa puede alojar clientes que sigan el protocolo CoAP y sirve como una ventana para administrar los OI. Como ejemplo, es posible utilizar una aplicación móvil como interfaz al usuario para administrar los OI del escenario. Esta aplicación puede consumir los métodos expuestos por el OI en la capa de servicios, tanto para consultar como para modificar información de cada OI

4.1.3 Vista dinámica. Enfoque como usuario

A continuación, se presenta la vista dinámica de la arquitectura vista como un usuario. Esta se divide en 3 partes diferenciadas por los tipos de servicios que ofrece la capa de servicios e inicializadas por un usuario a través de una aplicación externa.

Gestionar metadatos
El proceso inicia cuando el usuario selecciona una aplicación compatible con CoAP y genera una consulta sobre algún método que recae en los servicios expuestos para metadatos. Para un caso de ejemplo, se realiza una consulta para obtener los metadatos de un OI determinado. Cuando la capa de servicio recibe esta petición consulta sobre la base de persistencia del OI y genera un elemento JSON, el cual será utilizado para dar respuesta a la petición. La Figura 12 ilustra dicho proceso

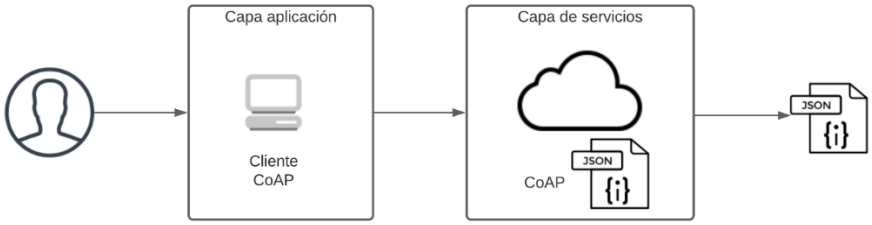


Figura 12. Proceso gestionar metadatos. Fuente: Propia

Gestionar ECA

El proceso inicia cuando el usuario selecciona una aplicación compatible con CoAP y genera una consulta sobre algún método que recae en los servicios expuestos para las unidades de interoperabilidad ECA. Para un caso de ejemplo, se realiza una consulta para modificar el estado de activación de un ECA (prender o apagar el ECA). Cuando la capa de servicio recibe esta petición consulta sobre la base de datos donde se encuentran persistidos todos los ECA del OI, selecciona el ECA requerido y cambia el registro de su estado. Inmediatamente la lógica que maneja la ejecución del ECA detecta el estado y detiene la ejecución de esta tarea. La Figura 13 ilustra dicho proceso.

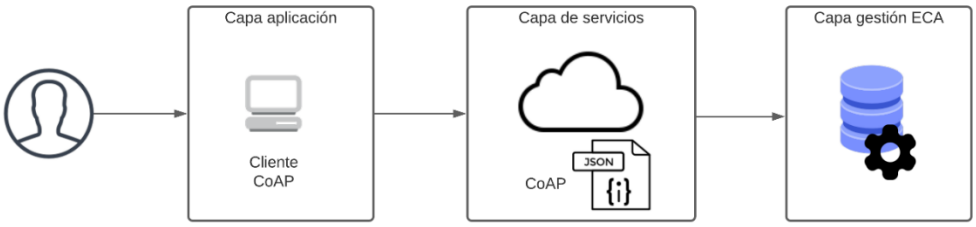


Figura 13. Proceso gestionar ECA. Fuente: Propia

Consulta índice semántico

El proceso inicia cuando el usuario selecciona una aplicación compatible con CoAP y genera una consulta sobre el índice semántico. Esta consulta específicamente recibe un texto libre y entrega un listado de los OI relacionados al contexto relacionado con la ontología de dominio. Cuando la capa de servicio recibe la petición, consulta a la capa semántica sobre el listado relacionado con el texto ingresado, a esta petición la capa semántica realiza los siguientes pasos:

- Pre-procesamiento del texto de entrada. El texto es procesado para tener un arreglo de términos ordenados, los cuales deben tener un valor descriptivo. En este caso se eliminan palabras que no agregan mucho significado (eliminación de *stop words*), se normalizan lo términos que gramaticalmente son nombres propios a su forma singular y se eliminan términos repetidos.

- Filtrado semántico. Cada término del arreglo, es comparado con la ontología de dominio utilizando un proceso de distancia semántica. Si el valor resultado de la distancia semántica entre el término es mayor a un canon prestablecido, este término será desechado y se considera lejano al contexto de la ontología de dominio

- Obtener primer listado. Un listado primitivo de los OI es consultado en el índice semántico. Este listado representa todos los OI contenidos en las filas de la matriz que representa al índice, pero no está ponderado por ningún criterio.

- Obtener listado final. En la matriz que representa al índice, los términos son las columnas, los OI son las filas y la intersección entre filas y columnas es el peso del término en el OI, el cual es un valor numérico. Para cada OI del primer listado se suman los pesos de todos los términos que no fueron desechados en la etapa del filtrado semántico y al final se ordena el primer listado de acuerdo al valor de la anterior sumatoria. En este punto ya tenemos el listado final.

La lista organizada de los OI es encapsulada en un documento JSON, el cual será utilizado para dar respuesta a la petición. Figura 14 ilustra dicho proceso

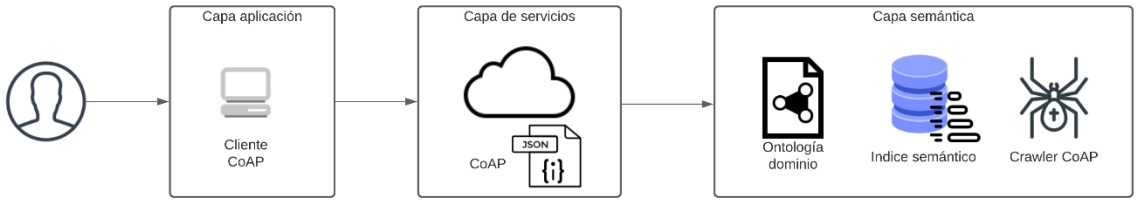


Figura 14. Proceso de consulta índice semántico. Fuente: Propia

4.1.4 Vista dinámica. Enfoque sistema interno

A continuación, se presenta la vista dinámica de la arquitectura vista desde el sistema interno, es decir los procesos que se ejecutan sin mediación de un usuario. Esta vista se divide en 2 partes diferenciadas por los tipos de servicios que ofrece la arquitectura internamente.

Ejecutar ECA

El proceso inicia cuando la capa de gestión ECA detecta un nuevo servicio de interoperabilidad ECA o el cambio del estado (prender) de uno antiguo. Esta capa se asegura de tener el recurso condición dentro de su OI y luego de esta tarea, consulta de forma periódica a la capa de interacción, sobre el estado del recurso evento. Cuando el recurso evento cumpla la condición establecida en el ECA la capa de gestión ECA se comunicará con la capa de objeto inteligente para modificar el estado del recurso acción según lo establecido en el ECA. La Figura 15 ilustra dicho proceso

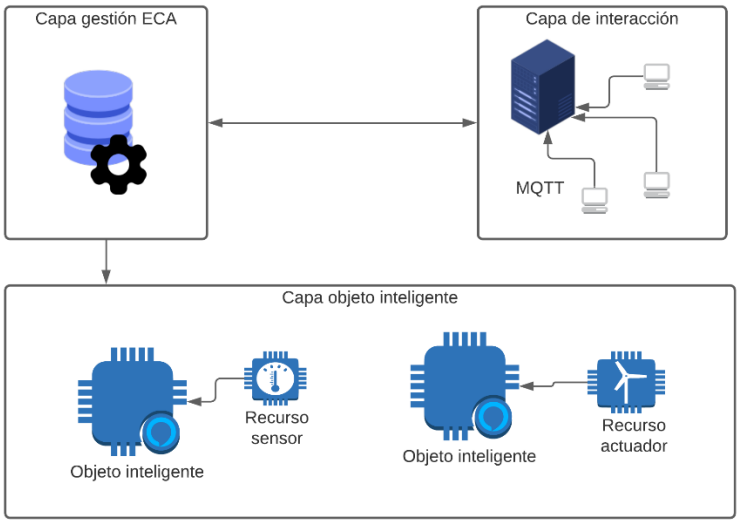


Figura 15. Proceso ejecutar ECA. Fuente: Propia

Descubrir objeto inteligente

Este proceso inicia cuando el sistema es inicializado la primera vez o cuando el Crawler CoAP cumple un ciclo de espera para el descubrimiento de nuevos OI. Para este caso tendremos dos OI diferentes, el primero el OI destino, el cual debe ser un dispositivo ubicado en una red al alcance del Crawler CoAP, o el OI a descubrir y el segundo, el OI madre, el cual es el administrador del Crawler CoAP y la capa semántica que realizará el procesamiento. Primero el Crawler CoAP consulta sobre los metadatos del OI destino, esto lo realiza en la capa de servicios de este OI, la cual responde con sus metadatos en un documento JSON. Luego la información de los metadatos del OI destino será entregada a la capa semántica del OI madre. Finalmente, dentro de la capa semántica se realizan los siguientes pasos:

- Análisis documento JSON. El documento es analizado para encontrar los apartados donde se guardan los títulos, las descripciones y las etiquetas, con estos datos se genera una base descriptiva compilada en un texto del OI destino.

- Preprocesamiento de metadatos. El texto que describe al OI destino es procesado para tener un arreglo de términos ordenados, los cuales deben tener un valor descriptivo. En este caso se eliminan palabras que no agregan mucho significado (eliminación de *stop words*) y se normalizan lo términos que gramaticalmente son nombres propios a su forma singular.

- Filtrado semántico. Cada término del arreglo, es comparado con la ontología de dominio utilizando un proceso de distancia semántica. Si el valor resultado de la distancia semántica entre el término es mayor a un canon prestablecido, este término será desechado y se considera lejano al contexto de la ontología de dominio

- Indexación términos. Los términos que no fueron desechados en la anterior etapa, son indexados a la matriz que representa el índice. Si el OI es nuevo en la matriz se genera

una fila y si el término es nuevo en la matriz se genera una nueva columna y se pondrá una unidad en la intersección con la fila del respectivo OI. Si el OI y el término ya existen, en la intersección entre el OI y el término se sumará una unidad. En este sentido la intersección entre la fila de la matriz (OI destino) y la columna de la matriz (término) representa el peso del término en el OI.

Los anteriores pasos actualizan la matriz que representa al índice y el Crawler CoAP realiza la búsqueda de nuevos OI utilizando las url encontradas en los metadatos del OI destino. La Figura 16 ilustra dicho proceso.

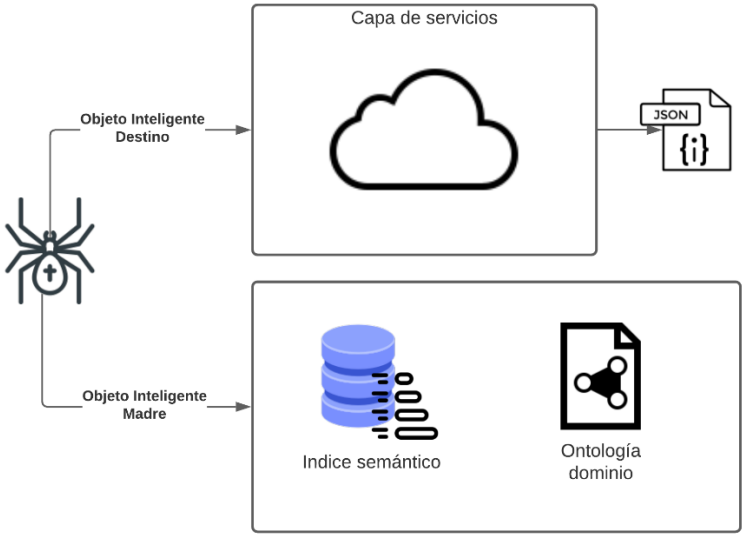


Figura 16. Proceso descubrir objeto inteligente. Fuente: Propia

4.2 FRAMEWORK

Este apartado presenta todos los elementos que componen al Framework y como estos se organizan para conformar una librería y una serie de comandos de un alto nivel de abstracción, los cuales son expuestos a los usuarios finales o usuarios desarrolladores.

El desarrollo del Framework está fuertemente basado en la arquitectura de soporte presentada en al anterior apartado, por lo tanto, hereda todos sus conceptos y utiliza las capas presentadas en la arquitectura para conformar los elementos de la librería. En base a la vista estática de la arquitectura, el Framework abstrae y automatiza la mayor parte de la implementación de los elementos expuestos en las capas. Con respecto a la vista dinámica de la arquitectura, el Framework entrega un archivo de configuración y comandos para que el usuario final o usuario desarrollador pueda interactuar con los elementos del Framework.

El Framework a vista del usuario desarrollador se divide esencialmente en dos partes principales. El archivo de configuración, el cual contiene las directrices para regular el comportamiento de la aplicación deseada, y los comandos de ejecución, los cuales exponen acciones de gestión del Framework.

En el Anexo C podemos encontrar los manuales de usuario del Framework y las instrucciones para acceder al repositorio del código.

4.2.1 Vista general

A continuación, se presenta la Figura 17, donde se muestran los distintos elementos que conforman al Framework. En esta podemos diferenciar cuatro partes fundamentales

- Elementos base (color violeta) Estos elementos corresponden a las tecnologías sobre las cuales está construido el Framework, por lo tanto, estás enmarcan todo el desarrollo tanto de la librería como la implementación de aplicaciones basadas en el Framework

- Elementos centrales (color amarillo) Estos elementos corresponden a los componentes internos que estructuran el Framework, al final son archivos organizados que exponen clases y funciones, los cuales resuelven los requerimientos de las capas presentadas en la arquitectura de soporte. Estos elementos fueron construidos sobre las tecnologías de los elementos base.

- Elemento de abstracción (color azul) Este elemento corresponde a la abstracción de los elementos centrales, se podría decir que corresponde al Framework como tal. Este expone las configuraciones de sus diferentes elementos y comandos los cuales son la interfaz para que el usuario final genere aplicaciones.

- Elementos de aplicación (color verde) Estos elementos corresponden a la capa de aplicación expuesta en la arquitectura. Como nos muestra la Figura este grupo de elementos está se soporta en el elemento Framework, debido a que en la práctica

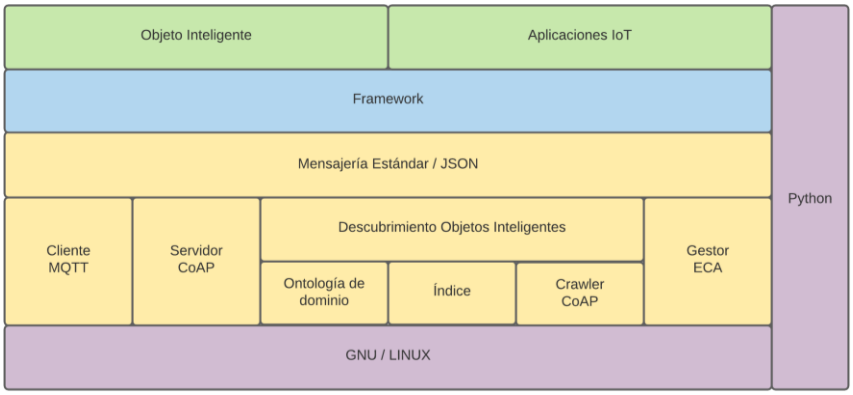


Figura 17. Vista general del Framework. Fuente: Propia

Elemento GNU/LINUX

Este elemento está ubicado en la base de la estructura, y representa el sistema operativo sobre el cual está construido el Framework y, por lo tanto, es la base estructural y un requerimiento principal para la construcción de todas las aplicaciones desarrolladas utilizando el Framework. La selección de esta tecnología fue debido a que GNU/LINUX es completamente abierto y utilizar este sistema operativo no limita los desarrollos, además es altamente utilizado y documentado en el campo de la IoT.

Elemento Python

Este elemento se encuentra ubicado de forma horizontal, esto significa que influye de forma transversal en los otros elementos del Framework. Este elemento representa el lenguaje de programación utilizado para desarrollar el Framework y, por lo tanto, es la base programática y un requerimiento principal para la construcción de todas las aplicaciones desarrolladas utilizando el Framework. La selección de esta tecnología fue debido a que Python es un lenguaje de programación altamente versátil y permite implementar software que maneje elementos hardware hasta aplicaciones web, además Python es altamente utilizado y documentado en el campo de la IoT.

Elemento Objeto Inteligente

Este elemento se encuentra ubicado en la parte superior de la estructura, y representa un tipo de aplicación la cual se puede implementar utilizando el Framework. Aunque estos elementos no hacen parte interna del Framework, fue ubicado en la estructura debido a que las aplicaciones son consideradas el desenlace del Framework y se encuentran enlazadas con este a nivel de código. Estas aplicaciones deben estar construidas sobre el sistema operativo GNU/LINUX y el lenguaje de programación Python.

Elemento Aplicación IoT

Este elemento se encuentra ubicado en la parte superior de la estructura, y representa cualquier aplicación que sea implementada utilizando el Framework. Aunque desarrollar la aplicación Objeto Inteligente es el principal objetivo por el cual el Framework fue construido, en el transcurso del presente proyecto fue descubierto que es posible desarrollar diferentes tipos de aplicaciones enfocadas en IoT utilizando el Framework. Estas aplicaciones deben estar construidas sobre el sistema operativo GNU/LINUX y el lenguaje de programación Python.

Elemento Framework

Este elemento se encuentra ubicado en el centro de la estructura, y representa la abstracción del Framework, se puede entender como la ventana por la cual el usuario final o usuario desarrollador interactúa con los elementos del Framework para generar aplicaciones. En la parte de abajo tiene todos los elementos que lo conforman y funcionan como puente para comunicarlo con el sistema operativo, entendiendo que el Framework expone un mayor nivel de abstracción. Este elemento expone a los usuarios finales las siguientes características:

- Archivo de configuraciones en formato JSON, el cual contiene los parámetros de inicialización y calibración de los diferentes elementos que conforman el Framework

- Archivo simplificado de metadas en formato JSON, el cual contiene de forma simplificada la información del OI, como el nombre, descripción y los datos de sus recursos

- Comando generar metadatos. Este comando es utilizado para generar y persistir los metadatos del OI. Utilizando el archivo simplificado de metadatos (descrito en el punto anterior) el Framework genera un nuevo archivo JSON basado en la recomendación de la W3C *WOT Thing Description,* este archivo será utilizado por el Framework para compartir sus metadatos. Esto es transparente para el usuario final.

- Comando iniciar objeto inteligente. Este comando es utilizado para crear todos los archivos y los organiza para poder iniciar con la creación de las aplicaciones dentro de un OI. En la Figura 18 podemos ver el sistema de carpetas y de archivos creados. La carpeta “app” es donde el usuario desarrollador puede generar sus aplicaciones, sin embargo, por defecto son creados dentro de esta carpeta, archivos dependiendo de los recursos asignados en los metadatos al OI, cada archivo muestra de forma básica como hacer el llamado de las funciones auxiliares del Framework. La carpeta “generated” contiene archivos necesarios para que el Framework realice las conexiones a la persistencia de los estados de cada recurso, la persistencia de los ECA y la persistencia de los metadatos, además contiene archivos auxiliares para manejar cada recurso del OI.

