ANEXOS
INTERACCIÓN SEMÁNTICA DE OBJETOS EN LA WEB DE LAS COSAS

MIGUEL ÁNGEL NIÑO ZAMBRANO
Tesis para obtener el título de Doctor en Telemática

Director:
PhD. Ing. Gustavo Adolfo Ramírez González

Universidad del cauca

Instituto de Postgrados en Electrónica y Telecomunicaciones
Doctorado en Ingeniería Telemática
Departamento de Telemática
Línea de Investigación Aplicaciones y Servicios sobre Internet

Popayán, Diciembre de 2015

CONTENIDO

Página

LISTA DE TABLAS

Página

LISTA DE FIGURAS

Página

ANEXO 1. MARCO DE REFERENCIA

El propósito del presente capítulo es establecer con mayor profundidad los principales referentes teóricos que se necesitan para abordar el presente proyecto. La mecánica utilizada consiste en dividir el marco de referencia en secciones bien diferenciadas, pero relacionadas unas con otras en la medida del avance cronológico del conocimiento y en otros casos en las necesidades que surgían y las soluciones propuestas en la evolución de la Web Semántica y que desean ser aplicadas al Internet de Objetos – (IoT).

Al final de cada apartado hay una reflexión de que elementos son necesarios estudiar e incorporar al proyecto para el logro de sus objetivos.

1.1 Internet de "Objetos" – "Cosas" (Internet of Things)

Los avances en las tecnologías de la información y la comunicación – TIC, han permitido, dejar de lado las barreras geográficas y temporales para transmitir información, creando consigo una nueva revolución de pensamiento y maneras de hacer las cosas. La Internet ha evolucionado a tal punto que en nuestro estilo de vida actual se ha convertido en una necesidad básica para poder interactuar. Los avances en la telefonía móvil y los servicios Web en los últimos años han marcado la vida de las personas cambiando la manera de comunicarnos, distraernos, trabajar y hacer vida social entre otros aspectos relevantes de las personas.

Normalmente las personas vivimos en una realidad mediada por nuestra relación con los objetos físicos – conceptuales y como a través de ellos intervenimos en dicha realidad. Parece lógico pensar que la siguiente evolución natural de la Internet es que estos objetos se conecten a la misma para que se comuniquen con nosotros y entre ellos mismos, con el fin de crear mejores servicios y bienestar; es precisamente lo que se busca en el Internet de Objetos.

Así, formalmente podemos decir que el IoT busca lograr conectividad entre cualquier objeto o cosas y además entre las personas y los objetos, todo el tiempo y en cualquier momento y lugar. Estas capacidades requieren dispositivos que puedan interactuar con el medio que los rodea como los sensores, e introducir capacidades de inteligencia y conectividad[13].

El IoT se le ha denominado la "*Tercera Ola de la Información"* [14] en el mundo de la industria después de la computadora y el Internet. El "*Internet de las Cosas*" se refiere a la información proveniente de dispositivos de detección tales como redes de sensores inalámbricos (Wireless Sensor Network – WSN), tecnologías de Identificación por radiofrecuencia (R*adio Frecuency Identification – RFID*), código de barras bidimensionales, sistemas de posicionamiento global y redes de corto alcance ad hoc (ver Figura 1), que se basan en el concepto de comunicación de

maquina a máquina (*machine to machine - M2M*) combinando una variedad de redes de acceso y el internet para conformar una gran red inteligente.

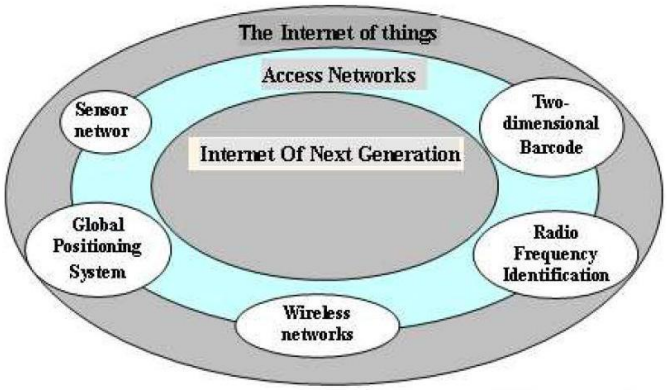


Figura 1. Modelo de Conceptos de la Internet de Objetos
Tomado de: L. Yuxi, "Key Technologies and Applications of Internet of Things"[14]

1.1.1 Convergencia de Visiones en la IoT

El IoT se puede ver como una intersección de tres visiones diferentes[15]: Visión orienta a las cosas, visión orientada a la semántica y la visión orientada a la Internet (ver Figura 2).

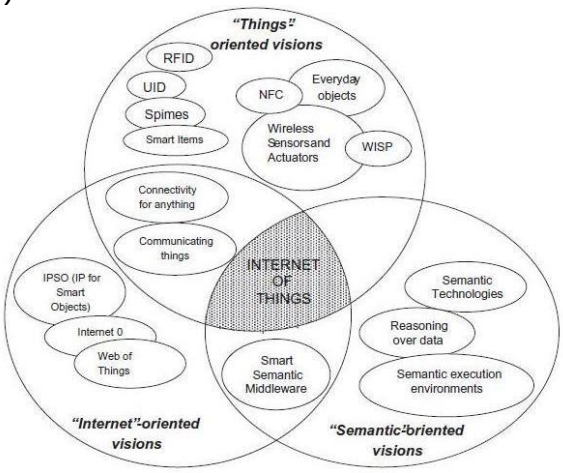


Figura 2. Internet de Objetos convergencia de visiones.
Tomado de: L. Atzori, "The Internet of Things: A survey"[12]

- **Thing Oriented Visions / Visión orienta a las cosas**: La visión orientada a las cosas corresponde a las tecnologías desarrolladas para identificar (*etiquetar*) y comunicarse con los objetos como: RFID, sensores, actuadores y teléfonos móviles entre otros, los cuales pueden operar e interactuar entre si y buscar objetivos comunes.
- **Internet Oriented Visions / Visión Orientada a la Internet**: La visión orientada a la Web se basa en protocolos de internet (*Internet Protocol* –

IP) para el desarrollo de servicios de comunicación con los objetos, éstos pueden ser introducidos en elementos del mundo real como vallas publicitarias, automóviles e interruptores de luz entre otras.

- **Semantic Oriented Visions / Visión Orientada a la Semántica**: Se basa en la separación entre datos y el significado de los mismos, almacenando por separado los significados semánticos de los datos y la capacidad de razonar sobre ellos mismos. Estandarizar las descripciones de recursos heterogéneos es clave para interoperabilidad del IoT.

La investigación aporta a la visión orientada a la semántica, por cuanto se pretende incorporar técnicas de la recuperación de la Información – (RI) en la IoT, con un enfoque particular que permita desarrollar mejor ésta área.

1.2 Características, Clasificación y Funciones de Objetos del IoT

1.2.1 Características de los objetos

Los objetos del IoT se les puede atribuir el desempeño de una función o rol[14], dependiendo de sus capacidades informáticas tales como: conectividad de red, energía disponible, escenarios de tiempo y espacio entre otras. En función de características individuales, efectos producidos y relaciones que se plantean. En la Tabla 1, se pueden apreciar con más detalle dichas características.

Características	Descripción
Básicas	Un objeto puede ser un objeto del mundo real o del mundo virtual puede haber comunicación entre el mundo físico real y el mundo virtual generando una información de su estado.
Generales (Adicionales a las características funcionales básicas)	Un objeto puede utilizar un *Servicio* como interfaz para comunicarse con otro objeto, un 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 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 sistemas físicos.

Tabla 1. Características de los Objetos según su función o rol de la IoT.

Estas características se pueden tener en cuenta al momento de clasificar los sensores de acuerdo a las capacidades que van adquiriendo. El presente estudio pretende aportar a los tres primeros comportamientos: básico y general.