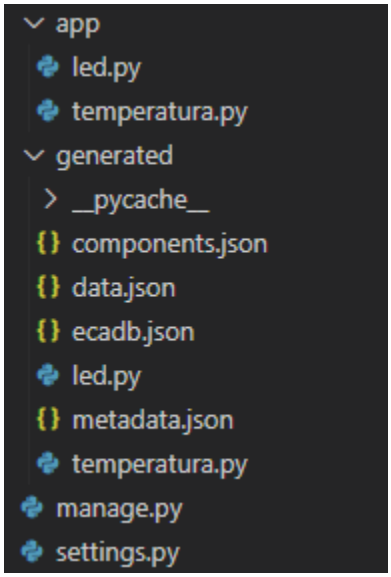


Figura 18. Carpetas y archivos de un OI. Fuente: Propia

Elemento Mensajería Estándar/JSON

Este elemento se encuentra ubicado en la parte central de la estructura, y representa el proceso de encapsular los archivos que describen al OI, el archivo de los metadatos y el archivo de los ECA. Este proceso se realiza con el fin de compartir de forma estandarizada

la información del OI a otros OI o entre los elementos internos, para esto se utilizó la recomendación *Thing Description* de la W3C. Este elemento se encuentra dentro del grupo de elementos centrales del Framework, y afecta de forma transversal a los demás elementos de este grupo. Se encuentra ubicado en la parte inferior con respecto al elemento Framework debido a que la administración directa de estos archivos el usuario final o usuario desarrollador la realiza utilizando el Framework.

Elemento Cliente MQTT

Este elemento se encuentra ubicado en la parte central de la estructura, y representa la gestión de los clientes MQTT. Este elemento se encuentra basado en la capa de interacción propuesto en la arquitectura de soporte, por lo tanto, este elemento fue construido basado en las definiciones expresadas en esta capa. Para este elemento se deben tener en cuenta los siguientes apartados:

- Cada recurso definido en los metadatos del OI tendrán un canal de publicación único, el cual publicará el estado de dicho recurso periódicamente. La frecuencia de publicación y el servidor MQTT serán seleccionados en el documento de configuración.

- El usuario desarrollador tiene la autonomía de prender o apagar el elemento cliente MQTT de cada recurso utilizando el documento de configuración

- La ruta para consumir el canal donde publica el estado cada recurso de un OI puede ser encontrada en los metadatos en forma de URL

Elemento Servidor CoAP

Este elemento se encuentra ubicado en la parte central de la estructura, y representa los servicios expuestos del OI a la red por medio de CoAP. Este elemento se encuentra basado en la capa de servicios propuesta en la arquitectura de soporte, por lo tanto, los servicios fueron construidos homologando los métodos propuestos en esta capa, sin embargo, por defecto el Framework limita los servicios que expone a los siguientes casos:

- Leer metadatos. Este método se encuentra dentro de los servicios para metadatos expuesto en la capa de servicios de la arquitectura soporte. Para este caso la persistencia de los metadatos se realiza sobre un documento JSON construido bajo la recomendación *Thing Description* de la W3C y se expone a la red en una ruta con la siguiente nomenclatura "*host/metadata/get*" bajo el método GET de CoAP

- Nuevo ECA. Este método se encuentra dentro de los servicios para ECA expuesto en la capa de servicios de la arquitectura soporte. Este expone a la red una ruta con la siguiente nomenclatura "*host/eca/new*" bajo el método POST de CoAP, el método espera recibir un JSON con la información básica para construir el ECA, como recurso OI evento, recurso OI acción, valor de la condición y valor del nuevo estado del recurso del OI acción

- Prender/Apagar ECA. Este método se encuentra dentro de los servicios para ECA expuesto en la capa de servicios de la arquitectura de soporte. Este expone a la red una ruta con la siguiente nomenclatura "*host/eca/state*" bajo el método PUT de CoAP, el método espera recibir un valor con la información para cambiar el estado de un ECA, como identificador del ECA y el valor del estado deseado

- Obtener listado OI (contexto). Este método se encuentra dentro de los servicios para índice somático expuesto en la capa de servicios de la arquitectura de soporte. Este expone a la red una ruta con la siguiente nomenclatura "*host/context/get*" bajo el método POST de CoAP, el método espera recibir un texto y responde con un listado de OI ponderados según el peso de los términos entregados en el texto. Para entender más detalladamente este proceso consultar el apartado Consulta índice semántico. Que se encuentra dentro de "Vista dinámica. Enfoque como usuario" de la sección Arquitectura de Soporte del presente capítulo.

Elemento Descubrimiento Objetos Inteligentes

Este elemento se encuentra ubicado en la parte central de la estructura, y representa la unión de tres elementos diferentes, los cuales tienen tareas bien definidas dentro del Framework, pero la unión de sus resultados la podemos definir como el descubrimiento de objetos inteligentes. Este elemento está relacionado con la capa semántica de la arquitectura de soporte, por lo tanto, el flujo trabajo del descubrimiento hereda de esta capa y define los siguientes pasos:

- Inicialización. El elemento Crawler CoAP inicia con la búsqueda de nuevos OI en la red, esto la hace periódicamente, el producto de esta etapa es un documento JSON que contiene los metadatos del nuevo OI

- Análisis. El documento JSON es analizado para encontrar términos conceptos y oraciones que describan el nuevo OI.

- Preprocesamiento. El texto es procesado para tener un arreglo ordenado y homogéneo de términos.

- Filtrado semántico. Los términos son aceptados según su relación con el contexto del OI, esto se hace utilizando cálculos de similitud semántica contra la ontología de dominio.

- Indexación términos. Los términos aceptaros en la anterior etapa son agregados al índice y sumados en su peso con respecto al nuevo OI

Por otra parte, el elemento para descubrimiento de OI contiene las siguientes configuraciones:

- Valor aceptado para distancia semántica. Para el cálculo de la distancia semántica o en el caso del Framework la similitud semántica, se utiliza la fórmula de fórmula de Wu & Palmer, la cual compara dos términos y calcula la similitud entre estos arrojando un valor entre 0 y 1. El Framework permite la parametrización del valor aceptado para que esta similitud semántica acepte o rechace un término o concepto. Este ajuste del valor entre 0 y 1 que puede ser configurado por el usuario desarrollador, representa la apertura y flexibilidad del contexto que menaje el OI.

- Tipo de ontología de dominio a utilizar. El Framework soporta diversas ontologías de dominio que estén construidas con OWL. El usuario desarrollador puede cargar en el Framework múltiples ontologías de dominio y en el archivo de configuración seleccionar una de ellas, para que represente el contexto del OI.

Elemento Ontología de Dominio

Este elemento se encuentra ubicado en la parte central de la estructura, y representa la gestión de la ontología de dominio seleccionada para generar el contexto del OI. Este está directamente relacionado con su semejante en nombre ubicado en la capa semántica de la arquitectura de soporte, por lo tanto, la interoperabilidad y funcionamiento de este elemento es heredado de esta capa. Por defecto la ontología de dominio que entrega el Framework es DogOnt [75], aunque el usuario desarrollador tiene la libertad de modificar esta ontología.

La ontología de dominio es consultada cada vez que el elemento para descubrimiento de OI necesita realizar un filtrado de términos. Las consultas a este elemento se realizan con SPARQL [76] y son las siguientes:

- Consulta clases. Esta consulta obtiene todas las clases existentes en la ontología de dominio sin tener en cuenta su relacionamiento interno. Esta consulta es utilizada para un filtrado primitivo, en donde se listan las clases de la ontología de dominio y se las compara con un término para encontrar similitudes exactas.

- Consulta clase con relaciones. Esta consulta obtiene la información de una sola clase, sus relaciones con otras clases y el tipo de relación. Esta consulta es utilizada para el filtrado semántico, en donde se compara un término con la clase y se busca filtrarlo por medio de la similitud semántica.

Elemento Índice

Este elemento se encuentra ubicado en la parte central de la estructura, y representa el sistema de indexación de los OI. Está directamente relacionado con el componente índice semántico ubicado en la capa semántica de la arquitectura de soporte, sin embargo, el presente elemento a diferencia de su paralelo en la arquitectura solo es un índice, debido a que, en este nivel dentro de la estructura, no representa un contexto por si solo y básicamente este elemento es una matriz invertida. La Figura 19 muestra cómo se conforma la matriz.

	término 1	término 2	término 3	término 4
Objeto Inteligente 1	peso	peso	peso	peso
Objeto Inteligente 2	peso	peso	peso	peso
Objeto Inteligente 3	peso	peso	peso	peso

Figura 19. Matriz invertida. Fuente: Propia

Se pueden diferenciar la matriz en las siguientes partes:

- Columna de OI. Cada elemento de esta columna representa un OI y contiene un identificador alfanumérico

- Fila términos. Cada elemento de esta fila representa un concepto y contiene una palabra

- Pesos. La intersección entre un OI y un término representa el peso que tiene el concepto en el OI. Este elemento contiene un número natural incluyendo el 0 (para términos que no tienen peso en el OI)

El índice se convierte en un índice semántico al momento de interoperar con otros elementos como la ontología de dominio y el elemento Crawler CoAP, por esta razón se habla de índice semántico al nivel del elemento descubrimiento de OI.

Elemento Crawler CoAP

Este elemento se encuentra ubicado en la parte central de la estructura, y representa el agente dedicado a buscar nuevos OI de una red o escenario. Al igual que el componente ubicado en la capa semántica de la arquitectura de soporte, este elemento funciona de forma similar a un Crawler Web y su finalidad es obtener los metadatos de otros OI para que estos sean indexados y hagan parte del contexto de un OI. Las características de este elemento se listas a continuación

- El protocolo para realizar las consultas sobre los metadatos es CoAP, específicamente busca sobre las rutas que contengan la palabra "*metadata*" y realiza una petición GET esperando obtener un documento JSON bajo la recomendación *Thing Description* de la W3C

- Realiza un análisis sobre los metadatos y separa los elementos descriptivos de las direcciones o rutas (*URL*). Los elementos descriptivos como títulos, nombres, descripciones, etiquetas son entregados al elemento de descubrimiento de OI para continuar con el proceso de indexar el OI. Los elementos *URL* son guardadas en un arreglo de rutas interno del Crawler para que, de forma recursiva, pueda buscar nuevos OI en la red.

- El Crawler realiza el proceso de búsqueda de OI en un periodo de tiempo preestablecido en el archivo de configuraciones del Framework. Este periodo es de libre elección para el usuario desarrollador y por defecto es de 1 hora. Cuando el Crawler termina un ciclo de búsqueda de OI sobre la red, entra en una etapa de espera y luego reinicia la búsqueda de nuevos OI y actualización de OI ya existentes.

- El Crawler inicia el proceso de búsqueda de OI utilizando un arreglo semilla tipo lista de valores, preestablecido en el archivo de configuraciones del Framework. Este arreglo está conformado por direcciones IP o *URL*, las cuales son las rutas para acceder a otros OI dentro de una red. Esta opción es de libre elección para el usuario desarrollador y por defecto no contienen ningún ítem dentro del arreglo.

Elemento Gestor ECA

Este elemento se encuentra ubicado en la parte central de la estructura, y representa los procesos que realiza el sistema para guardar y ejecutar los servicios de interoperabilidad ECA. Este elemento hereda las funcionalidades de forma limitada de la capa de gestión de ECA de la arquitectura soporte, contemplando los siguientes partes en los procesos:

- Persistencia de los servicios de interoperabilidad ECA. Para realizar el proceso de persistencia de los ECA el Framework aprovisiona al sistema una base de datos no relacional bajo el motor SQLITE [77] la cual organiza los ECA en elementos JSON en una base de datos ubicada en la misma infraestructura que soporta al OI.

- Filtrar ECA. Las solicitudes para le gestión de ECA son analizadas previamente en esta etapa, para asegurar que el sistema reciba únicamente los servicios que tengan relación con el OI y sus recursos, esto con el fin de no ejecutar servicios de terceros, ocupando recursos de la infraestructura del OI. Si dentro de una petición para gestionar ECA no se encuentra un recurso perteneciente al OI que lo aloja, tanto en el apartado de los eventos como las acciones, este servicio es desechado.

- Guardar ECA. Este proceso tiene como fin guardar un nuevo ECA en el sistema, para esto se tienen las siguientes etapas, considerando previamente el filtro de ECA. Primero, el elemento gestor ECA recibe un documento JSON primitivo donde se encuentran los datos principales del servicio de interoperabilidad, como recurso OI evento, recurso OI acción, valor de la condición y valor del nuevo estado del recurso del OI acción. Luego, genera un JSON extendido del ECA el cual es construido bajo la recomendación *Thing Description* de la WC3. Finalmente, con la ayuda del motor de consultas de SQLITE se inserta el JSON extendido en la base de datos.

- Prender/Apagar ECA. Este proceso tiene como fin cambiar el estado del ECA, para que este pueda ser o no se ejecutado en el sistema, para esto se tienen las siguientes etapas, considerando previamente el filtro de ECA. Primero, el elemento gestor ECA recibe una petición de cambio de estado con el identificador del ECA y el valor del estado deseado. Luego, con la ayuda del motor de consultas de SQLITE se busca el

ECA y se modifica su estado. Finalmente, se suspende la ejecución de este servicio en el sistema.

- Ejecutar ECA. Este proceso tiene como fin ejecutar los servicios de interoperabilidad ECA que se encuentran alojados en la base de datos y se encuentren activos con el estado “prendido”. El ciclo de vida de cada ECA es el siguiente. Primero, el servicio es asignado a un hilo de ejecución independiente donde ejercita el algoritmo del ECA. Segundo, del apartado “*event*” del OI evento se obtiene la ruta MQTT para hacer la suscripción en el canal al recurso evento y poder conocer su estado. La frecuencia de actualización de este dato depende de la configuración del OI evento. Tercero, el estado del recurso del OI evento es comparado con el valor de la condición expuesta en el apartado “*condition*” del servicio ECA. Quinto, cuando se cumple la condición se cambia el estado del recurso de OI acción, utilizando la ruta y método CoAP expuestos en el apartado “*action*” del servicio ECA. De esta forma, el ciclo de vida del ECA termina cuando se cumple la condición y se cambia el estado del recurso del OI acción. En la Figura 20 podemos ver esta secuencia.

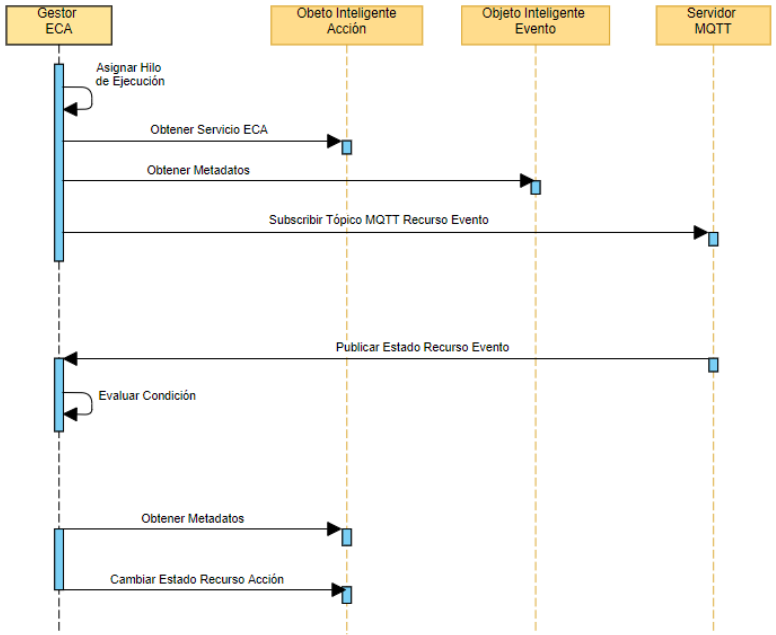


Figura 20. Diagrama secuencia Ejecutar ECA. Fuente: Propia

4.2.2 Vista de configuración

A nivel de usuario desarrollador el Framework expone los elementos centrales para que sean gestionados por medio de un archivo de configuración tipo JSON. Dentro de este archivo se encuentran condensados todos los elementos centrales para que el usuario desarrollador pueda gestionar el comportamiento de la aplicación deseada.

Todos los elementos del Framework no son directamente configurables, algunos se configuran de forma indirecta utilizando una configuración en específico. A continuación, se detallan las secciones del archivo de configuración.

Sección configuración Metadatos
Esta sección configura dos elementos centrales del Framework.

- Servidor CoAP. Esta sección entrega la configuración de los metadatos del OI, los cuales son consumidos por el elemento Servidor CoAP y los expone en el servicio de Leer Metadatos. Este servicio es utilizado tanto por los elementos de aplicación como el elemento central de descubrimiento de OI.

- Cliente MQTT. Esta sección entrega la configuración de los recursos del OI y con ellos la configuración del servidor MQTT en donde se desea publicar el estado de cada recurso.

La Figura 21 muestra el esquema de la sección de configuración de los metadatos. Podemos detallar los siguientes elementos principales

- Elementos "*name*" y "*description*". Conforman el nombre y la descripción del OI respectivamente y son datos fundamentales para el descubrimiento del OI dentro de una red o escenario.

- Elemento "*resources*". Es un listado de objetos que representan los recursos del OI. Cada objeto de este elemento contiene los datos de nombre "*name*" y descripción "*description*".

- Elemento "*tag*". Se encuentra dentro de un objeto del elemento "*resources*" y representa el identificador simplificado del recurso dentro del sistema. Este elemento es utilizado ampliamente en la generación de archivos y clases internas de la aplicación que utilice el Framework.

- Elemento "type". Se encuentra dentro de un objeto del elemento "resources" y representa el tipo de dato que maneja el estado del recurso. El Framework soporta cuatro tipos: entero, decimal, texto y booleano

- Elemento "mqtt". Se encuentra dentro del un objeto del elemento "resources" y contiene la configuración para realizar la conexión al servidor MQTT en donde se expone el estado del recurso

- Elemento "enabled". Se encuentra dentro del elemento "mqtt" y representa si la publicación MQTT del recurso en particular se encuentra habilitada o no en la aplicación. Si este elemento es "false" los demás elementos dentro del "mqtt" no serán tomados en cuenta para la aplicación desarrollada con el Framework.

- Elemento “delay_time”. Se encuentra dentro del elemento “mqtt” y representa la frecuencia en segundos de publicación del estado en el servidor MQTT

- Elemento “qos”. Se encuentra dentro del elemento “mqtt” y representa la calidad de servicio MQTT.

- Elementos “host” y “port”. Se encuentran dentro del elemento “mqtt” y contienen los datos para realizar la conexión al servidor MQTT

- Elementos “user” y “password”. Se encuentran dentro del elemento “mqtt” y contienen los datos para realizar la autenticación en el servidor MQTT. Estos elementos no son obligatorios y dependen de los requerimientos de seguridad que maneje el servidor MQTT.

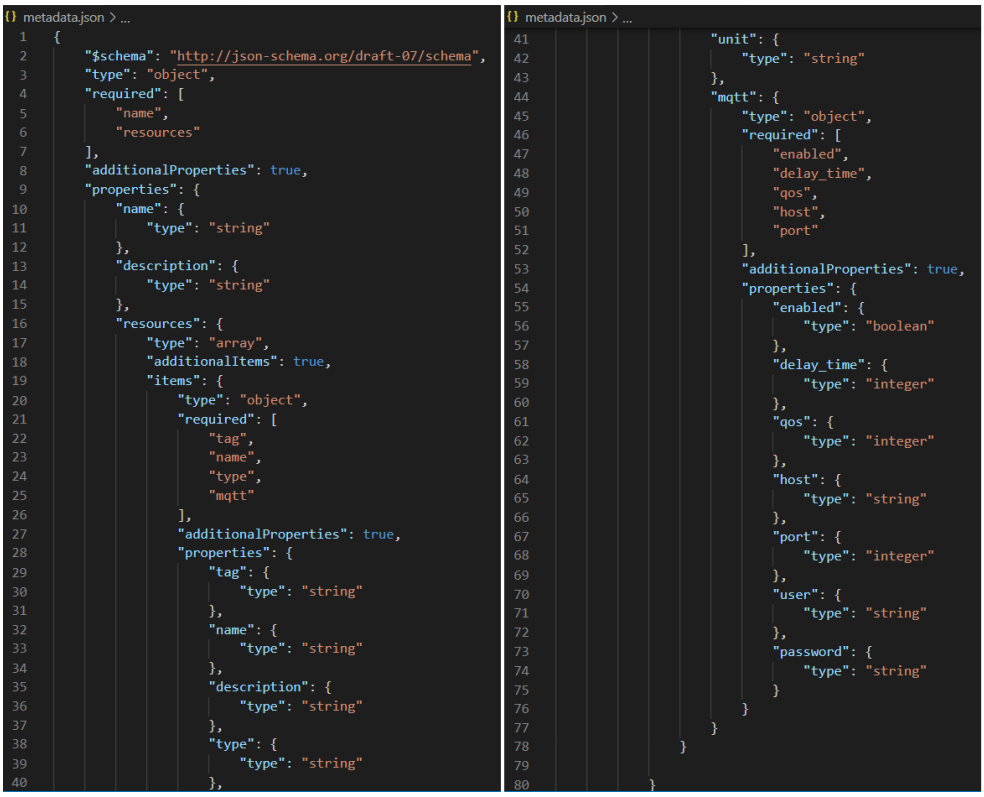


Figura 21. Esquema sección configuración metadatos. Fuente: Propia

Sección configuración Descubrimiento de OI

Esta sección configura el elemento central del Framework descubrimiento de OI, principalmente configura el elemento ontología de dominio y Crawler CoAP, el elemento restante Indice, utiliza estas configuraciones de forma indirecta debido a que este elemento consume las funciones de los dos antes mencionados.

La Figura 22 muestra el esquema de la sección de configuración del Crawler CoAP. Podemos detallar los siguientes elementos principales

- Elemento “enabled”. Define si el Crawler CoAP se encuentra habilitado o no en la aplicación. Si este elemento es “false” los demás elementos dentro de este esquema no serán tomados en cuenta para la aplicación desarrollada con el Framework, además la aplicación no realizará el paso inicial de búsqueda en el descubrimiento de objetos, y la matriz tendrá un comportamiento estático.

- Elemento “*protocol*”. Define el protocolo de comunicación habilitado para el Crawler CoAP. El Framework tiene habitados dos protocolos, CoAP y HTTP, siendo el primero el seleccionado por defecto.

- Elemento “*path*”. Contiene la ruta semilla por la cual el Crawler CoAP inicia la búsqueda en los dispositivos huéspedes. Por defecto esta se ubica en los metadatos, pero el usuario desarrollador es libre de modificar esta ruta.

- Elemento “*delay_time*”. Contiene el valor en segundo de la frecuencia con la que el Crawler CoAP inicia una nueva búsqueda por la red.

- Elemento “*url_list”.* Contiene una lista de objetos que representan las direcciones semilla de los disipativos con los cuales el Crawler CoAP inicia la búsqueda en la red.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "required": [
        "enabled",
        "protocol",
        "path",
        "delay_time",
        "url_list"
    ],
    "additionalProperties": true,
    "properties": {
        "enabled": {
            "type": "boolean"
        },
        "protocol": {
            "type": "string"
        },
        "path": {
            "type": "string"
        },
        "delay_time": {
            "type": "integer"
        },
        "url_list": {
            "type": "array",
            "additionalItems": true,
            "items": {
                "type": "string"
            }
        }
    }
}
```

Figura 22. Esquema sección configuración Crawler CoAP. Fuente: Propia

La Figura 23 muestra el esquema de la sección de configuración de la ontología de dominio. Esta sección se conforma de una lista de objetos que representan una ontología de dominio,

esto significa que el Framework soporta múltiples contextos basados cada uno en una ontología de dominio. Podemos detallar los siguientes elementos principales del objeto ontología de dominio a continuación.

- Elemento “enabled”. Define si la ontología de dominio se encuentra habilitada o no en la aplicación. Si este elemento es “false” los demás elementos dentro de este esquema no serán tomados en cuenta para la aplicación desarrollada con el Framework, además la aplicación no realizará el filtrado basado en la ontología deshabilitada.

- Elemento “tag”. Representa el identificador simplificado de la ontología de dominio dentro del sistema. Por defecto el Framework solo ofrece una ontología de dominio DOGONT la cual define un contexto de domótica o de casa inteligente.

- Elemento “*min_wup_similarity*”. Contiene el valor entre 0 y 1 para evaluar la distancia semántica de dos términos. Si el valor es más cercano a 1, significa que los términos tienen una alta similitud, y si son cercanos a cero, significa que son lejanos. Este elemento es utilizado como el canon para evaluar la distancia semántica al momento de filtrar un término entrante en el índice.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "array",
    "additionalItems": true,
    "items": {
        "type": "object",
        "required": [
            "enabled",
            "tag",
            "min_wup_similarity"
        ],
        "additionalProperties": true,
        "properties": {
            "enabled": {
                "type": "boolean"
            },
            "tag": {
                "type": "string"
            },
            "min_wup_similarity": {
                "type": "number"
            }
        }
    }
}
```

Figura 23. Esquema sección configuración Ontología de dominio. Fuente: Propia

Sección configuración ECA

Esta sección configura el elemento central del Framework gestor ECA. La Figura 24 muestra el esquema de la sección de configuración del gestor del ECA. Podemos detallar los siguientes elementos principales

- Elemento “enabled”. Define si el gestor ECA se encuentra habilitado o no en la aplicación. Si este elemento es “false” los demás elementos dentro de este esquema no

serán tomados en cuenta para la aplicación desarrollada con el Framework, además la aplicación no tendrá servicios de interoperabilidad semántica.

- Elemento "*coap_request_delay_time*". Define el valor del tiempo en segundos de la frecuencia con la que el gestor ECA realiza peticiones a los recursos evento y acción dado un servicio de interoperabilidad semántica. Estas peticiones se realizan si el recurso evento o acción, no tienen una dirección MQTT accesible, o está deshabilitada.

- Elemento "*mqtt_listener*". Contiene la configuración para realizar la conexión al servidor MQTT en donde se expone el estado de los recursos evento y acción.

- Elementos "user" y "password". Se encuentran dentro del elemento "mqtt_listener" y contienen los datos para realizar la autenticación en el servidor MQTT. Estos elementos no son obligatorios y dependen de los requerimientos de seguridad que maneje el servidor MQTT.

```
{
    "$schema": "http://json-schema.org/draft-07/schema",
    "type": "object",
    "required": [
        "enabled",
        "coap_request_delay_time",
        "mqtt_listener"
    ],
    "additionalProperties": true,
    "properties": {
        "enabled": {
            "type": "boolean"
        },
        "coap_request_delay_time": {
            "type": "integer"
        },
        "mqtt_listener": {
            "type": "object",
            "required": [
                "enabled",
                "user",
                "password"
            ],
            "additionalProperties": true,
            "properties": {
                "enabled": {
                    "type": "boolean"
                },
                "user": {
                    "type": "string"
                },
                "password": {
                    "type": "string"
                }
            }
        }
    }
}
```

Figura 24. Esquema sección configuración gestor ECA. Fuente: Propia

4.2.3 Vista de comandos

A nivel de usuario desarrollador el Framework expone los elementos centrales para que sean gestionados por medio de línea de comandos. Existen dos archivos para ejecutar los comandos con los cuales el Framework puede gestionar la aplicación y los elementos que la conforman.

Archivo "clipio_gen.py"

Este archivo se encuentra ubicado en el núcleo de la librería que conforma al Framework. Su única función es generar los archivos semilla o esqueleto de una aplicación. La Figura 25 muestra el comando utilizado para esto y el resultado a nivel de carpetas y archivos. Las partes parametrizables del comando son la siguientes.

- Parte "*name_some_oi*". El usuario desarrollador parametriza el nombre del nuevo OI o aplicación IoT

- Parte "route_some_oi". El usuario desarrollador parametriza la ruta donde será generado el nuevo OI o aplicación IoT

El resultado de este comando contiene 2 archivos y 2 carpetas descritas a continuación:

- Archivo "*settings.py*". Este archivo es utilizado para configurar los elementos principales del OI o aplicación IoT

- Archivo "*manage.py*" Este archivo contiene otros comandos utilizados para gestionar el OI o aplicación IoT

- Carpeta "*generated*". Contiene todos los archivos y documentos generados por el Framework para administrar la aplicación. Podemos destacar archivos auxiliares para manejo de recursos, el documento que representa la base de datos NoSQL para los servicios de interoperabilidad semántica ECA, el documento que representa el índice y el documento que representa los metadatos. Esta carpeta es restringida al usuario desarrollador.

- Carpeta "*app*". En este espacio el usuario desarrollador tiene si ambiente para escribir el código de su aplicación.

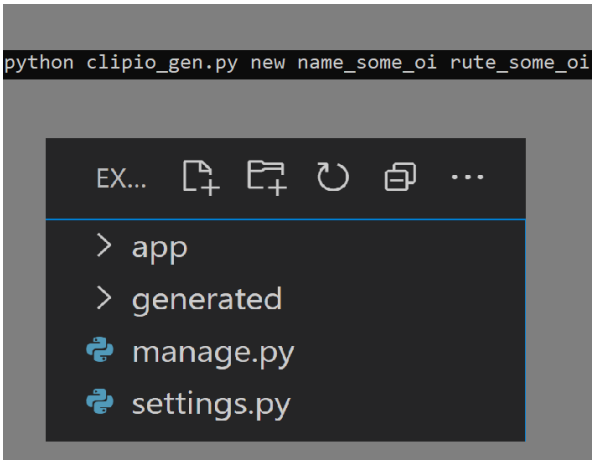


Figura 25. Comando y resultado nuevo OI. Fuente: Propia

Archivo "manage.py"

Este archivo se encuentra en cada OI o aplicación IoT. Tiene tres funciones principales las cuales son

- Generar archivo de los metadatos. La Figura 26 muestra el comando para generar el archivo de los metadatos. Este archivo es guardado en la carpeta "*generated*". La condición para ejecutar este comando es tener llena la sección de configuración de los metadatos entregada en el archivo de configuración.

```
/home/santiago/some_oi# python manage.py generate metadata
```

Figura 26. Comando generar metadatos. Fuente: Propia

- Generar archivos adicionales. La Figura 27 muestra el comando para generar los archivos adicionales, como archivos auxiliares para el manejo de recursos, base de datos NoSQL para los servicios de interoperabilidad semántica ECA y el documento que guarda el índice.

```
/home/santiago/some_oi# python manage.py generate all
```

Figura 27. Comando generar archivos adicionales. Fuente: Propia

- Iniciar servidor. La Figura 28 muestra el comando para iniciar una instancia del OI o aplicación. Este me habilita una instancia CoAP exponiendo todos los servicios referidos en el elemento servidor CoAP y que se encuentren habilitados en el archivo de configuración.

```
/home/santiago/some_oi# python manage.py server
```

Figura 28. Comando iniciar servidor. Fuente: Propia

4.2.4 Repositorio

El código fuente del Framework se encuentra expuesto en un repositorio público accesible desde la siguiente dirección: https://github.com/santomecoide/clipio/tree/dev

5 IMPLEMENTACIÓN DEL ESCENARIO

En este punto del proyecto se materializa el Framework en una prueba de concepto con el fin de analizar las fortalezas, debilidades y su impacto en la construcción de este tipo de escenarios de interoperabilidad semántica de OI en la WoT. Para este desarrollo se hace uso de la metodología de desarrollo de software UP Ágil [78] que define 4 fases. Para el presente documento se presentan las fases de inicio, de elaboración, de construcción y de transición.

5.1 FASE DE INICIO

La fase de inicio define 4 etapas, requisitos, casos de uso críticos, elementos tecnológicos y diseño de evaluación. Para fines de organización de la presente documentación se describirán los parámetros de evaluación en el capítulo del experimento del Framework. A continuación, se presentan las otras etapas.

5.1.1 Requisitos y casos de uso críticos

En este apartado se presentan los requisitos básicos que los OI deben cumplir para poder implementar correctamente la prueba de concepto propuesta. Adicionalmente a los requisitos se presentan los casos de uso críticos por cada requerimiento en formato UML. Estos requerimientos están definidos para un ambiente de 3 OI los cuales son implementados utilizando el Framework antes propuesto.

- Desarrollar herramientas que permitan crear y modificar los metadatos del OI. En la Figura 29 podemos ver los casos de uso UML.

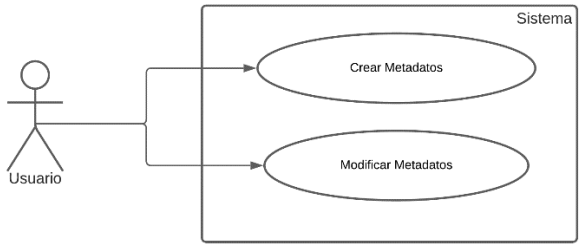


Figura 29. Casos de uso críticos. Primer requerimiento. Fuente: Propia

- Desarrollar herramientas que permitan crear y modificar servicios de interoperabilidad semántica ECA del OI. En la Figura 30 podemos ver los casos de uso UML

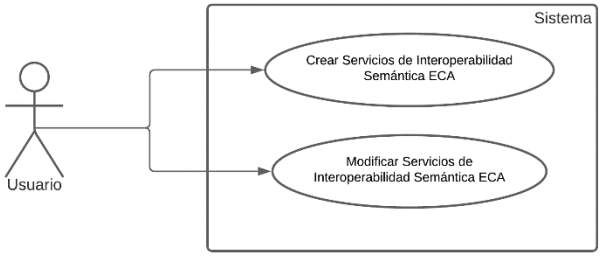


Figura 30. Casos de uso críticos. Segundo requerimiento. Fuente: Propia

- Desarrollar herramientas que permitan ejecutar servicios de interoperabilidad semántica ECA del OI. En la Figura 31 podemos ver el caso de uso UML

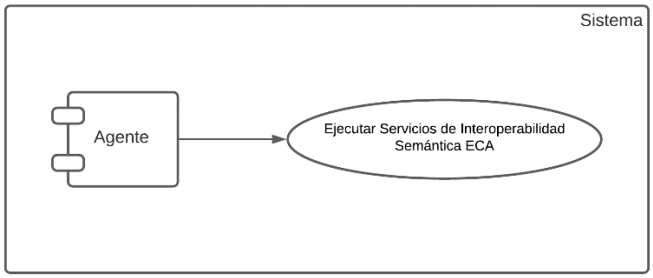


Figura 31. Casos de uso críticos. Tercer requerimiento. Fuente: Propia

- Desarrollar herramientas que permitan descubrir nuevos OI según un contexto prestablecido. En la Figura 32 podemos ver el caso de uso UML.

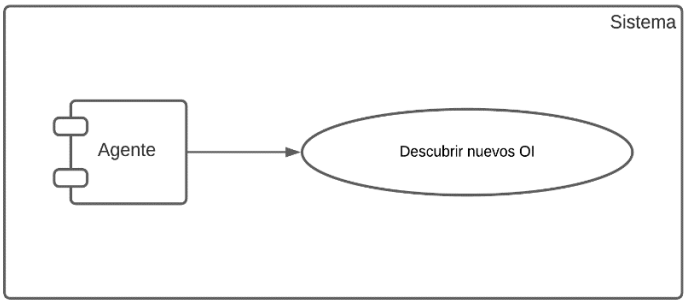


Figura 32. Casos de uso críticos. Cuarto requerimiento. Fuente: Propia

- Desarrollar herramientas que permitan publicar el estado de los recursos del OI. En la Figura 33 podemos ver el caso de uso UML.

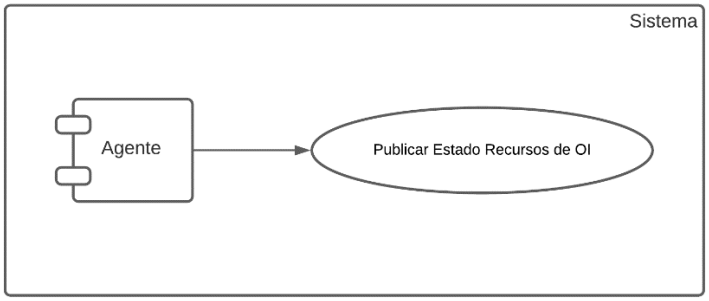


Figura 33. Casos de uso críticos. Quinto requerimiento. Fuente: Propia

- Desarrollar la aplicación base de cada OI. Esta aplicación corresponde al servicio básico el cual debe permanecer constante en el OI. Ejemplo. El regular de Luz debe prender y apagar la luz dependiendo de una constante de luminosidad, esto corresponde a su servicio básico. En la Figura 34 podemos ver el caso de uso UML.

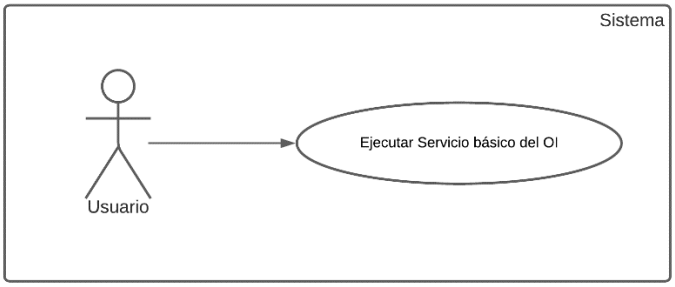


Figura 34. Casos de uso críticos. Quinto requerimiento. Fuente: Propia

5.1.2 Elementos tecnológicos

El escenario donde se despliegan los OI seleccionado es en un ambiente domótico, donde se pueden implementar muchos objetos con diversos propósitos, se puede identificar variadas propiedades, automatizar procesos y los distintos espacios dan lugar a plantear gran variedad de servicios. Esto debido a que la ontología de domino que por defecto entrega el Framework es DOGONT, la cual contiene conceptos y relaciones que definen un ambiente domótico o casa inteligente.