1.2.2 Propiedades de la Red de Objetos

El IoT extiende el mundo físico hacia ubicuidad de tiempo – espacio (ver **¡Error! No se encuentra el origen de la referencia.**) e incluye propiedades de: contenidos, recopilación, comunicación, conectividad, combinación y cálculos, con el fin de permitir la construcción de entornos complejos de objetos y personas los cuales se pueden comunicar e interactuar entre sí para colaborarse.

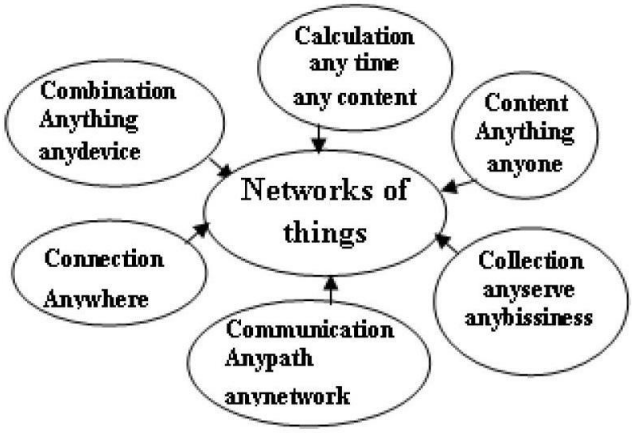


Figura 3. Internet de Objetos propiedades básicas
Tomado de: L. Yuxi, "Key Technologies and Applications of Internet of Things"[14].

Las propiedades descritas en este apartado se pueden tener en cuenta al momento de crear el modelo conceptual que recopila el conocimiento de cada objeto del IoT.

1.2.3 Taxonomía de Objetos del IoT

Los objetos tienen unas características especiales como: diferentes tipos de datos [16] o señales, capacidad de almacenamiento, procesamiento, comunicación, autonomía energética, localización, origen, estado, utilización y una identificación única [17]. Para esto es necesario iniciar con una taxonomía de las características y propiedades de los objetos actuales, con el fin de servir como base para su conceptualización. Así el trabajo de Mathew, et al. [18], permite caracterizar los objetos basado en tres dimensiones: comunicación, procesamiento y almacenamiento, además de una identidad, la cual permite establecer una jerarquización de los mismos. Las características de las dimensiones las hemos resumido en la Tabla 2.

Capacidades Intrínsecas	Descripción	Ejemplos
Identidad	Identificación de los objetos.	Código de barras, códigos RFID.
Procesamiento	Control y administración de los	Uso de chip, uso de sistemas operativos embebidos.

Capacidades Intrínsecas	Descripción	Ejemplos
	objetos.	
Comunicación	Interfaces de entrada y salida, medios de comunicación y protocolos.	Puertos USB, Bluetooth, API's de programación web.
Almacenamiento	Capacidad de almacenamiento para los estados de los objetos y los valores.	Almacenamiento del identificador del objeto por ejemplo un código RFID, almacenamiento para interfaces de comunicación de puertos y direccionamiento, memoria RAM y discos duros HDD.

Tabla 2. Dimensiones de clasificación de los objetos del IoT.

De acuerdo a lo anterior, Mathew propone una clasificación de objetos de la IoT de acuerdo a sus capacidades intrínsecas. En la Tabla 3, se presenta esto:

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.

*I(Identificación), P(Procesamiento), C(Comunicación) y A(Almacenamiento)

Tabla 3. Clasificación de los objetos del IoT.

Aparte de la taxonomía propuesta por Mathew, et al. [18], este hace referencia a otras propuestas que ofrecen clasificaciones de los objetos en aspectos como: la Longevidad (Lifespan), Propietario (Ownership), Compartabilidad (Shareability), Amigos (Friends), Buscabilidad y Accesibilidad (Searhability and accessibility), que están relacionadas más con el contexto. Sin embargo, [19] hacen un estudio extenso sobre los datos importantes relacionados a la conciencia y contexto que se puede manejar en la WoT, presentan dos categorías: la *primaria* son datos de los sensores que no necesitan ser procesados (Ej. Datos GPS), la *secundaria* son datos a los cuales se les aplica procesos de fusión y recuperación de la información (Ej. Identificación de distancia entre sensores), adicionalmente hace un producto cartesiano con cuatro perspectivas de contexto: actividad, tiempo, identidad y localización. Estos elementos se convierten para el proyecto en una fuente importante de información semántica a ser indexada.

Mathew propone una ontología con las clasificaciones expuestas. Esta ontología ha sido tomada como elemento de partida en el presente proyecto, con el fin de crear la base de una ontología de objetos, la cual es enriquecida con otros elementos importantes para realizar la indexación semántica.

A partir de lo anterior, podemos definir los objetos inteligentes (*Smart Things*)[20], como objetos de la IoT que se puede identificar claramente y exhiben capacidades de procesamiento, comunicación y almacenamiento, además cuenta con una representación digital, la cual permite interactuar con las aplicaciones y los usuarios en la Web. Estos objetos, 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 (*Gateway residenciales, routers o PC*) y les dote de las capacidades necesarias para su correcto aprovechamiento, a esto se le ha denominado Smart Gateway[17, 21-23]. En el caso de las redes de sensores (Wireless Sensor Network – WSN) se buscan topologías en los cuales los nodos coordinadores tengan las capacidades de los objetos inteligentes o se comuniquen con los Gateway para su interacción con la Web.

Teniendo en cuenta las características de los objetos, las propiedades del red IoT y la intercepción de las tres visiones que intervienen en el IoT, podemos identificar los temas principales que se deben abordar al momento de crear una red de objetos en el IoT, además de permitir reconocer los elementos básicos que se pueden utilizar en el IoT. Se puede resumir éstos elementos en cinco preguntas:

1. ¿Qué tipo de Objetos se desea crear de acuerdo a sus características?
2. ¿Qué propiedades de la red de Objetos se van a implementar?
3. ¿Qué tecnologías para representar e identificar los objetos se debe utilizar?
4. ¿Qué servicios se desea desarrollar en la Web de Objetos?
5. ¿Qué capacidades semánticas tiene los objetos?

1.3 Arquitectura de la IoT

Se han realizado varias propuestas alrededor de la arquitectura [24] de la IoT. Sin embargo existe una coincidencia en utilizar una estrategia de capas bien definidas, en la cual las capas inferiores adquieren los datos y las capas superiores utilizan los datos en aplicaciones y la capa del medio se convierte en un mediador común de comunicación (ver **¡Error! No se encuentra el origen de la referencia.**).

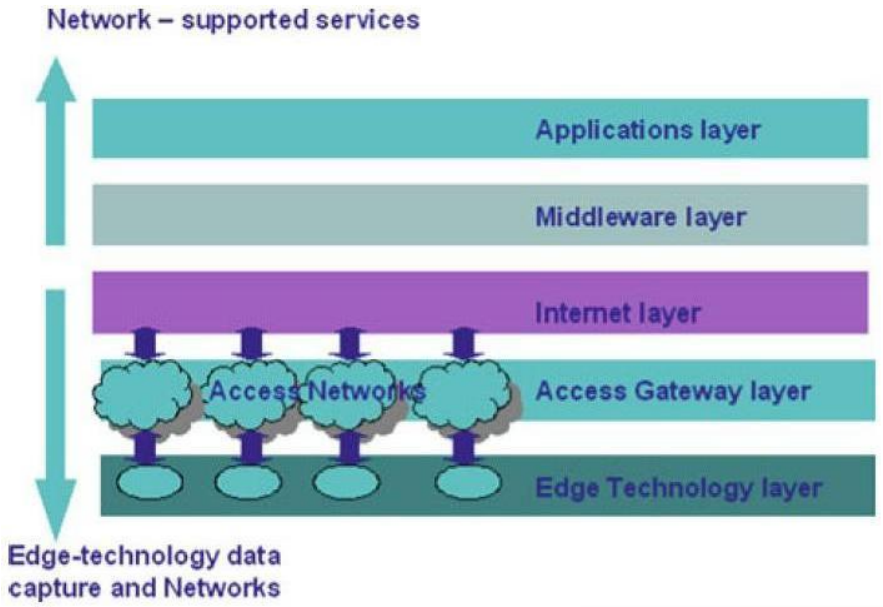


Figura 4. Internet de Objetos arquitectura en capas.
Tomado de: D. Bandyopadhyay, " Internet of Things: Applications and Challenges in Technology and Standardization"[24].