Cada OI en esta prueba de concepto está desplegado sobre la siguiente infraestructura:

- Placa Raspberry Pi 3B+, la cual contiene los componentes hardware necesarios para soportar un OI, tanto los puertos de múltiple propósito como la arquitectura hardware de microprocesadores para soportar sistemas operativos y herramientas software más complejas.

- Sistema operativo, Linux distribución Raspbian. GNU/Linux es la base requerida por el Framework para construir las aplicaciones. La distribución Raspbian, contiene los drivers para el manejo de la arquitectura hardware de las placas Raspberry Pi y este sistema operativo se encuentra adecuado a nivel de interfaz para administrar fácilmente este dispositivo.

- Lenguaje de programación Python 3.8. Python es el elemento transversal requerido por el Framework para construir las aplicaciones. La versión 3.8 de Python es la más actualizada a la fecha y comprende mejoras de rendimiento y de sintaxis con respecto a las anteriores versiones.

Los objetos propuestos para la presente prueba de concepto son:

Regulador de temperatura

Establece la temperatura de su ambiente cercano utilizando mecanismos que permiten detectarla, reducirla y aumentarla. Contiene los siguientes recursos

- Recurso sensor de temperatura. Permite medir la propiedad temperatura utilizando un circuito, el cual consta de: un termistor que detecta los cambios de la temperatura ambiente y una resistencia sobre la cual varía el voltaje acorde a los cambios en las

propiedades del termistor. El rango de detección es de -40°C a 125°C con una precisión de ±1.5ºC

- Recurso actuador ventilador. Permite disminuir el nivel de la propiedad temperatura utilizando un ventilador. Este elemento consta de un circuito donde al ventilador se le suministra energía regulada por un módulo relé o relevador de energía.

- Recurso actuador calentador. Permite aumentar el nivel de la propiedad temperatura el actuador calefactor. Este elemento es simulado por un led rojo.

Regulador de luz

Establece el nivel de luz de su ambiente cercano utilizando mecanismos que permiten detectar cuando la propiedad luminosidad se encuentre por debajo o por encima de un umbral deseado y aumentarla o reducirla. Contiene los siguientes recursos:

- Recurso sensor de luz. Permite medir la propiedad luminosidad utilizando un circuito, el cual consta de: un foto-resistor que detecta los cambios de luminosidad, un condensador y una resistencia sobre los cuales varía el voltaje acorde a los cambios en las propiedades de la foto-resistor se retiene el voltaje.

- Recurso actuador bombillo. Permite aumentar o disminuir el nivel de luminosidad mediante el prendido o apagado de este elemento. Este recurso es simulado por un led blanco

Regulador de humedad en una planta

Establece el nivel de humedad en una planta mediante mecanismos que permiten detectar cuando la propiedad se encuentra por debajo o por encima de un umbral deseado y aumentar o reducir la propiedad humedad. Contiene los siguientes recursos:

- Recurso sensor de humedad. Permite medir el nivel de humedad de la tierra donde está sembrada una planta. Las mediciones son emuladas por un arreglo de resistencias variables (a causa de no tener el sensor o módulo que permite realizar este tipo de mediciones)

- Recurso actor riego. Permite aumentar o disminuir el nivel de humedad utilizando un motor que abre o cierra las compuertas para dejar pasar cierta cantidad de agua hacia la plata.

Otros elementos y decisiones tecnológicas utilizadas para esta prueba de concepto son:

- El elemento coordinador. Para este elemento se seleccionó la aplicación IoTCoAP, la cual se encuentra disponible en la tienda PlayStore[7] para dispositivos Android con versiones mayores o iguales a la versión 5.0. El dispositivo debe tener como requisito

[7] Tienda de aplicaciones de Android

básico conexión a internet. La aplicación permite generar archivos JSON y construir peticiones CoAP con diferentes métodos. Para las pruebas se utiliza el dispositivo móvil Huawei P10 Lite.

- Para el servidor bróker MQTT se utiliza la infraestructura de Paho (iot.eclipse.org) soportado por Eclipse. Este servidor es capaz de soportar aplicaciones basadas en TCP y WS, facilitando la conexión de varias aplicaciones.

5.2 FASE DE ELABORACIÓN

Para el desarrollo del escenario con los tres OI se definen los siguientes módulos, cada uno responde a los requerimientos básicos planteados en la fase de inicio, por lo tanto, cada módulo contiene un grupo de casos de uso relacionado

- Módulo de servicios. Este módulo contiene los casos de uso: crear metadatos, modificar metadatos, crear servicio de interoperabilidad semántica ECA y modificar servicio de interoperabilidad semántica ECA.

- Módulo ejecución interna. Este módulo contiene los casos de uso: descubrir nuevos OI, ejecutar servicios de interoperabilidad semántica ECA y publicar estados recursos de OI.

- Módulo de aplicación. Este módulo contiene el caso de uso: ejecutar servicio básico del OI. Cada OI inteligente contiene por lo menos un servicio básico asociado a su funcionamiento.

A continuación, se presentan los formatos expandidos de los casos de uso críticos desarrollados asociados a los distintos módulos. Los casos de uso reales pueden ser consultados en el anexo D.

5.2.1 Módulo de servicios

La construcción de este módulo está basada en la capa de servicios de la arquitectura de soporte del Framework, en donde se exponen los servicios de la aplicación utilizando CoAP y encapsulando los documentos en estándares basados en JSON. El uso del Framework para la construcción de este módulo presenta un despliegue a un alto nivel de los casos de uso y los formatos expandidos se presentan en función al desarrollo del módulo bajo el Framework.

Nombre	Crear Metadatos
Actores	Usuario
Propósito	Permite crear los metadatos de un OI.
Resumen	Crea el documento de metadatos del OI a partir del archivo de configuración y el comando de generación de metadatos facilitados por el Framework.

Tipo	Primario.
Precondición	- El OI debe tener un soporte Hardware, sistema operativo y el Framework instalado - El OI debe tener el archivo de configuración semilla entregado por el Framework - El OI debe tener el archivo de ejecución entregado por el Framework
Curso normal de los eventos	
Acción del actor usuario	**Respuesta del sistema**
El usuario completa el documento de configuración del OI con la información correspondiente a los metadatos	
El usuario ejecuta el comando para generar el documento de metadatos	El sistema genera un documento JSON bajo la recomendación definida por el Framework

Tabla 4. Caso de uso crear metadatos. Fuente: Propia

Nombre	**Modificar Metadatos**
Actores	Usuario
Propósito	Permite editar los metadatos de un OI.
Resumen	Edita el documento de metadatos del OI a partir del archivo de configuración y el comando de generación de metadatos facilitados por el Framework.
Tipo	Primario.
Precondición	- El OI debe tener un soporte Hardware, sistema operativo y el Framework instalado - El OI debe tener el archivo de configuración semilla entregado por el Framework - El OI debe tener el archivo de ejecución entregado por el Framework - El OI debe tener metadatos
Curso normal de los eventos	
Acción del actor usuario	**Respuesta del sistema**
El usuario modifica el documento de configuración del OI con la nueva información correspondiente a los metadatos	
El usuario ejecuta el comando para generar el documento de metadatos	El sistema genera un documento JSON bajo la recomendación definida por el Framework y elimina el anterior.

Tabla 5. Caso de uso modificar metadatos. Fuente: Propia

Nombre	**Crear Servicio de interoperabilidad semántica ECA**
Actores	Usuario
Propósito	Permite crear un ECA
Resumen	El usuario consulta sobre OI relacionados, genera un nuevo ECA en formato simple, el sistema lo transforma a un formato extendido y lo persiste.
Tipo	Primario
Precondición	- El OI debe tener un soporte Hardware, sistema operativo y el Framework instalado - El OI debe tener metadatos - El OI debe tener los servicios activos

	- El usuario debe tener un dispositivo con una aplicación para hacer peticiones CoAP
Curso normal de los eventos	
Acción del actor usuario	**Respuesta del sistema**
El usuario consulta al OI sobre OI relacionados. Esta consulta es un texto en lenguaje natural.	El sistema responde con una lista ponderada de OI relacionados al contexto dado.
El usuario selecciona un OI evento.	El sistema entrega los metadatos del OI evento y los metadatos de su OI huésped como el OI acción
El usuario genera un JSON simplificado con la información del ECA	
El usuario realiza una petición con la aplicación tercera, para generar un nuevo ECA	El sistema genera un documento JSON bajo la recomendación definida por el Framework
	El sistema persiste el documento con el nuevo ECA

Tabla 6. Caso de uso crear servicio de interoperabilidad ECA. Fuente: Propia

Nombre	**Modificar Servicio de interoperabilidad semántica ECA**
Actores	Usuario
Propósito	Permite editar un ECA
Resumen	El usuario realiza una petición de cambio el estado de la actividad de un ECA.
Tipo	Primario
Precondición	- El OI debe tener un soporte Hardware, sistema operativo y el Framework instalado - El OI debe tener metadatos - El OI debe tener los servicios activos - El usuario debe tener un dispositivo con una aplicación para hacer peticiones CoAP - El OI debe tener por lo menos un ECA creado
Curso normal de los eventos	
Acción del actor usuario	**Respuesta del sistema**
El usuario realiza una petición con la aplicación tercera, para modificar el estado de un ECA. Entrega identificador del ECA y el estado deseado	El sistema modifica el estado del ECA.
El usuario selecciona un OI evento.	El sistema interrumpe los procesos relacionados al ECA (si el nuevo estado es apagado)

Tabla 7. Caso de uso modificar servicio de interoperabilidad ECA. Fuente: Propia

5.2.2 Módulo de ejecución interna

La construcción de este módulo está basada en la capa semántica, en la capa gestión ECA y la capa de interacción de la arquitectura de soporte del Framework. Para estos casos de uso se omiten varias etapas, debido a que son ejecuciones que el sistema hace de forma interna y con el uso del Framework este módulo no corresponde un trabajo de desarrollo e implementación significativo.

Nombre	Descubrir Nuevos OI	
Actores	Agente interno	
Propósito	Permite buscar e indexar nuevos OI.	
Resumen	Un agente interno se dedica periódicamente a consultar e indexar nuevos OI dentro de una red y bajo un contexto prestablecidos por el usuario.	
Tipo	Secundario.	
Precondición	- El OI debe tener un soporte Hardware, sistema operativo y el Framework instalado - El OI debe tener el archivo de configuración semilla entregado por el Framework - El OI debe tener el archivo de ejecución entregado por el Framework	
Curso normal de los eventos		
Acción del actor usuario		**Respuesta del sistema**
El usuario completa el documento de configuración del OI con la información correspondiente al descubrimiento de OI		
		El Crawler CoAP realiza periódicamente la búsqueda de nuevos OI en la red definida en el archivo de configuración
		Los nuevos OI son indexados bajo un contexto entregado por la ontología de dominio definida en el archivo de configuración

Tabla 8. Caso de uso descubrir nuevos OI. Fuente: Propia

Nombre	Ejecutar servicios de interoperabilidad semántica ECA	
Actores	Agente interno	
Propósito	Permite ejecutar ECA.	
Resumen	Un agente interno se dedica periódicamente a consultar el estado de los servicios ECA y los que se encuentren activos los ejecuta.	
Tipo	Secundario.	
Precondición	- El OI debe tener un soporte Hardware, sistema operativo y el Framework instalado - El OI debe tener el archivo de configuración semilla entregado por el Framework - El OI debe tener el archivo de ejecución entregado por el Framework - El OI debe tener un por lo menos un ECA con estado activo	
Curso normal de los eventos		
Acción del actor usuario		**Respuesta del sistema**
El usuario completa el documento de configuración del OI con la información correspondiente la gestión de ECA		
		El sistema busca ECA con estado activo
		El sistema ejecuta en un proceso aislado el ECA
		Cuando el ECA cumple el apartado de la condición, el sistema finaliza al ECA

	El sistema cambia el estado del ECA a inactivo

Tabla 9. Caso de uso Ejecutar servicios de interoperabilidad semántica ECA. Fuente: Propia

Nombre	Publicar estado recursos del OI
Actores	Agente interno
Propósito	Permite publicar en MQTT los estados de los recursos.
Resumen	Un agente interno se dedica periódicamente a publicar los estados de cada recurso del OI.
Tipo	Secundario.
Precondición	- El OI debe tener un soporte Hardware, sistema operativo y el Framework instalado - El OI debe tener el archivo de configuración semilla entregado por el Framework - El OI debe tener el archivo de ejecución entregado por el Framework - El OI debe tener los metadatos
Curso normal de los eventos	
Acción del actor usuario	**Respuesta del sistema**
El usuario completa el documento de configuración del OI con la información correspondiente a los metadatos de los recursos del OI y al servicio MQTT	
	El sistema genera un documento por recurso para guardar el estado actual del recuro
	El sistema genera rutas (tópicos MQTT) por recurso
	El sistema conecta cada recurso con el servidor MQTT preestablecido en el documento de configuración
	El sistema publica el estado del recurso con una frecuencia establecida en el documento de configuración

Tabla 10. Caso de uso Publicar estado recursos del OI. Fuente: Propia

5.2.3 Módulo de aplicación

La construcción de este módulo corresponde al elemento de aplicaciones establecido en el Framework. Estos casos de uso se manejan de forma abstracta debido a que depende del servicio básico correspondiente a cada OI que se desea construir. La implicación del Framework en este módulo no es alta ya que en este punto es un soporte de las aplicaciones. Para este módulo se omiten los formatos expandidos y se explican los requerimientos de cada servicio básico a un alto nivel.

Recurso regulador de temperatura

Este servicio básico pertenece al primer OI, el cual busca regular la temperatura utilizando tres recursos, un recurso actuador termostato, un recurso actuador ventilador y un recurso sensor de temperatura. El usuario puede establecer un valor deseado de la temperatura

ambiente y el OI debe hacer uso de sus recursos para mantener la temperatura lo más cercana al valor solicitado por el usuario.

Recurso regulador de luz
Este servicio básico pertenece al segundo OI, el cual busca regular la luz en un espacio, esto lo hace con dos recursos, un recurso actuador bombillo y un recurso sensor de luz. El usuario puede establecer un valor límite de luminosidad para apagar su fuente de luz y el OI debe hacer uso de sus recursos para detectar este valor en em ambiente y apagar el recurso actuador bombillo.

Recurso regulador humedad en planta
Este servicio básico pertenece al tercer OI, el cual busca regular la humedad del suelo de una planta, esto lo hace con dos recursos, un recurso actuador riego y un recurso sensor de humedad. El usuario puede establecer un valor límite de humedad en el suelo de la planta y cuando este se cumpla el OI debe apagar el recurso actuador riego.

5.3 FASE DE CONSTRUCCIÓN

En esta fase se implementan las OI basados en los casos de uso antes expuestos. Esto se realiza en 4 iteraciones, las cuales contienen diferentes actividades incrementales relacionadas a los módulos antes descritos. Igualmente se definen las interfaces resultantes tanto del manejo del Framework, como de aplicaciones terceras utilizadas para esta implementación.

5.3.1 Iteración 1

Para esta iteración se realizan dos grupos de actividades que se encuentran por fuera de los casos de uso críticos, pero si corresponden a una base importante para poder continuar con la implementación de los OI del escenario.

Acondicionamiento Hardware
Aunque las actividades relacionadas al aprovisionamiento hardware no se definan en los casos de uso críticos, estas corresponden al soporte físico de nuestro escenario. En este grupo de actividades encontramos las siguientes:

- Adquisición de elementos electrónicos como cables, resistencias, capacitores, sensores y reguladores. Además de la adquisición de las placas Raspberry Pi 3B y sus elementos de conexión y carga eléctrica

- Ensamble de los OI. Cada OI propuesto utiliza diferentes sensores y actuadores, por lo tanto, es necesario construir los circuitos donde se definen elementos pasivos y carga para cada OI.

- Punto de red. El escenario es propuesto dentro de una red local, por lo tanto, es necesario adquirir y configurar un punto de red para realizar la conexión de los OI.

Acondicionamiento Software

Aunque algunas de las actividades relacionadas a la infraestructura software no se definen en los casos de uso críticos, estas corresponden al soporte físico de nuestro escenario. En este grupo de actividades encontramos las siguientes:

- Instalación del sistema operativo dentro de cada OI. Como o define el Framework, es necesario soportar las aplicaciones sobre GNU/LINUX

- Instalación del lenguaje de programación. Aunque La instalación de Linux trae Python por defecto, es necesario instalar Python 3.8, debido a que esta es la versión soportada por el Framework

- Instalación del Framework. Se debe descargar las librerías que conforman al Framework de un repositorio público, además es necesario instalar algunas dependencias terceras
- Generación de los OI a partir de la funcionalidad ECA para crear el documento de configuración semilla y el archivo de ejecución de comandos semilla.

Pruebas alfa

Las pruebas alfa correspondientes a esta iteración se entrega evidencia fotográfica de los OI ensamblados con sus respectivos recursos y conectados a un punto de red común, además se entrega evidencia del sistema operativo funcional con los archivos del Framework y Python 3.8 instalado. Estas evidencias pueden ser consultadas en el Anexo E.

5.3.2 Iteración 2

Para esta iteración se implementa el módulo de servicios. Este módulo comprende los casos de uso, crear metadatos, editar metadatos, crear servicio de interoperabilidad semántica ECA y editar servicio de interoperabilidad semántica ECA. Las actividades en esta iteración se dividen en dos grupos, el grupo de actividades referentes a los metadatos y el grupo de actividades referente a servicios de interoperabilidad semántica ECA

Gestión de metadatos

En este grupo tenemos las actividades relacionadas a los casos de uso para los metadatos. Para los casos de uso crear como modificar los metadatos de un OI es necesario interactuar con el archivo de configuración del Framework. La Figura 35 muestra El archivo de configuración de los metadatos para el OI regulador de temperatura y a continuación se describen las actividades en función a la configuración de los parámetros de la sección contenida dentro de la constante METADATA.

```
METADATA = {
    "name": "Regulador de temperatura",
    "description": "Regulador de temperatura ubicado en la sala",
    "resources": [
        {
            "tag": "temperatura",
            "name": "Sensor de temperatura",
            "description": "Mide la temperatura del ambiente",
            "type": "integer",
            "unit": "C",
            "mqtt": {
                "enabled": True,
                "delay_time": 60,
                "qos": 0,
                "host": "iot.eclipse.org",
                "port": 1883,
                "user": "",
                "password": ""
            }
        },
        {
            "tag": "led",
            "name": "Calentador",
            "description": "Actuador calentador",
            "type": "boolean",
            "unit": "",
            "mqtt": {
                "enabled": True,
                "delay_time": 60,
                "qos": 0,
                "host": "iot.eclipse.org",
                "port": 1883,
                "user": "",
                "password": ""
            }
        }
    ]
}
```

Figura 35. Documento configuración módulo ejecución interna. Fuente: Propia

- Configurar parámetros generales del OI. Los parámetros "*name*" y "*description*" corresponden al nombre y a la descripción del OI.

- Configurar los parámetros de los recursos. La etiqueta "*resources*" contiene un listado de elementos tipo recurso, cada uno de estos representa los metadatos de un recurso dentro el OI.

- Configurar parámetro identificador. El parámetro "tag" contiene un identificador único del recurso dentro del sistema, este parámetro es utilizado para crear los archivos y elementos internos para gestionar el recurso.