La descripción general de cada una de las capas es como sigue:

- **Edge Technology Layer / Tecnologia capa de borde**: En la primera capa de frontera están los objetos físicos dotados de las tecnologías de identificación como etiquetas RFID, sensores para recoger información del entorno y lectores para almacenamiento de datos, además de sistemas embebidos para procesamiento y comunicación.
- **Access Gateway Layer / Acceso a capa de entrada**: La capa de acceso o puerta de enlace se realiza el enrutamiento de mensajes, publicación y suscripción. Esta capa permite tener acceso a la información recopilada por los objetos específicos hacia la red (capa Internet), ocultando los detalles técnicos o físicos de transporte y conversión de datos.
- **Internet Layer / Capa de Internet**: En la (capa Internet) los datos de los objetos se transportan a través de protocolos IP, almacenándose en servidores de información físicos y accesibles por un software intermediario (Middleware), que pondrá a disposición de las aplicaciones los mismos.
- **Middleware Layer / Capa de middleware**: La capa intermedia es una interfaz entre la capa de componentes físicos y la capa de aplicación, se lleva a cabo la gestión de dispositivos, datos, filtrado, agregación y análisis semántico. Un ejemplo es el descubrimiento de información a través de formas de servicios como de código de producto electrónico (*Electronic Product Código – EPC*) y sus servicios web de información de nombres objetos (*Object Naming Service – ONS*).
- **Aplication Layer / Capa de aplicación**: Finalmente en la capa de aplicación se encuentran aplicaciones en producción, logística, medio

ambiente, seguridad pública y salud entre otras, además, con el creciente desarrollo de la tecnologías RFID se están integrando nuevas aplicaciones.

La capa de Middleware en las últimas propuestas la dividen en dos: la primera centrada en los Middleware y Servidores IoT y la segunda en una *capa de representación semántica*, la cual desarrolla las técnicas de anotación y gestión de la información de los objetos.

La capa de aplicación a su vez ha sido desplazada primero por una capa de servicios en la cual se plantea el uso de servicios Web semánticos [11] como herramienta para implementar la interoperabilidad en la capa de servicios los cuales pueden servir para crear ciudades inteligentes [8] y luego si la capa de aplicaciones, en esta capa se trabajan conceptos como: physical mashups [25], bases de conocimiento ubicuo y compresión de datos [9].

Los principales enfoques arquitectónicos propuestos son: SOA (Service-Oriented Architecture) [7], RESTful (Sistemas que usan Representational State Transfer) [26]. Sin embargo, las últimas propuestas han incorporado protocolos de telemetría para la comunicación entre los mismos objetos como MQTT.

Scioscia and Ruta [9] hacen los primeros aportes hacia la incorporación de semántica en los objetos de la IoT, propone como alternativa de solución un enfoque de compresión de datos y su consulta eficiente, además de consulta a ontologías y servicios web semánticos. La consulta eficiente se tiene en cuenta el presente proyecto con el fin de bajar los niveles de almacenamiento y procesamiento que son críticos en la implementación de software para la IoT. Los servidores seleccionados aportan soporte para la compresión y la seguridad de las comunicaciones.