- Configurar parámetros generales del recurso. Los parámetros "*name*" y "*description*" corresponden al nombre y a la descripción del recurso y a.

- Configurar parámetros de estado del recurso. Los parámetros "*type*" y "*unit*" corresponden al tipo de variable y la unidad que maneja el estado del recurso. Estos dos parámetros están enfocados en delimitar el valor entregado por el recurso.

El apartado "*mqtt*" es configurado en la siguiente iteración, por lo tanto, la ejecución de esta actividad no se realiza en esta sección.

Gestión de ECA

En este grupo tenemos las actividades relacionadas a los casos de uso para los ECA. Para los casos de uso crear y modificar servicio de interoperabilidad semántica ECA se tienen las siguientes actividades:

- Generar un documento JSON simplificado. Este documento contiene la información básica del nuevo servicio de interoperabilidad semántica ECA. La Figura 36 muestra un ejemplo del documento. Podemos encontrar los parámetros generales "*name*", "*description*" con la información principal del ECA. El parámetro "event" contiene la información necesaria para conocer el recurso evento y las conexiones al OI evento y al servidor MQTT para consumir el estado del recurso evento. El parámetro "condition" entrega el operador y el valor para ejecutar la condición del ECA. El parámetro "action" contiene la información necesaria para conocer el recurso acción y las conexiones al OI acción para modificar el estado del recurso acción.

```
payload = {
    "name": "Nombre del ECA",
    "description": "Descripcion del ECA",
    "event": {
        "type": "integer",
        "unit": "ºC",
        "mqtt_href": "mqtt://iot.eclipse.org:1883/90072060600/ejemplo",
        "coap_href": "coap://192.168.0.25:5683/temperature/get"
    },
    "condition": {
        "operator": ">",
        "value": 10
    },
    "action": {
        "type": "boolean",
        "value": 1,
        "coap_href": "coap://192.168.0.25:5683/heater/set"
    }
}
```

Figura 36. Documento JSON simplificado del ECA. Fuente: Propia

- Enviar el documento JSON simplificado al método de creación de ECA del OI acción. Para esta actividad es utilizada la aplicación IoTCoAP, la cual puede interactuar con servicios CoAP a nivel de red local. en la Figura 37 se muestran las interfaces de esta aplicación. En la primera interfaz se configura la IP del OI y el protocolo y el puerto. En la segunda interfaz se configura la ruta del método y el cuerpo del mensaje, el cuerpo corresponde al Documento JSON simplificado del ECA. Finalmente se envía la solicitud y en la última interfaz la aplicación muestra el estado de la petición.

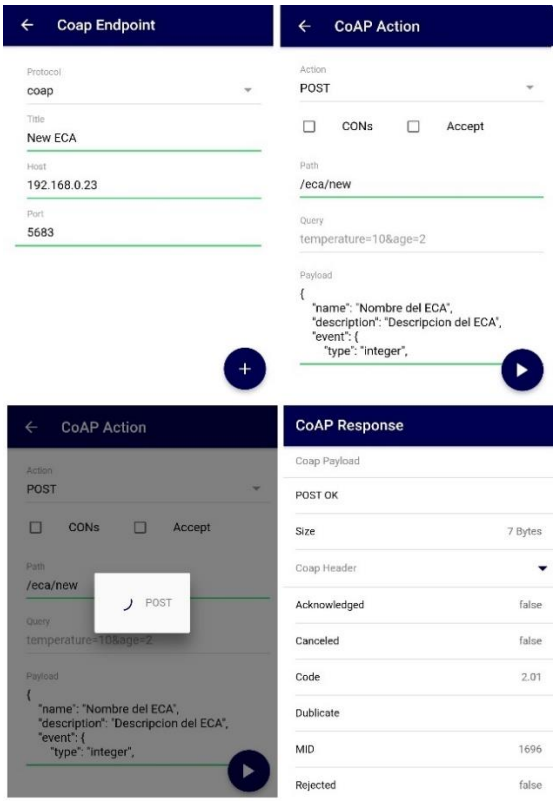


Figura 37. Nuevo ECA en IoTCoAP. Fuente: Propia

Pruebas alfa

Para este módulo no se realizan pruebas alfa a nivel técnico, debido a que no se realiza desarrollo de código, por lo tanto, las pruebas de este módulo son únicamente funcionales y tienen como fin comprobar el funcionamiento a un alto nivel. Para estas pruebas se crean los siguientes servicios de interoperabilidad semántica ECA

Evento	Condición	Acción	
Recurso		Recurso	Estado
T	Temperatura > 50 ºC	R	ON
T	Temperatura < 50 ºC	B	ON
T	Temperatura > 10 ºC	R	ON
T	Temperatura < 10 ºC	R	OFF
T	Temperatura > 50 ºC	R	ON
H	Humedad > 50 %	V	ON
H	Humedad < 50 %	C	ON
H	Humedad > 60 %	V	ON
H	Humedad > 60 %	V	ON
L	Luz < 30 %	C	ON
R	Riego = ON	V	ON
R	Riego = ON	C	ON
V	Ventilador = ON	R	ON

Tabla 11. Pruebas funcionales. Crear nuevos servicios de interoperabilidad semántica ECA

En la Tabla 11 podemos distinguir los siguientes elementos:
T, recurso sensor de temperatura, parte del OI regulador de temperatura.

L, recurso sensor de luz, parte del OI regulador de luz.
H, recurso sensor de humedad, parte del OI regulador de humedad en planta.
R, recurso actuador de riego, parte del OI regulador de humedad en planta.
B, recurso actuador bombillo, parte del OI regulador de luz
V, recurso actuador ventilador, parte del OI regulador de temperatura.
C, recurso actuador calefactor, parte del OI regulador de temperatura.

Como resultado se observó que no existe variación de tiempos para la creación de servicios ECA entre los distintos objetos. Y en cuanto a la creación de ECA con la aplicación tercera no surgieron problemas de acople o de cuotas de consumo.

5.3.3 Iteración 3

Para esta iteración se implementa el módulo de ejecución interna. Este módulo comprende los casos de uso, descubrir nuevos OI, ejecutar servicios de interoperabilidad semántica ECA y publicar estados recursos de OI. Como el Framework realiza la gran parte de estas actividades de forma transparente, las actividades en este punto son pocas y se relacionan con completar o modificar el documento de configuración entregado por el Framework.

En la Figura 38 podemos ver las secciones de configuración de los diferentes elementos que comprenden el módulo de ejecución interna y las actividades se relacionas con la descripción de cada variable.

```
CRAWLER = {
    "enabled": True,
    "protocol": "coap",
    "path": "metadata",
    "delay_time": 3600,
    "url_list": ["192.168.0.28", "192.168.0.32", "192.168.0.17"]
}

ECA = {
    "enabled": True,
    "coap_request_delay_time": 60,
    "mqtt_listener": {
        "enabled": True,
        "user": "",
        "password": "",
    }
}

ONTOLOGIES = [
    {
        "enabled": True,
        "tag": "home",
        "min_wup_similarity": 0.5,
    }
]
```

Figura 38. Documento configuración módulo ejecución interna. Fuente: Propia

- Configurar la constante CRAWLER. Esta sección está relacionada con el caso de uso descubrimiento de nuevos OI y configura el elemento Crawler CoAP del Framework. El parámetro "*enabled*" permite habilitar o deshabilitar este agente, para este caso está habilitado. El parámetro "*protocol*" permite seleccionar el protocolo web para realizar las consultas en los otros OI del escenario, para este caso es CoAP. El parámetro "*path*"

permite seleccionar el camino semilla el cual será el predeterminado para hacer la búsqueda del documento de metadatos en los ortos OI del escenario. El parámetro "*delay_time*" corresponde al periodo de tiempo en segundos donde se reinicie la búsqueda de los metadatos en la red. El parámetro "*url_list*" permite seleccionar las rutas semilla sobre las cuales se inicia la búsqueda de nuevos OI, para este caso son listadas las IP locales de los otros OI deplegados.

- Configurar la constante ECA. Esta sección está relacionada con el caso de uso ejecutar servicios de interoperabilidad semántica ECA y configura el elemento gestor ECA del Framework. El parámetro "*enabled*" permite habilitar o deshabilitar el agente que gestiona los ECA, para este caso está habilitado. El parámetro "*coap_request_delay_time*" corresponde al periodo de tiempo en segundos para realizar una consulta CoAP sobre un recurso, esta característica se utiliza cuando los recursos no exponen sus estados por MQTT, para los OI desplegados en el escenario todos los recursos son expuestos por MQTT, por lo tanto, este parámetro no es utilizado. El parámetro "*mqtt_listener.enabled*" permite habilitar o deshabilitar el agente MQTT que realiza las subscripciones a los canales de los recursos de los OI tipo acción definidos en el ECA. Los parámetros "*mqtt_listener.user*" y "*mqtt_listener.password*" son utilizados para manejar la autenticación en el servidor MQTT, para este caso no son requeridos.

- Configurar la constante ONTOLOGIES. Esta sección está relacionada con el caso de uso descubrimiento de nuevos OI y configura el elemento ontología de dominio e índice del Framework. Esta constante corresponde a un arreglo de objetos, por lo tanto, es posible configurar varias ontologías de dominio, para que el OI pueda tener varios contextos, para este caso solo se configura una. El parámetro "*enabled*" permite habilitar o deshabilitar este elemento, para este caso está habilitado. El parámetro "*tag*" selecciona a nivel interno el archivo OWL asociado a la ontología de dominio, para este caso "*home*" corresponde a DOGONT. El parámetro "*min_wup_similarity*" permite configurar el mínimo valor de la distancia semántica aceptado para realizar el proceso de filtrado de loe términos en el índice, para este caso el valor es 0.5 dejando un rango medio de términos que pueden ser indexados.

En la Figura 39 tenemos la configuración realizada para el recurso sensor de temperatura del OI regulador de temperatura. Esto para ejemplificar como es la configuración de los recursos dentro del archivo de configuración del Framework. Para estos casos la actividad es la siguiente:

Figura 39. Documento configuración. Configuración recursos. Fuente: Propia

- Configurar los recursos del OI. Esta sección de configuración está relacionada con el caso de uso publicar estados recursos del OI y configura el elemento cliente MQTT del Framework. Para este caso analizaremos la etiqueta "*mqtt*" la cual resuelve el caso de uso en cuestión. El parámetro "*enabled*" permite habilitar o deshabilitar este elemento, para este caso está habilitado. El parámetro "*delay_time*" corresponde al periodo de tiempo en segundos cuando se publica el estado del recurso en el servicio MQTT. El parámetro "*qos*" corresponde al nivel de calidad MQTT de la publicación del estado del recurso. Los parámetros "*host*" y "*port*" corresponde a la dirección y el puerto del servidor MQTT. Los parámetros "*user*" y "*password*" son utilizados para manejar la autenticación en el servidor MQTT, para este caso no son requeridos.

Pruebas alfa

Para este módulo no se realizan pruebas alfa a nivel técnico, debido a que no se realiza desarrollo de código, por lo tanto, las pruebas de este módulo son únicamente funcionales y tienen como fin comprobar el funcionamiento a un alto nivel. Para esto se realizaron las pruebas sobre los servicios de interoperabilidad semántica creados en la anterior iteración y se concluyó que no existían problemas a nivel funcional en cuanto al comportamiento deseado de la ejecución de los ECA en el escenario.

5.3.4 Iteración 4

Para esta iteración se implementa el módulo de aplicación. Este módulo comprende el caso de uso ejecutar servicio básico del OI. Las actividades en esta iteración se dividen en 3 grupos referentes a la implementación de los servicios básicos de cada OI.

Regulador de temperatura

Este servicio básico corresponde al primer OI. El código fuente de esta aplicación puede ser encontrado en el Anexo E. las actividades para desarrollar esta aplicación son:

- Los recursos se comunican con el Framework utilizando la clase asignada a cada recurso, de esta forma los valores capturados o modificados para cada uno de los recursos serán persistidos, publicados y expuestos por los componentes del Framework

- Establecer un valor constante el cual es utilizado para compararlo con el valor del sensor de temperatura

- Cuando el valor constante es menor al entregado por el sensor, el recurso actuador ventilador se prende y el recurso actuador termostato se apaga

- Cuando el valor constante es mayor al entregado por el sensor, el recurso actuador ventilador se apaga y el recurso actuador termostato se prende

- La lógica de los anteriores puntos está en un bucle infinito la cual se ejecuta en un periodo de 10 segundos

En la Figura 40 se puede ver el código de la aplicación del regulador de temperatura. La clase importada "*ValueHelper*" funciona como intermediario para conectar a los recursos con el Framework. Los objetos creados a partir de esta clase pueden obtener y modificar el valor del recurso con la variable "*value*" esto se hace de forma transversal a la aplicación y al Framework.

```python
# file for temperatura iot resource
# use ValueHelper for share your iot resources with clipio network
# if you need push and pull data use ValueHelper -> value

import time
from clipio.utils.value_helper import ValueHelper
temperatura_vh_obj = ValueHelper('temperatura', 'integer')
fan_vh_obj = ValueHelper('fan', 'boolean')
heater_vh_obj = ValueHelper('heater', 'boolean')

TEMP_CANON = 10  #10°C
def main():
    t_val = temperatura_vh_obj.value
    if t_val < TEMP_CANON:
        heater_vh_obj.value = True
        fan_vh_obj.value = False
    if t_val >= TEMP_CANON:
        heater_vh_obj.value = False
        fan_vh_obj.value = True

while True:
    main()
    time.sleep(60)
```

Figura 40. Código aplicación regulador de temperatura. Fuente: Propia

Regulador de luz

Este servicio básico corresponde al segundo OI. El código fuente de esta aplicación puede ser encontrado en el Anexo E. Las actividades para desarrollar esta aplicación son:

- Los recursos se comunican con el Framework utilizando la clase asignada a cada recurso, de esta forma los valores capturados o modificados para cada uno de los recursos serán persistidos, publicados y expuestos por los componentes del Framework

- Establecer un valor constante el cual es utilizado para compararlo con el valor del recurso sensor de luz

- Cuando el valor constante es menor al entregado por el sensor, el recurso actuador bombillo se apaga

Regulador de humedad en planta

Este servicio básico corresponde al tercer OI. El código fuente de esta aplicación puede ser encontrado en el Anexo E. Las actividades para desarrollar esta aplicación son:

- Los recursos se comunican con el Framework utilizando la clase asignada a cada recurso, de esta forma los valores capturados o modificados para cada uno de los recursos serán persistidos, publicados y expuestos por los componentes del Framework

- Establecer un valor constante el cual es utilizado para compararlo con el valor del recurso sensor humedad en la plata

- Cuando el valor constante es menor al entregado por el sensor, el recurso actuador bombillo se apaga

Pruebas alfa

Para las pruebas alfa correspondientes a esta iteración se realizaron pruebas unitarias con la librería "*unittest*" de Python, al ser aplicaciones relativamente simples, las pruebas no fueron difíciles de implementar y los resultados en su mayoría fueron positivos. La Figura 41 muestra el código de una prueba unitaria sobre la aplicación del regulador de temperatura y su resultado:

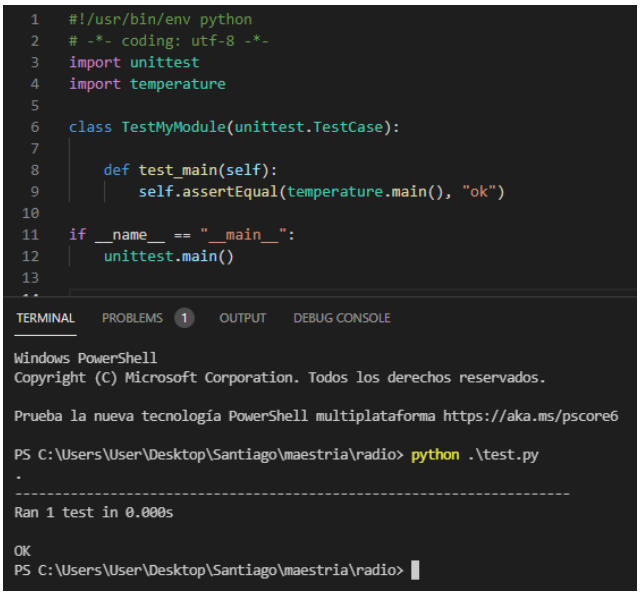


Figura 41. Código prueba unitaria aplicación regulador de temperatura. Fuente: Propia

5.4 FASE DE TRANSICIÓN

Para esta fase se cuenta con tres OI desarrollados utilizando el Framework, En el Anexo E se entrega el código fuente de cada uno de los OI implementados. Para esta fase no se realizaron pruebas beta debido a que los OI desarrollados fueron construidos con el propósito de probar y validar el Framework y no tienen la madurez necesaria para ser presentados a un usuario final no técnico, por lo tanto, se descartan este tipo de pruebas. Adicionalmente todo lo que compete con el paso a producción y la etapa de mantenimiento se descartan debido a que las aplicaciones construidas sobre los OI son netamente para fines investigativos y no comerciales.

6 EVALUACIÓN

Este capítulo presenta la evaluación del Framework, en donde se busca evaluar el esfuerzo invertido para la implementación de aplicaciones IoT entre ellas OI que generen servicios de interoperabilidad semántica con herramientas de la WoT. Está afirmación está alineada con la hipótesis propuesta en el proyecto la cual dice: “El Framework de interoperabilidad semántica de OI para la WoT, reduce el esfuerzo para la implementación de entornos de interoperabilidad semántica de OI”. Teniendo en cuenta lo anterior, el indicador que se debe tener en cuenta para esta evaluación es el esfuerzo, el cual nos determina si el Framework tiene un valor agregado en la construcción de entornos de interoperabilidad semántica de OI o de una forma más general, la construcción de aplicaciones IoT.

Realizar la evaluación para determinar si el Framework propuesto representa una reducción de esfuerzo para la construcción de aplicaciones IoT, puede ser una tarea compleja, debido a que, reducir el esfuerzo en cuanto a un Framework puede contemplar varias dimensiones, desde aspectos técnicos, hasta aspectos de impacto en la comunidad, pasando por aspectos subjetivos. Por lo tanto, es necesario establecer cuáles serán las medidas más adecuadas para determinar el índice de esfuerzo.

Definición Indicador de esfuerzo
Para lograr definir el indicador de esfuerzo se proponen las siguientes medidas:

- Tiempo. Se refiere a la medida del tiempo invertido en la construcción de una aplicación IoT.

- Cumplimiento. Se refiere a la medida de cumplimiento de los requerimientos establecidos para desarrollar una aplicación IoT.

- Dificultad. Se refiere a la medida del nivel de dificultad percibido en el desarrollo de la aplicación IoT.

A partir de estas medidas el indicador de esfuerzo se define con la ecuación expuesta a continuación, la expresa que, el esfuerzo es directamente proporcional con la medida del tiempo y la dificultad y es inversamente proporcional a la medida del cumplimiento.

$$Esfuerzo\ \alpha\ Tiempo\ \alpha\ \frac{1}{\text{Cumplimiento}}\ \alpha\ Dificultad$$

Dado los anteriores argumentos, el presente capítulo propone el diseño y le ejecución de un experimento para lograr determinar si el índice de esfuerzo al desarrollar una aplicación IoT utilizando el Framework es inferior al índice de esfuerzo al desarrollar una aplicación IoT sin utilizar el Framework. A continuación, se expone, el diseño del experimento, la ejecución del experimento y las conclusiones [79].