Finalmente, el trabajo de Vega-Barbas, et al. [23] plantean una arquitectura muy completa, orientada a servicios dinámicos para la interoperabilidad semántica en el IoT. Establece cinco módulos bien definidos que permiten capacidades de abstracción de los objetos y las aplicaciones a diferentes niveles. Un elemento importante es que enfoca los esfuerzos a desarrollar aplicaciones hacia los objetos locales a través de los gateways residenciales, como elemento de abstracción física que puede permitir conectar objetos con pocas capacidades de procesamiento y almacenamiento. Aunque hasta ahora no se han presentado implementaciones reales de dicha propuesta, se puede tener en cuenta su enfoque para ventilar la posibilidad de tener indices distribuidos en dichos dispositivos.

Las capas y los diferentes estudios de arquitecturas serán tenidos en cuenta para hacer una propuesta arquitectónica del modelo conceptual propuesto. En especial la capa de Middleware se realizará una aportación en el análisis semántico de la información provista por los sensores y sus características.

1.4 Tecnologías del IoT

El concepto de Planeta Inteligente (Smart Planet)[13] es propuesto en el año 2008 por IBM el cual se centra en el concepto de sabiduría y su implantación en el IoT. Las tecnologías clave son el RFID (Radio Frequency Identification), el (Wireless Sensor Network – WSD), los (Global Positioning Systems – *GPS*), redes móviles, servicios web (Web Service – *SW*), aplicaciones industriales y el desarrollo de chips son componentes del concepto Smart Planet. Según Miao and Bu [27] las tecnologías de la IoT y la relación de las distintas clases de redes se resumen en la **¡Error! No se encuentra el origen de la referencia.**.

Tecnologías	Especificaciones
RFID technology / Tecnología RFID	La inducción y la propagación electromagnética evita el contacto físico, se ha desarrollado seguridad anti replica, cifrado y certificados.
Sensor Network / Red de sensores	Plataforma para realizar mediciones del mundo con un gran número e nodos que interactúan entre sí, posee aplicaciones en la defensa, industria, agricultura y funcionamiento de las ciudades.
Smart technology / Tecnología inteligente	Los objetos reciben inteligencia para comunicarse activa y pasivamente entra en el campo de la (Inteligencia Artificial – IA), interacción (Maquina humano – M2H) y tecnologías de control inteligente y procesamiento inteligente de señales.
Nanotechnology	Estudio de partículas diminutas y objetos que interactúan y se conectan entre sí, se han aplicado en campos como la medicina, energía y transporte.

Tabla 4. Tecnologías del IoT.

Miao and Bu [27] establecen las relaciones de las diferentes tecnologías y redes de objetos que intervienen en la IoT, tal como se puede apreciar en la Figura 5.

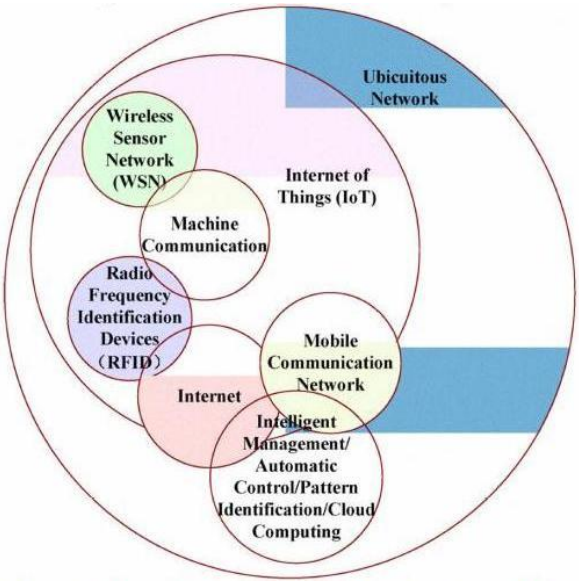


Figura 5. Relación entre la IoT y las redes existentes.
Recuperado de: Y. Miao, "Research on the Architecture and Key Technology of Internet of Things (IoT) Applied on Smart Grid"[27].