6.1 DISEÑO DEL EXPERIMENTO

El presente apartado presenta el diseño experimental propuesto para lograr medir la el esfuerzo en la construcción de aplicaciones IoT utilizando el Framework propuesto. Para eso separamos el diseño en la presentación de sus características y asignación de los participantes en los grupos.

6.1.1 Características del diseño del experimento

El diseño del experimento es de tipo verdadero debido a que el diseño reúne los dos requisitos para lograr el control y la validez interna. Contiene grupos de comparación (mínimo dos) y los participantes de estos grupos son homogéneos. Además, el tipo de

diseño es cuasi-experimental debido a que, los grupos de comparación no son asignados totalmente al azar, existe un proceso previo para ponderar a los participantes.

La Figura 42 muestra la generalidad del diseño del experimento. A continuación, se detallan las características básicas que definen el diseño del experimento

Figura 42. Diseño del experimento. Fuente: Propia

- Estímulo experimental (X). El estímulo para este experimento es la explicación y la utilización del Framework. Por lo tanto, el uso o no uso del Framework para desarrollar los requerimientos de la aplicación IoT se convierte en la variable dependiente.

- Grupos de comparación. Teniendo en cuenta que es necesario comparar el índice de esfuerzo de realizar una aplicación IoT utilizando el Framework y sin utilizarlo, el diseño propone tener dos grupos de participantes. Grupo experimental (Ga), este grupo realiza la aplicación IoT utilizando el estímulo (X). Grupo de control (Gb), este grupo realiza la aplicación IoT sin utilizar el estímulo (X)

- Post-prueba (O1, O2). Una vez los participantes de los grupos de comparación terminen el experimento, serán sometidos a una prueba, la cual es una revisión del cumplimiento de los requerimientos propuestos para la aplicación IoT. Cada participante presenta la aplicación IoT lograda y recibe dos tipos de calificación, una calificación de dificultad (0 - 5) la cual representa el nivel de dificultad del requerimiento expresado por el participante y otra calificación de cumplimiento (0% – 100%) la cual representa el cumplimiento a nivel técnico y funcional de los requerimientos.

- Etapa experimental. El experimento consiste en entregar a los participantes de ambos grupos una serie de requerimientos los cuales conforman la implementación de una aplicación IoT. Todos los participantes tienen un tiempo preestablecido para construir la aplicación IoT, y al final se entrega un producto que será evaluado a nivel técnico y funcional.

- Participantes. Los participantes del experimento deben tener un perfil específico. El participante debe tener un conocimiento medio en el lenguaje de programación Python,

debe tener experiencia en herramientas IoT tanto a nivel software como hardware y debe conocer sobre herramientas semánticas. Debido a estos prerrequisitos, la población de participantes se limita mucho y al final, los participantes del experimento quedaron en 6. Los participantes fueron buscados dentro del semillero de investigación IoTIC de la Universidad del Cacua, dentro de los posibles candidatos se decantaron a los estudiantes por sus conocimientos previos en el lenguaje de programación Python y el manejo de herramientas IoT.

- Tiempo del experimento. Se establece una semana para la culminación de todos los requerimientos para ambos grupos de comparación. Sin embargo, a cada participante se le debe solicitar las horas efectivas invertidas en la implementación de la aplicación IoT.

6.1.2 Asignación de participantes en grupos

Para lograr mitigar la principal variable independiente, que es el nivel de conocimientos técnicos previos en desarrollo de aplicaciones IoT que pueda tener el participante, se decide definir una prueba de conocimientos previa y según el ponderado de esta prueba se asignan los grupos de comparación, intentando dejar de forma equivalente ambos grupos.

Esquema de pruebas

La Figura 43 muestra el esquema de la prueba previa a la asignación de grupos realizada a los participantes.

```
Nombre

python
1. Para que sirve print()                                   #
2. Diferencia entre tupla, diccionario y lista              #
3. Elemento para concatenar multiples strings               #
4. Que representa la expresión __init__ en una clase        #
5. Cual es el simbolo para instanciar un decorador          #

herramientas
1. que es pip                                               #
3. Que es mqtt                                              #
4. Que es coap                                              #
5. Que es ontología                                         #

hardware
1. Elemento que guarda el SO de una raspberrypi             #
2. Que representa Vcc                                       #
3. Que es GPIO                                              #
4. Cuales son los voltajes de trabajo de la raspberrypi     #
5. Diferencia entre sensor y actuador                       #

total                                                       #
```

Figura 43. Esquema de prueba para signar grupos. Fuente: Propia

Es un total de 15 preguntas con igual peso en la calificación final y están divididas en tres grupos. Grupo de preguntas sobre conocimientos en el lenguaje de programación Python. Grupo de preguntas sobre herramientas variadas, como uso de gestores de paquetes, herramientas IoT y herramientas semánticas. Grupo de preguntas sobre hardware.

Resultados de la prueba y asignación

La Tabla 12 muestra la comparativa de los participantes la calificación final en la prueba y el grupo al cual fueron asignados. La calificación representa el porcentaje preguntas

acertadas (0% - 100%). Los nombres de los participantes no se muestran y son remplazados por “Px”

Participantes	Calificación(%)	Grupo
P1	43	Gb
P2	59	Gb
P3	64	Ga
P4	54	Ga
P5	44	Ga
P6	60	Gb

Tabla 12. Resultado prueba y asignación de grupos. Fuente: Propia

6.2 EJECUCIÓN DEL EXPERIMENTO

El presente apartado presenta la implementación del experimento. Utilizando el diseño del experimento definido en el anterior apartado se ejecuta el experimento. A continuación, se presentan los requerimientos que cada participante debe culminar para implementar una aplicación IoT y luego se muestran los resultados obtenidos por cada participante.

6.2.1 Requerimientos

Para el desarrollo de la aplicación IoT se establece una serie de requerimientos, los cuales conforman en parte la implementación de un OI, pero con ciertas limitaciones. No se asignan todos los requerimientos necesarios para crear un OI debido a que esto aumentaría el tiempo del experimento incurriendo en más limitaciones para los potenciales participantes. A continuación, se listan los requerimientos que deben cumplir los participantes.

Requerimiento módulo recursos

Este requerimiento contiene las conexiones y aprovisionamiento del dispositivo hardware que soporta la aplicación IoT. El dispositivo debe ser una placa de desarrollo basada en arquitectura de microprocesadores. Las actividades de este requerimiento son:

- Aprovisionamiento del dispositivo con sus componentes de carga
- Conexión de por lo menos dos recursos IoT al dispositivo
- Instalación del sistema operativo Linux y el lenguaje de programación Python en el dispositivo

Este requerimiento es realizado de igual forma ambos grupos de comparación, debido a que en este punto el Framework no realiza ninguna implementación.

Requerimiento módulo servicios

Este requerimiento hace referencia a exponer por medio de CoAP la gestión de los metadatos del dispositivo. El usuario debe poder consultar y editar un archivo JSON con

los metadatos que describen al dispositivo y sus recursos. Las actividades de este requerimiento son:

- Generar y persistir un documento JSON el cual contenga los metadatos que describan al dispositivo y a los recursos conectados

- Exponer un servicio CoAP con la cual pueda consultar el documento JSON de los metadatos por medio de un método GET

- Exponer un servicio CoAP con la cual pueda modificar el documento JSON de los metadatos por medio de un método POST

Para este requerimiento el grupo experimental hace uso del Framework y el grupo de control debe realizar estas actividades por sus propios medios.

Requerimiento módulo MQTT

Este requerimiento hace referencia a publicar el estado de los recursos utilizando MQTT. El disipativo debe poder publicar el estado de los recursos conectados de forma periódica en un servidor MQTT. Las actividades de este requerimiento son:

- Generar un tópico MQTT diferente por recurso conectado al dispositivo

- Publicar periódicamente (periodo de tiempo se establece de forma libre por el participante) el estado de los recursos en un servidor MQTT (el servidor se establece de forma libre por el participante)

Para este requerimiento el grupo experimental hace uso del Framework y el grupo de control debe realizar estas actividades por sus propios medios.

Requerimiento módulo ECA

Este requerimiento hace referencia a la gestión de los servicios de interoperabilidad semántica ECA. El usuario debe poder crear un documento JSON con la estructura ECA, donde se defina recurso evento, parámetros de la condición y recurso acción. Las actividades de este requerimiento son:

- Generar un documento JSON el cual contenga los elementos necesarios para generar un servicio de interoperabilidad semántica ECA

- Persistir el documento JSON dentro del dispositivo.

Para este requerimiento el grupo experimental hace uso del Framework y el grupo de control debe realizar estas actividades por sus propios medios.

Requerimiento módulo semántico

Este requerimiento hace referencia a la generación de un índice semántico utilizando una ontología de dominio para filtrar términos mediante la distancia semántica. Las actividades de este requerimiento son:

- Generar una matriz invertida donde se relacionen el peso de términos (palabras) con el dispositivo.

- Los textos candidatos ingresar deben ser pre-procesados, eliminando palabras sin mucho significado (*stop words*) y separando los términos en una lista.

- Filtrar los términos utilizando la distancia semántica entre el término y las clases de la ontología de dominio. Los términos deben tener una distancia semántica menor a un valor preestablecido.

- Los términos que pasen el filtro pueden ser ingresados en la matriz invertida. Sumando al peso del término si este ya se encontraba indexado previamente.

Para este requerimiento el grupo experimental hace uso del Framework y el grupo de control debe realizar estas actividades por sus propios medios.

6.2.2 Resultados

A continuación, son presentados los resultados de cada uno de los participantes en formato de tablas comparativas acompañados de un detalle y un análisis de cada resultado. Además, se presenta el consolidado de resultados por grupo de comparación. Las tablas contienen los resultados de cumplimiento y dificultad de cada requerimiento, además muestra el tiempo invertido en horas expresado por el participante. Los resultados son discutidos con más detalle en el apartado de conclusiones del experimento.

Participante 1 (P1)

La Tabla 13 muestra los resultados obtenidos por el participante 1 (P1)

Participante 1 (P1)			Grupo experimental (Ga)
Requerimiento	Calificacion de requerimientos		Tiempo invertido (horas)
	Cumplimiento (%)	Dificultad	
Módulo de recursos	100	2	1
Módulo de servicios	100	1	1
Módulo MQTT	50	3	1
Módulo EAC	50	3	2
Módulo semántico	100	1	2
Total	80	2	7

Tabla 13. Resultados experimento participante 1. Fuente: Propia

El participante 1 (P1) pertenece al grupo experimental, por lo tanto, para la implementación de los módulos utilizó el Framework. La calificación de cumplimiento obtenida en promedio fue de 80%, lo cual demuestra que los requerimientos fueron cumplidos en gran medida. La calificación de dificultad obtenida en promedio fue de 2, lo cual indica que los requerimientos no representaron mayor dificultad para el participante. El tiempo invertido fue de 7 horas.

Participante 2 (P2)

La Tabla 14 muestra los resultados obtenidos por el participante 2 (P2)

Participante 2 (P2)			Grupo control (Gb)
Requerimiento	Calificacion de requerimientos		Tiempo invertido (horas)
	Cumplimiento (%)	Dificultad	
Módulo de recursos	70	3	4
Módulo de servicios	0	3	0
Módulo MQTT	70	2	4
Módulo EAC	30	4	4
Módulo semántico	0	5	0
Total	34	3,4	12

Tabla 14. Resultados experimento participante 2. Fuente: Propia

El participante 2 (P2) pertenece al grupo de control, por lo tanto, para la implementación de los módulos no utilizó el Framework. La calificación de cumplimiento obtenida en promedio fue de 34%, lo cual demuestra que los requerimientos no fueron cumplidos o fueron poco satisfactorios. La calificación de dificultad obtenida en promedio fue de 3.4, lo cual indica que los requerimientos no representaron mayor dificultad para el participante. El tiempo invertido fue de 12 horas. En el módulo de servicios y el módulo semántico de este participante se nota una anomalía, en donde los valores para el cumplimiento y tiempo invertido es 0. Esta anomalía se debe a que el participante no realizó dicho módulo debido a problemas de carácter personal. Estos valores son promediados ya que las variables independientes de carácter humano que inciden sobre los resultados objetivos, deben ser evaluadas e incluidas en el análisis del experimento.

Participante 3 (P3)

La Tabla 15 muestra los resultados obtenidos por el participante 3 (P3)

Participante 3 (P3)			Grupo control (Gb)
Requerimiento	Calificacion de requerimientos		Tiempo invertido (horas)
	Cumplimiento (%)	Dificultad	
Módulo de recursos	70	2	4
Módulo de servicios	100	4	5
Módulo MQTT	100	2	5
Módulo EAC	100	5	5
Módulo semántico	0	5	0
Total	74	3,6	19

Tabla 15. Resultados experimento participante 3. Fuente: Propia

El participante 3 (P3) pertenece al grupo de control, por lo tanto, para la implementación de los módulos no utilizó el Framework. La calificación de cumplimiento obtenida en promedio fue de 74%, lo cual demuestra que los requerimientos fueron cumplidos en gran medida. La calificación de dificultad obtenida en promedio fue de 3.6, lo cual indica que los requerimientos representaron una dificultad considerable para el participante. El tiempo invertido fue de 19 horas.

Participante 4 (P4)

El participante 4 (P4) abandonó el experimento alegando falta de tiempo, por lo tanto, es anotado este inconveniente en el presente documento con fines de análisis, pero los resultados no son ponderados al final.

Participante 5 (P5)

La Tabla 16 muestra los resultados obtenidos por el participante 5 (P5)

Participante 5 (P5)			Grupo experimental (Ga)
Requerimiento	Calificacion de requerimientos		Tiempo invertido (horas)
	Cumplimiento (%)	Dificultad	
Módulo de recursos	100	2	1
Módulo de servicios	100	1	2
Módulo MQTT	50	3	1
Módulo EAC	100	2	1
Módulo semántico	100	1	2
Total	90	1,8	7

Tabla 16. Resultados experimento participante 5. Fuente: Propia

El participante 5 (P5) pertenece al grupo experimental, por lo tanto, para la implementación de los módulos utilizó el Framework. La calificación de cumplimiento obtenida en promedio fue de 90%, lo cual demuestra que los requerimientos fueron cumplidos en gran medida. La calificación de dificultad obtenida en promedio fue de 1.8, lo cual indica que los requerimientos no representaron mayor dificultad para el participante. El tiempo invertido fue de 7 horas.

Participante 6 (P6)

La Tabla 17 muestra los resultados obtenidos por el participante 6 (P6)

Participante 6 (P6)			Grupo control (Gb)
Requerimiento	Calificacion de requerimientos		Tiempo invertido (horas)
	Cumplimiento (%)	Dificultad	
Módulo de recursos	100	2	2
Módulo de servicios	100	3	2
Módulo MQTT	100	1	2
Módulo EAC	100	3	2
Módulo semántico	40	4	1
Total	88	2,6	9

Tabla 17. Resultados experimento participante 6. Fuente: Propia

El participante 6 (P6) pertenece al grupo de control, por lo tanto, para la implementación de los módulos no utilizó el Framework. La calificación de cumplimiento obtenida en promedio fue de 88%, lo cual demuestra que los requerimientos fueron cumplidos en gran medida. La calificación de dificultad obtenida en promedio fue de 2.6, lo cual indica que los requerimientos no representaron mayor dificultad para el participante. El tiempo invertido fue de 9 horas.

Grupo experimental (Ga)

La Tabla 18 muestra los resultados obtenidos por el grupo experimental (Ga)

Grupo Experimental (Ga)			
Requerimiento	Calificacion de requerimientos		Tiempo invertido (horas)
	Cumplimiento (%)	Dificultad	
Módulo de recursos	100	2	1
Módulo de servicios	100	1	1,5
Módulo MQTT	50	3	1
Módulo EAC	75	2,5	1,5
Módulo semántico	100	1	2
Total	85	1,9	7

Tabla 18. Resultados experimento Ga. Fuente: Propia

Para el grupo experimental (Ga) tenemos los resultados promediados entre los participantes por requerimiento. La calificación de cumplimiento obtenida en promedio fue de 85%, lo cual demuestra que los requerimientos fueron cumplidos en gran medida por los participantes del grupo. La calificación de dificultad obtenida en promedio fue de 1.9, lo cual indica que los requerimientos no representaron mayor dificultad para los participantes. El tiempo invertido promedio por todos los participantes fue de 7 horas.

Grupo de control (Gb)

La Tabla 19 muestra los resultados obtenidos por el grupo de control (Gb)

Grupo Control (Gb)			
Requerimiento	Calificacion de requerimientos		Tiempo invertido (horas)
	Cumplimiento (%)	Dificultad	
Módulo de recursos	80	2,3	3,3
Módulo de servicios	66,7	3,3	3,5
Módulo MQTT	90	1,7	3,7
Módulo EAC	76,7	4	3,7
Módulo semántico	13,3	4,7	1
Total	65,3	3,2	15,2

Tabla 19. Resultados experimento Gb. Fuente: Propia

Para el grupo de control (Gb) tenemos los resultados promediados entre los participantes por requerimiento. La calificación de cumplimiento obtenida en promedio fue de 65%, lo cual demuestra que los requerimientos no fueron cumplidos o fueron poco satisfactorios. La calificación de dificultad obtenida en promedio fue de 3.2, lo cual indica que los requerimientos representaron mediana dificultad para los participantes. El tiempo invertido promedio por todos los participantes fue de 15.2 horas.

6.3 CONCLUSIONES DE LA EXPERIMENTACIÓN

El presente apartado muestra las conclusiones finales que los autores del documento establecieron a parir del análisis de los resultados.

La Figura 44 refiere a la calificación del cumplimiento, nos muestra una comparativa, de los dos grupos y los resultados obtenidos en los diferentes requerimientos y adicionalmente una comparativa entre los resultados totales.

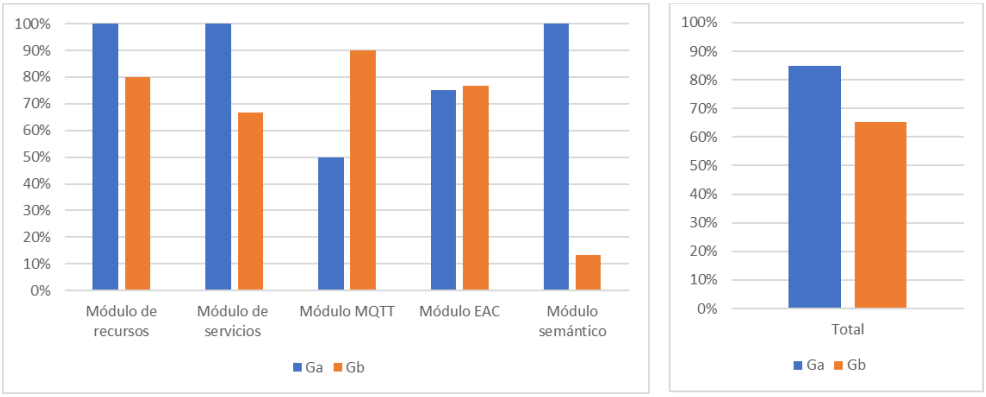


Figura 44. Comparativa resultados de cumplimiento. Fuente: Propia

Es posible notar que el grupo experimental (Ga) obtiene resultados más altos en la mayoría de los requerimientos, en especial el requerimiento módulo semántico, esta alta diferencia se debe a que este requerimiento es el más complejo. Al final el promedio total del grupo experimental (Ga) resulta ser más alto que del grupo de control (Gb)

La Figura 45 refiere a la calificación de la dificultad, nos muestra una comparativa, de los dos grupos y los resultados obtenidos en los diferentes requerimientos y adicionalmente una comparativa entre los resultados totales.

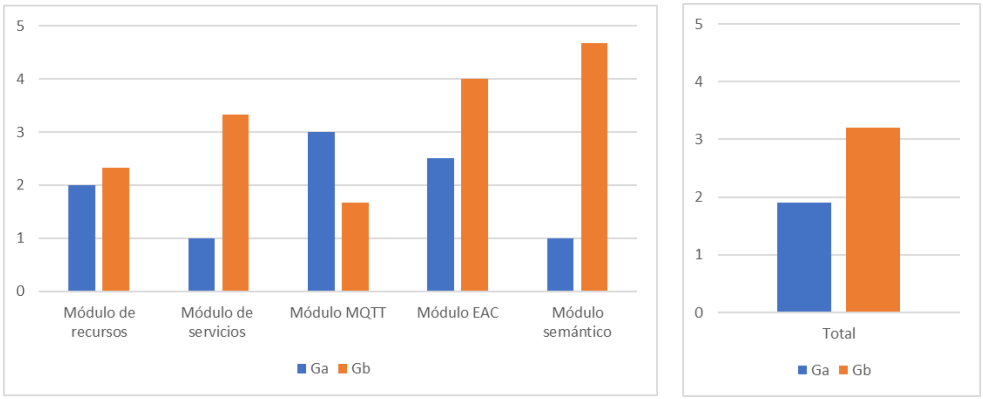


Figura 45. Comparativa resultados de dificultad. Fuente: Propia

Es posible notar que el grupo experimental (Ga) obtiene resultados más bajos en la mayoría de los requerimientos, en especial los requerimientos módulo semántico y módulo de servicios, esta diferencia tan grande en este requerimiento se debe a que las herramientas semánticas y el servicio CoAP son administradas en gran medida por el Framework, entregado una capa muy simplificada de estas a los usuarios. Al final el promedio total del grupo experimental (Ga) resulta ser más bajo que del grupo de control (Gb)
Si hacemos una comparación entre los módulos de los resultados de cumplimiento con los resultados de dificultad, podemos ver una correlación para el grupo de control (Gb), mientras el cumplimiento es de un menor porcentaje la dificultad expresada por los participantes es mayor, esto se hace muy claro en el módulo semántico, el cual contiene la mayor complejidad, por ende, su cumplimiento fue menor, y la dificultad mayor. Esta correlación no aplica para el grupo experimental (Ga) ya que el estímulo (X) el Framework representa una capa que permite simplificar la resolución de los requerimientos.

Para el grupo experimental (Ga), la comparación entre los módulos de los resultados de cumplimiento con los resultados de dificultad, es una medida del nivel de reducción de esfuerzo que el Framework entrega por módulo. En este sentido, el módulo de servicios y el módulo semántico son los que entregan un mayor nivel de abstracción y con ello la reducción del esfuerzo, ya que tienen un cumplimiento alto de los requerimientos y una percepción de la dificultad muy baja.

La Figura 46 refiere al tiempo (en horas) invertido en desarrollador los diferentes requerimientos, nos muestra una comparativa, de los dos grupos y los resultados obtenidos en los diferentes requerimientos y adicionalmente una comparativa entre los resultados totales.

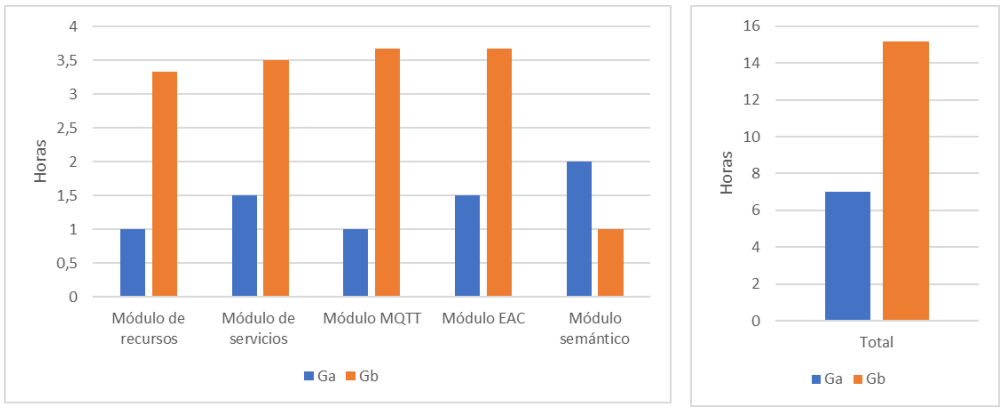


Figura 46. Comparativa resultados tiempo invertido (Horas). Fuente: Propia

Es posible notar que el grupo experimental (Ga) obtiene resultados más bajos en la mayoría de los requerimientos, esto se debe a que el Framework entrega una capa de configuración para implementar la aplicación IoT, lo que impacta al tiempo de desarrollo. Al final el promedio total del grupo experimental (Ga) resulta ser más bajo que del grupo de control (Gb)

A partir de las conclusiones sobre cada factor de medida podemos determinar lo siguiente:

- Tiempo. El grupo experimental (Ga) tiene un menor nivel de tiempo invertido en la construcción de la aplicación IoT que el grupo de control (Gb)

- Cumplimiento. El grupo experimental (Ga) tiene un mayor nivel de cumplimiento de requerimientos para la construcción de la aplicación IoT que el grupo de control (Gb)

- Dificultad. El grupo experimental (Ga) tiene un menor nivel de dificultad en la construcción de la aplicación IoT que el grupo de control (Gb)

Si tomamos en cuenta las relaciones que tiene el esfuerzo con las medidas de tiempo, cumplimiento y dificulta podemos determinar que el esfuerzo del grupo experimental (Ga) resulta ser menor al esfuerzo del grupo de control (Gb)

$$Esfuerzo\ Ga < Esfuerzo\ Gb$$

Con esto podemos concluir que: utilizar el Framework para construir una aplicación IoT entre ellas, un ambiente de interoperabilidad semántica de OI, resulta en un menor esfuerzo utilizando el Framework propuesto.

7 CONCLUSIONES Y TRABAJO FUTURO

Este capítulo describe inicialmente las principales conclusiones del trabajo realizado a las que se llegó durante su desarrollo y posteriormente se propone los trabajos futuros.

7.1 CONCLUSIONES

A continuación, se listan las principales conclusiones del proyecto:

- El estudio bibliométrico revela que para el último periodo de tiempo los conceptos “IoT” “Tratamiento del Lenguaje Natural” son conceptos que conforman nuevos estudios acoplados al concepto de “Web Semántica” Con esto podemos concluir que existe un mayor interés por explotar el trabajo conjunto de estas áreas del conocimiento.

- El Framework propuesto, establece mecanismos para gestionar la información de metadatos de los OI soportado en recomendaciones internacionales. Esto incentiva el correcto apropiamiento generalizado. El uso de recomendaciones permite avanzar más rápido en las soluciones y aseguran el uso posterior de las mismas por la comunidad investigativa.

- El diseño de los experimentos, su ejecución y el análisis, sugieren que el Framework entrega valor a los usuarios desarrolladores, haciendo que el esfuerzo en la implementación de ambientes donde OI puedan interoperar semánticamente sea menor.

- Se presentaron anomalías en la ejecución del experimento, como valores de cumplimiento y tiempo invertido en cero (0). Esto se explica con variables independientes de tipo humano que entran a hacer parte del experimento, como, problemas de carácter personal, falta de tiempo para la inversión en el desarrollo de los requerimientos, debido a otras prioridades. Sin embargo, estos valores son promediados ya que las variables independientes de carácter humano que inciden sobre los resultados objetivos, deben ser evaluadas e incluidas en el análisis del experimento.

- El Framework puede crear otro tipo de aplicaciones IoT, diferentes al propuesto en este proyecto. Esta capacidad la brinda el nivel de flexibilidad que maneja el Framework en su capa de configuración, la cual permite habilitar y deshabilitar sus elementos internos.

- Hay que anotar que la evaluación al tener un diseño experimental con una sola preprueba, nos da una tendencia a que la hipótesis sea cierta y esto hace necesario al desarrollo de más experimentos y pruebas de mayor alcance para ver la utilidad efectiva del Framework.

7.2 TRABAJOS FUTUROS

A continuación, se listan los principales trabajos futuros:

- Es posible adicionar una ontología o herramienta semántica auxiliar, la cual me ayude guardando los servicios de interoperabilidad semántica ECA, con el fin de realizar operaciones de razonamiento autónomo sobre las relaciones que contiene el ECA y brindar al ambiente mayores capacidades de autonomía, haciendo que el usuario intervenga en menor medida.

- Implementar una interfaz gráfica para realizar la configuración, esto con el fin de mejorar la experiencia de usuario y ampliar la población añadiendo usuarios menos técnicos.

- Definir una capa de seguridad para que el Framework pueda exponer la posibilidad de generar procesos de autenticación, autorización, sesión y demás servicios de seguridad informática.

- Acoplar en la capa de descubrimiento de OI, nuevas ontologías de dominio para expandir las funcionalidades que entrega el Framework a nuevos contextos.

- Mejorar el componente Crawler CoAP para que sea capaz de buscar de mejores formas OI, tanto en la red local como en internet.

7.3 PUBLICACIONES

A continuación, se listan las publicaciones realizadas a partir del presente proyecto

- Guerrero-Narváez, S.; Niño-Zambrano, M.-Á.; Riobamba-Calvache, D.-J.; Ramírez-González, G.-A. Test Bed of Semantic Interaction of Smart Objects in the Web of Things. Future Internet 2018, 10, 42. https://doi.org/10.3390/fi10050042 ISSN 1999-5903 SJR Q2

- Narváez, S. G., Zambrano, M. Á. N., & González, G. A. R. (2020). Mapeando la evolución investigativa de la web semántica: análisis y visualización cienciométrica. Revista Ibérica de Sistemas e Tecnologias de Informação, (E38), 325-336. http://www.risti.xyz/issues/ristie38.pdf ISSN 1646-9895 Publindex B

8 CUMPLIMIENTO DE OBJETIVOS

8.1 CONFORMACIÓN E INTERPRETACIÓN DE LOS INDICADORES

Con el fin de expresar los resultados finales de cada uno de los objetivos se presenta a continuación una explicación sencilla de los tipos de indicadores utilizados en la evaluación de los resultados y la forma correcta de interpretarlos.

Los indicadores de desempeño que se evalúan, básicamente adoptan la forma de un cociente, en el cual, el denominador es un valor numérico que ayuda a efectuar la comparación con el logro obtenido. De esta forma se definen los siguientes modelos de indicadores que se deben personalizar y aplicar a los actores, productos, funciones, etc. dependiendo del contexto del objetivo evaluado:

Indicador de Eficiencia.
Permite identificar la relación que existe entre las metas alcanzadas, el tiempo y los recursos consumidos con respecto a un estándar. Representa el buen uso de los recursos.

$$Eficiencia = \left(\frac{Metas\ alcanzadas}{Recursos\ consumidos}\right) * 100\%$$

Indicador de Calidad
Están orientados a medir la satisfacción de los beneficiarios. La calificación está discriminada en: 1: mala (0%), 2: regular (50%), 3: buena (75%) y 4: excelente (100%)

Con el modelo de indicadores aquí presentado, se desarrolló un conjunto de indicadores que permiten evaluar adecuadamente el nivel de cumplimiento de cada uno de los objetivos.

8.2 DESCRIPCIÓN Y ALCANCE DEL CUMPLIMIENTO DE LOS OBJETIVOS

Para el cumplimiento de los objetivos se presenta un modelo de indicadores que permite evaluar de manera objetiva el cumplimiento de los mismos. A continuación, se especifican, de arriba hacia abajo, los objetivos comprometidos en el proyecto, los productos esperados derivados de cada objetivo, los resultados obtenidos, los indicadores que evalúan el objetivo, los medios de verificación de los resultados y finalmente, unas observaciones que permiten aclarar los resultados en cada objetivo. Una tabla por cada objetivo específico

Objetivo	1
Descripción	Crear un marco conceptual que identifique los elementos más importantes para la implementación de entornos de interoperabilidad semántica de OI para la WoT
Productos	a. Estudio bibliométrico en torno al concepto principal del proyecto.

	b. Marco teórico por núcleos temáticos. c. Publicación de los resultados de este objetivo de forma oral o escrita.
Resultados	a. En el Capítulo 3 apartado 3.1, podemos encontrar el mapeo científico que se realizó sobre el concepto "*semantic web*" este proceso es un estudio bibliométrico desarrollado utilizando para esto dos herramientas y dos bases de datos. b. En el Capítulo 2 está dedicado a definir el marco teórico definiendo varios conceptos organizados por núcleos temáticos. c. Se realizó una publicación en una revista científica sobre el mapeo científico del concepto "*semantic web*"
Indicadores	**Eficiencia** $Eficiencia = \frac{número\ productos\ resultado}{número\ productos\ requeridos} * 100\% = \frac{3}{3} * 100\% = 100\%$ **Calidad** a. El estudio bibliométrico fue realizado sobre un solo concepto principal del proyecto, aunque este es el concepto considerado de mayor centralidad en el proyecto se omiten dos conceptos. $Calidad = 3 = 75\%$ b. El marco teórico encierra los 3 conceptos centrales del proyecto $Calidad = 4 = 100\%$ c. La publicación se realizó sobre el primer producto y se omite el segundo $Calidad = 3 = 75\%$
Problemas y/o observaciones	- El estudio bibliométrico fue realizado sobre un concepto omitiendo los otros dos conceptos centrales del proyecto. Esta decisión se toma debido a que en la realización del estudio sobre el concepto "semantic web" se encontraron fuertemente relacionados los otros conceptos centrales del proyecto, haciendo que el estudio bibliométrico sobre estos se torne redundante. - El estudio bibliométrico fue publicado en una revista científica clase C bajo el nombre de "Mapeando la evolución investigativa de la web semántica: análisis y visualización cienciometría" - Para el estudio del estado del arte en cuanto a trabajos relacionados, se observó que las propuestas relacionadas con Frameworks de desarrollo son escasas en el ámbito científico, conformando estas propuestas un nicho interesante.

Tabla 20. Cumplimiento del primer objetivo específico. Fuente: Propia

Objetivo	2
Descripción	Implementar un conjunto de librerías en un lenguaje de programación especifico y basado en el marco conceptual, que apoyen el desarrollo de entornos de interoperabilidad semántica de OI para la WoT
Productos	a. Arquitectura de soporte basada en un referente teórico. b. Framework a nivel conceptual basado en la arquitectura de soporte. c. Framework a nivel lógico. Librerías, comandos y manuales.

Resultados	a. En el Capítulo 4 apartado 4.1, podemos encontrar la definición en varios niveles de la arquitectura de soporte. b. En el Capítulo 4 apartado 4.2, podemos encontrar la definición a nivel conceptual del Framework, este se encuentra fuertemente basado en la arquitectura de soporte. c. En el Anexo C, se comparte el acceso al repositorio donde se encuentra alojado el código del Framework, además en el Anexo C se exponen los manuales y los comandos necesarios para implementar el Framework
Indicadores	**Eficiencia** $Eficiencia = \frac{número\ productos\ resultado}{número\ productos\ requeridos} * 100\% = \frac{3}{3} * 100\% = 100\%$ **Calidad** a. Es definida una arquitectura de soporte con todos los elementos y condiciones necesarias para crear el Framework. $Calidad = 4 = 100\%$ b. Es definido un Framework con todos los elementos necesarios para implementar un escenario de OI que puedan realizar servicios de interoperabilidad semántica con herramientas de la WoT $Calidad = 4 = 100\%$ c. Es construido un Framework a nivel lógico que cumple con todos los parámetros conceptuales, entregando los archivos y documentos necesarios para ejecutarlo de forma abierta en un repositorio de internet $Calidad = 4 = 100\%$
Problemas y/o observaciones	- Para algunos procesos de tratamiento de lenguaje natural que se realiza dentro del elemento descubrimiento de OI, fue necesario incorporar unas herramientas auxiliares para traducir los términos y poderlos comparar con la ontología de dominio - La recomendación Thing Description de la W3C ha sufrido dos actualizaciones mientras se realizaba el presente proyecto. Esto obligo a actualizar algunos elementos en el Framework. - Python sufrió la actualización a la versión 3.8, mientras se realizaba el presente proyecto. Esto obligó a migrar el código fuente del Framework.

Tabla 21. Cumplimiento del segundo objetivo específico. Fuente: Propia

Objetivo	3
Descripción	Realizar una prueba de concepto para el caso de estudio en domótica, mediante la implementación de un entorno IoT, con al menos tres OI utilizando el Framework propuesto
Productos	a. Diagramas UML de requisitos. b. Componentes software y hardware de los OI. c. Componentes de interoperabilidad de los OI. d. Componentes semánticos de los OI
Resultados	a. En el Capítulo 5 apartado 5.1 y el apartado 5.2, podemos encontrar los diagramas UML de requerimientos simplificados y los formatos expandidos de los requerimientos.

	b. En el Capítulo 5 apartado 5.3.1, podemos encontrar la primera iteración del desarrollo, en esta fase se construyen los componentes software y hardware de los OI c. En el Capítulo 5 apartados 5.3.2 y el apartado 5.3.3, podemos encontrar la segunda y tercera iteración del desarrollo, en estas fases se construyen los componentes que conforman la interoperabilidad de los OI d. En el Capítulo 5 apartado 5.3.3, podemos encontrar la tercera iteración del desarrollo, en esta fase se construye los componentes semánticos del OI
Indicadores	**Eficiencia** $Eficiencia = \frac{número\ productos\ resultado}{número\ productos\ requeridos} * 100\% = \frac{4}{4} * 100\% = 100\%$ **Calidad** a. Son entregados los diagramas UML donde se expresan los casos de uso necesarios para implementar un OI utilizando Framework. Se omiten otros diagramas UML como el de clases y el de secuencia. $Calidad = 3 = 75\%$ b. Los componentes hardware y software son generados con éxito utilizando el Framework $Calidad = 4 = 100\%$ c. Los componentes de interoperabilidad son generados con éxito utilizando el Framework $Calidad = 4 = 100\%$ d. Los componentes semánticos son generados con éxito utilizando el Framework $Calidad = 4 = 100\%$
Problemas y/o observaciones	- Algunos diagramas UML son omitidos de este objetivo, como diagramas de clases y diagramas de secuencia. Esta decisión fue tomada debido a que estos diagramas dilatan el trabajo de implementación y solo entregan entendimiento en la implementación de la aplicación mas no entregan un alto valor en el entendimiento del Framework. - La prueba de concepto resultante fue publicada en una revista Clase A2 científica bajo el nombre de “Test Bed of Semantic Interaction of Smart Objects in the Web of Things” - Es necesaria la constante evaluación y modificación de los elementos de los OI con el fin de alcanzar mitigar bugs y mejorar tiempos de respuestas - Existen el problema de coalición entre ECA, esto es causado cuando dos o más ECA tienen el mismo recurso acción. Esto genera incongruencias en el funcionamiento de los recursos, a nivel funcional

Tabla 22. Cumplimiento del tercer objetivo específico. Fuente: Propia

Objetivo	4
Descripción	Evaluar el esfuerzo invertido para la implementación de entornos de interoperabilidad semántica de OI para WoT, mediante la comparación del desarrollo utilizando el Framework y sin él.