En la Figura 5, se puede observar como la red ubicua está compuesta por varias redes agrupadas sobre la “Internet of Things” y unas capacidades de inteligencia, gestión y control automático en la nube. A su vez la IoT cubre las redes sensores inalámbricos, Internet, telecomunicaciones y redes RFID.
El trabajo aplica sus resultados específicamente en aportar a las capacidades de inteligencia y gestión de los objetos de las diferentes redes para la construcción de la computación ubicua. Sin embargo, se aporta en los procesos iniciales de enriquecimiento semántico y creación de servicios de información, para que a partir de los mismos se generen los comportamientos inteligentes.

1.5 Infraestructura de la IoT

Los desafíos [28] del IoT radican en que los recursos de sensores poseen algo llamado heterogeneidad profunda, es decir diferentes hardware y software que hace difícil su integración transparente, por otro lado se consumen en tiempo real, están en redes distribuidas y dinámicas, además presentan necesidades de limpieza de datos debido a condiciones ambientales unido a las limitaciones de los objetos de la baja capacidad de proceso y consumo de energía, la cantidad de objetos y datos.

Para afrontar los desafíos que propone la IoT se ha creado un conjunto de infraestructura de hardware y software, que permite resolver en parte los problemas encontrados, así para la heterogeneidad profunda se han creado los Middleware de Objetos y los Servidores de IoT, los cuales tienen enfoques diferentes de solución, pero efectivos al momento de integrar los objetos de la IoT.

Para establecer las características de los objetos, es necesario definir un mecanismo que permita etiquetarlos [29], para posteriormente ser consultados y actualizados, esto se puede hacer a través de metadatos. Las fuentes de metadatos para los objetos de la IoT se pueden obtener de los trabajos presentados que identificaron metadatos importantes de los sensores, los estándares propuestos, los servicios de hosting de objetos (Servidores mediadores de IoT) y las ontologías de sensores.

Posteriormente, hay que decidir en dónde almacenar los metadatos y como accederlos de manera eficiente, ya que las características de los objetos imponen restricciones en este sentido. Uno de los mecanismos de almacenamiento tanto de la información de metadatos como de datos que proveen los objetos, es la creación de *feeds* (Alimentadores) en Servidores mediadores IoT, los cuales funcionan como Middlewares que permiten conectar y abstraer los objetos físicos del IoT, además presentan varios servicios a sus usuarios, entre ellos, el de gestión de datos y metadatos en diferentes formatos como Json, Xml y Csv.

Una vez resuelto el problema de acceso, se pueden desarrollar aplicaciones como: minería de datos y búsqueda en indexación para aprovechar su información adecuadamente.

En los apartados siguientes se hace énfasis en los Middleware y Servidores de Objetos, los cuales fueron analizados con el fin de incorporar esta infraestructura en la solución propuesta.

1.5.1 Middlewares del IoT

Como se presentó en un capitulo anterior el middleware[30] es una pieza de software importante en la arquitectura del IoT por ello se hace especial énfasis en las infraestructura existente con el fin de establecer su estado de avance y las posibilidades de uso para el presente proyecto.

El papel de integración entre la capa tecnológica y la capa de aplicación es necesario en un entorno heterogéneo como el que presenta el IoT con la diversidad de sensores y tecnologías para hacer una aplicación sería muy complejo, los middleware absorben esta complejidad y presentan a los programadores y usuarios interfaces de programas de aplicación (Application Program Interfaces - API’s) e interfaces simples con servicios de alto nivel.

1.5.1.1 Componentes Funcionales De Los Middleware

Según Bandyopadhyay, et al. [31]. Para lograr la interoperabilidad en el IoT los middleware aportan abstracciones para ofrecer servicios múltiples, las principales funciones de los middleware se pueden clasificar en taxonomías de enfoques de adaptabilidad (adaptability), enfoques de conciencia del contexto (context-aware) y enfoques de descubrimiento de servicios (discovery service), además funciones en dominios específicos tales como WSN y RFID con bases de datos espaciales, tuplas (*tupla space*) y basado en eventos.