Productos	a. Plan de pruebas. Definir un experimento para constatar el valor agregado del Framework. b. Ejecución del plan de pruebas. Poner en marcha el experimento previamente diseñado. c. Análisis de resultados
Resultados	a. En el Capítulo 6 apartado 6.1, podemos encontrar el diseño de un experimento para medir el esfuerzo del Framework b. En el Capítulo 6 apartado 6.2, podemos encontrar los requerimientos y los resultados obtenidos del experimento, conformados por la ejecución del mismo. c. En el capítulo 6 apartado 6.3 podemos encontrar las conclusiones sobre los resultados obtenidos en el experimento.
Indicadores	**Eficiencia** $Eficiencia = \frac{número\ productos\ resultado}{número\ productos\ requeridos} * 100\% = \frac{3}{3} * 100\% = 100\%$ **Calidad** a. Es definido un plan de pruebas, describiendo la naturaleza del experimento y sus componentes, se omite una pila de pruebas en esa sección. $Calidad = 3 = 75\%$ b. El plan de pruebas es ejecutado con éxito y son ejercitados todos los requerimientos establecidos por el plan de pruebas $Calidad = 4 = 100\%$ c. Se realiza un análisis de resultados a partir de la construcción de gráficas de los datos entregados por módulo $Calidad = 4 = 100\%$
problemas y/o observaciones	- Los participantes del experimento que utilizaron el Framework, reportaron varios errores, los cuales fueron corregidos. - Se descubrió que el Framework puede ser ejecutado en sistemas operativos Windows, con leves cambios al código fuente

Tabla 23. Cumplimiento del cuarto objetivo específico. Fuente: Propia

9 BIBLIOGRAFÍA

[1] S. Mathew, Y. Atif, Q. Z. Sheng y Z. Maamar, «Web of things: Description, discovery and integration. In Internet of Things (iThings/CPSCom),» de *International Conference on and 4th International Conference on Cyber, Physical and Social Computing (pp. 9-15). IEEE.*, 2011.

[2] F. Scioscia y M. Ruta, «Building a Semantic Web of Things: issues and perspectives in information compression,» de *IEEE International Conference*, 2009.

[3] T. Teixeira, S. Hachem, V. Issarny y N. & Georgantas, «Service oriented middleware for the internet of things: A perspective,» de *Towards a Service-Based Internet*, Springer Berlin Heidelberg., 2011, pp. 220-229.

[4] L. Atzori, A. Iera y G. Morabito, «The internet of things: A survey,» *Computer networks,* vol. 54, nº 15, pp. 2787-2805, 2010.

[5] A. Banks y R. Gupta, «MQTT Version 3.1. 1.,» 29 Octubre 2014. [En línea]. Available: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html. [Último acceso: 7 Enero 2016].

[6] M. Zhou, H. Fan y Y. Ma, «Semantic annotation method of IOT middleware,» de *2013 Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, Beijing, 2013.

[7] C. Michael, B. Payam, B. Luis, G.-C. Raúl, C. Oscar, C. Simon, G. John, H. Manfred, H. Cory, H. Arthur, H. Vincent, J. Krzysztof, K. W. David, L. P. Danh y L. Laurent, «The SSN ontology of the W3C semantic sensor network incubator group,» *Journal of Web Semantics,* vol. XVII, pp. 25-32, 2012.

[8] B. Michael y L. Rodger, «IoT Interoperability: A Hub-based Approach,» de *2014 International Conference on the Internet of Things (IOT)*, Cambridge, 2014.

[9] W. Jiafu, L. Di, Z. Caifeng y Z. Keliang, «M2M Communications for Smart City: An Event-Based Architecture,» de *2012 IEEE 12th International Conference on Computer and Information Technology*, Chengdu, 2012.

[10] W. Wei, D. Suparna, T. Ralf, R. Eike y M. Klaus, «A Comprehensive Ontology for Knowledge Representation in the Internet of Things,» de *2012 IEEE 11th International Conference on Trust, Security and Privacy in Computing and Communications*, Liverpool, 2012.

[11] S. K., M. T. T., T. Tina Esther y K. R., «Integration of semantics, sensors and services on the ubiquitous web,» de *2012 International Conference on Recent Trends in Information Technology*, Chennai, 2012.

[12] A. Charu C. y A. Tarek, «Social Sensing,» de *Managing and Mining Sensor Data*, Boston, Springer, 2013.

[13] C. Perera, A. Zaslavsky, P. Christen y D. Georgakopoulos, «Context aware computing for the internet of things: A survey,» *Communications Surveys & Tutorials, IEEE,* vol. 16, nº 1, pp. 414-454, 2014.

[14] S. Guerrero y D. Riobamba, «Escenario de interacción semántica de objetos inteligentes en la web de las cosas,» Universidad del Cauca, Popayán, 2016.

[15] M. L. Barry, G. C. Vinton y D. C. David, «A Brief History of the Internet,» 23 Junio 1999. [En línea]. Available: http://arxiv.org/html/cs/9901011. [Último acceso: 7 1 2015].

[16] R. Piyare y S. R. Lee, «TOWARDS INTERNET OF THINGS (IOTS) INTEGRATION OF WIRELESS SENSOR NETWORK TO CLOUD SERVICES FOR DATA COLLECTION AND SHARING,» *International Journal of Computer Networks & Communications (IJCNC),* vol. 5, nº 5, pp. 59-72, 2013 .

[17] D. Bandyopadhyay y J. Sen, «Internet of things: Applications and challenges in technology and standardization,» *Wireless Personal Communications,* vol. I, nº 58, pp. 49-69, 2011.

[18] S. Bandyopadhyay, M. Sengupta, S. Maiti y S. Dutta, « Role of middleware for internet of things: A study,» *International Journal of Computer Science & Engineering Survey (IJCSES),* vol. 2, nº 3, pp. 94-105, 2011.

[19] Arduino, «Arduino,» [En línea]. Available: https://www.arduino.cc/. [Último acceso: 17 1 2016].

[20] Intel, «Intel Galileo Board,» [En línea]. Available: http://ark.intel.com/products/78919/Intel-Galileo-Board. [Último acceso: 7 Enero 2016].

[21] BeagleBone, «BeagleBone,» [En línea]. Available: http://beagleboard.org/bone. [Último acceso: 17 1 2016].

[22] Raspberrypi, «Raspberrypi,» [En línea]. Available: https://www.raspberrypi.org/. [Último acceso: 17 1 2016].

[23] Raspberry Pi Foundation, «FrontPage - Raspbian,» [En línea]. Available: https://www.raspbian.org. [Último acceso: 23 Enero 2021].

[24] S. S. Mathew, Y. Atif, Q. Z. Sheng y Z. Maamar, «Web of things: Description, discovery and integration,» de *International Conference on and 4th International Conference on Cyber, Physical and Social Computing*, 2011.

[25] Y. Liu y G. Zhou, «Key technologies and applications of internet of things,» de *Intelligent Computation Technology and Automation (ICICTA), 2012 Fifth International Conference*, IEEE, 2012, pp. 197-200.

[26] J. Cooper y A. & James, «Challenges for database management in the internet of things,» *IETE Technical Review,* vol. 5, nº 26, pp. 320-329, 2009.

[27] D. C. Mansilla, M. V. Barbas y M. T. L. Merayo, «Infraestructuras Inteligentes en el Internet del Futuro,» de *Actas del I Encuentro de Investigadores en Infraestructuras Inteligentes (EI3 2011)*, Guadalajara, Servicio de Publicaciones, 2011, pp. 97-100.

[28] G. Kortuem, F. Kawsar, D. Fitton y V. Sundramoorthy, «Smart objects as building blocks for the internet of things,» *Internet Computing IEEE,* vol. 14, nº 1, pp. 44-51, 2010.

[29] J. Pascoe, «Adding generic contextual capabilities to wearable computers,» de *Second International Symposium*, IEEE, 1998, pp. 92-99.

[30] C. Perera, P. P. Jayaraman, A. Zaslavsky, D. Georgakopoulos y P. Christen, «Sensor discovery and configuration framework for the internet of things paradigm,» *Internet of Things (WF-IoT), 2014 IEEE World Forum,* pp. 94-99, 2014.

[31] C. Perera, P. P. Jayaraman, A. Zaslavsky, P. Christen y D. Georgakopoulos, «Context-Aware Dynamic Discovery and Configuration of 'Things' in Smart Environments,» *Big Data and Internet of Things: A Roadmap for Smart Environments,* pp. 215-241, 2014.

[32] M. M. N. Hadley, J. Moreau, H. Nielsen y M. Gudgin, «SOAP Version 1.2 Part 1: Messaging Framework, W3C Recommendation,» 27 Abril 2007. [En línea]. Available: http://www.w3.org/TR/2007/REC-soap12-part1-20070427/. [Último acceso: 7 Enero 2016].

[33] Z. Shelby, K. Hartke y C. Bormann, «The Constrained Application Protocol (CoAP),» Junio 2014. [En línea]. Available: https://tools.ietf.org/html/rfc7252. [Último acceso: 17 1 2016].

[34] P. Saint-Andre, «Extensible Messaging and Presence Protocol (XMPP),» Octubre 2004. [En línea]. Available: https://tools.ietf.org/html/rfc3920. [Último acceso: 17 1 2016].

[35] K. Sebastian, K. Takuki, M. Michael, C. Victor y K. Matthias, «Web of Things (WoT) Thing Description,» [En línea]. Available: https://www.w3.org/TR/wot-thing-description/. [Último acceso: 27 Enero 2021].

[36] T. Berners-Lee, J. Hendler y O. Lassila, «The semantic web,» *Scientific american,* vol. 284, nº 5, pp. 28-37, 2001.

[37] S. Suárez, R. Martínez y P. Redondo, «Biblioteca semántica de webquest,» 2004. [En línea]. Available: LuceneTutorial.com. [Último acceso: 7 1 2015].

[38] M. Gudgin, M. Hadley, N. Mendelsohn, J. J. Moreau, H. F. Nielsen, A. Karmarkar y Y. & Lafon, *SOAP Version 1.2,* W3C recommendation 24, 2003.

[39] J. e. Cardoso, Semantic Web Services: Theory, Tools and Applications: Theory, Tools and Applications, IGI Global, 2007.

[40] M. Suárez Barón y K. Salinas Valencia, «An approach to semantic indexing and information retrieval,» *Revista Facultad de Ingeniería Universidad de Antioquia,* vol. 48, pp. 174-187, 2009.

[41] T. R. Gruber, «Toward principles for the design of ontologies used for knowledge sharing,» *International journal of human-computer studies,* vol. 43, nº 5, pp. 907-928, 1995.

[42] D. Oberle, «Semantic management of middleware,» *Proceedings of the 1st international doctoral symposium on Middleware,* pp. 299-303, 204.

[43] P. D. Karp, V. K. Chaudhri y J. Thomere, «XOL: An XML-based ontology exchange language,» 1999. [En línea]. Available: http://xml.coverpages.org/xol-03.html. [Último acceso: 7 Enero 2016].

[44] D. Brickley y R. V. Guha, «RDF vocabulary description language 1.0: RDF schema,» 2004. [En línea]. Available: http://www.w3.org/TR/PR-rdf-schema. [Último acceso: 7 Enero 2016].

[45] D. Fensel, I. Horrocks, F. Van Harmelen, S. Decker, M. Erdmann y M. Klein, «OIL in a nutshell,» *Knowledge Engineering and Knowledge Management Methods, Models, and Tools,* pp. 1-16, 2000.

[46] F. P.-S. P. F. van Harmelen y I. Horrocks, «Reference description of the DAML+OIL (March 2001) ontology markup language,» Mayo 2001. [En línea]. Available: http://www.daml.org/2001/03/reference.html. [Último acceso: 7 Enero 2016].

[47] D. C. M. Dean, F. v. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider y L. Stein, «OWL Web Ontology Language 1.0 Reference, W3C Recommendation,» 10 Febrero 2004. [En línea]. Available: http://www.w3.org/TR/owl-ref/. [Último acceso: 7 Enero 2016].

[48] W. Zhibiao y P. Martha, «Verbs semantics and lexical selection,» de *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*, Association for Computational Linguistics, 1994, pp. 133-138.

[49] C. Chen, «CiteSpace II: Detecting and visualizing emerging trends and transient patterns in scientific literature,» *Journal of the American Society for Information Science and Technology,* vol. III, nº 57, pp. 1532-2882, 2006.

[50] M. Cobo, A. López-Herrera, E. Herrera-Viedma y F. Herrera, «SciMAT: A new science mapping analysis software tool,» *Journal of the American Society for Information Science and Technology,* vol. VIII, nº 63, pp. 1532-2882, 2012.

[51] C. Hoyos Botero, Un modelo para investigación documental: guía teórico-práctica sobre construcción de Estados del Arte con importantes reflexiones sobre la investigación, Medellín: Señal Editora, 2000.

[52] C.-E. Serrano-Castaño, «Modelo para la Investigación Documental,» de *Modelo Integral para el Profesional en Ingeniería*, Popayán, Universidad del Cauca, 2005, p. 147.

[53] B. R. N., «Toward a definition of “bibliometrics”,» *Scientometrics,* nº 12, p. 373–379, 1987.

[54] B.-I. Judit, «Which h-index? — A comparison of WoS, Scopus and Google Scholar,» *Scientometrics,* nº 74, p. 257–271, 2008.

[55] G. Ylva y I. Lars, «Web of Science and Scopus: a journal title overlap study,» *Online Information Review,* vol. XXXII, nº 1, pp. 8-21, 2008.

[56] S. Jinbo, Z. Honglian y D. Wanli, «A review of emerging trends in global PPP research: analysis and visualization,» *Scientometrics,* nº 107, p. 1111–1147, 2016.

[57] J. Gómez, J. F. Huete, O. Hoyos, L. Perez y D. Grigori, «Interaction System based on Internet of Things as Support for Education,» *Procedia Computer Science,* vol. 21, pp. 132-139, 2013.

[58] F. Kawsar, G. Kortuem y B. Altakrouri, «Supporting interaction with the internet of things across objects, time and space,» de *Internet of Things (IOT)*, IEEE, 2010, pp. 1-8.

[59] P. E. Estrada–Martinez y J. A. Garcia–Macias, «Semantic interactions in the Internet of Things,» *International Journal of Ad Hoc and Ubiquitous Computing,* vol. 13, nº 3, pp. 167-175, 2013.

[60] R. Minwoo, K. Jaeho y Y. Jaeseok, «Integrated Semantics Service Platform for the Internet of Things: A Case Study of a Smart Office. Sensors,» *Sensors,* vol. 15, nº 1, pp. 2137-2160, 2015.

[61] Piyare, S. Rajeev y R. Lee, Towards internet of things (iots): Integration of wireless sensor network to cloud services for data collection and sharing., arXiv preprint arXiv, 2013.

[62] N. Rosen, R. Sattar, R. W. Lindeman, R. Simha y B. Narahari, «HomeOS : Context-Aware Home Connectivity,» de *International Conference on Wireless Networks*, 2004.

[63] K. Konstantinos y K. Artem, «Semantic interoperability on the web of things: The semantic smart gateway framework,» de *Sixth International Conference on Complex, Intelligent, and Software Intensive Systems*, Palermo, 2012.

[64] E. K. Charbel y P. André, «A Model Driven Approach Accelerating Ontology-based IoTApplications Development,» de *13th SEMANTiCS conference*, Amsterdam, 2017.

[65] B. Umesh, P. Pankesh, C. Saurabh y Q. Yongrui, «A Semantic-Enabled Framework for Future Internet of Things Applications,» de *IEEE World Congress on Services (SERVICES)*, Honolulu, 2017.

[66] R. Hafizur y H. Iftekhar, «Fog-based semantic model for supporting interoperability in IoT,» *IET Communications,* vol. 13, nº 11, pp. 1651-1661, 2019.

[67] T. Xu, C. Davide, E. Ferrera, L. Shuai, J. Götz, L. Maillet-Contoz, M. Emmanuel, M. Diaz-Nava, B. Abdelhakim y C. Salim, «Model Based Methodology and Framework for Design and Management of Next-Gen IoT Systems,» *SAM IoT,* 2020.

[68] SMARTEC, «SMARTEC: Domótica y Seguridad,» Estudio Flama, 2017. [En línea]. Available: http://www.smartec.com.ar/. [Último acceso: 8 Agosto 2017].

[69] Ozom, «Ozom,» Ozom, [En línea]. Available: http://help.ozom.me/help_center. [Último acceso: 24 1 2016].

[70] Control4, «Control4,» Control4, [En línea]. Available: http://www.control4.com/. [Último acceso: 24 1 2016].

[71] M.-A. Niño-Zambrano, «Interacción Semántica de Objetos en La Web de las Cosas,» Universidad del Cauca, Poayán, 2016.

[72] M.-A. Ardila-Nuñez, «Carlos A. Cobos L., M.A.N.Z., Metamodelo de evaluación para la educación en linea,» Universidad del Cauca, Popayán, 2003.

[73] F. Bonomi, R. Milito, J. Zhu y S. Addepalli, «Fog computing and its role in the internet of things,» de *presented at the Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, Helsinki, Finland, 2012.

[74] A. G. M. Hauswirth, S. Krco, N. Stojanovic, M. Bauer y R. Nielsen, «An architectural blueprint for a Real-World Internet,» *Springer Berlin Heidelberg,* pp. 67-80, 2011.

[75] D. Bonino y F. Corno, «Dogont-ontology modeling for intelligent domotic environments,» *Springer Berlin Heidelberg,* pp. 790-803, 2008.

[76] P. Eric y S. Andy, «SPARQL Query Language for RDF,» W3C Recommendation, 2008. [En línea]. Available: https://www.w3.org/TR/rdf-sparql-query. [Último acceso: 27 Enero 2021].

[77] SQLite Consortium, «SQLite Home Page,» [En línea]. Available: https://www.sqlite.org/. [Último acceso: 27 Enero 2021].

[78] W. A. Scott, «The Agile Unified Process (AUP) Home Page,» [En línea]. Available: http://www.ambysoft.com/unifiedprocess/agileUP.html. [Último acceso: 27 Enero 2021].

[79] R. Hernández, C. Fernández y P. Baptista, Metodología de la Investigación, México: McGraw Hill Interamericana, 1998.