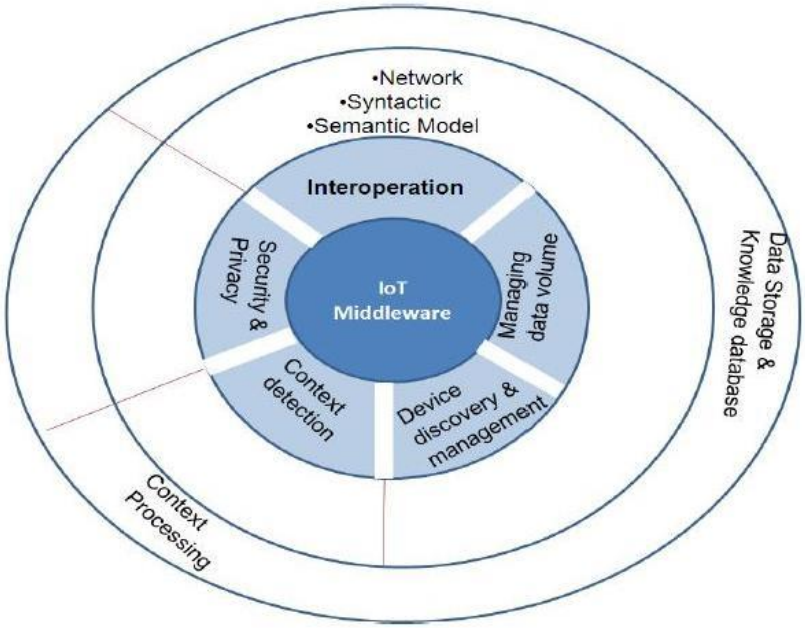


Figura 6. Componentes Funcionales de Los Middleware
Recuperado de "Role of middleware for internet of things: A study " Bandyopadhyay, et al. [31]

En la Figura 6, se puede apreciar de adentro hacia afuera que en el primer círculo se encuentra los bloques esenciales requeridos como: interoperación, gestión del volumen de datos, descubrimiento y gestión de dispositivos, detección de contexto y seguridad y privacidad. En el segundo círculo la división presenta porciones más grandes, abarcando varios bloques funcionales básicos del primer círculo, pero orientado a los modelos sintácticos, semánticos y de red. Finalmente el círculo externo no forma parte escencial de todos los middleware y se encuentran procesos complejos como: almacenamiento de datos (*Data Storage*) y base de datos de conocimiento (*Knowledge Database*) y procesamiento del contexto (*Context Processing*). La descripción más detallada se hace a continuación:

- **Interoperation / Interoperación**: Usa los mismos métodos de información sobre diversas interfaces de comunicación, los protocolos soportan intercambios de datos en redes diferentes sin tener en cuenta el contenido de la información. Existe interoperabilidad en la red sintáctica (network

sintáctica) y red semántica (network semantic), las interacciones sintácticas son a nivel de métodos y la semántica a nivel de descripciones o significados. La interoperabilidad semántica tiene en cuenta normas de comprensión, significados y contenidos, API's de programación de servicios SOA y Ubicuos, también lenguajes para modelado de sensores como SensorML para la conexión entre diversos sensores.

- **Contex Detection / Detección del Contexto**: El contexto identifica los factores de una situación. Una situación puede ser una persona, lugar y objetos, se caracteriza la situación y se genera una respuesta o decisión.
- **Context Processing / Procesamiento del Contexto**: Los identificadores de los servicios cambian dependiendo del contexto y se gestionan por medio de estructuras de datos con información de los identificadores y la extracción de la información se realiza con minería de datos.
- **Device Discovery and Management / Detección de dispositivos y gestión**: La interoperabilidad semántica y sintáctica se logra a través de la combinación de técnicas Ontológicas y Servicios Web Semánticos. Un Dispositivo Semántico es la representación física de un dispositivo almacenada en ontologías de dispositivos, el mapeo de físico a semántico incluye información del dispositivo como: nombre, proveedor, tipo, capacidades, funcionamiento y propiedades de seguridad, descripción de recursos software y hardware. En este módulo existen estrategias de descubrimiento y gestión las cuales soportan: identificación, descubrimiento, secuencias y propiedades de ciclo de vida (*life-cycle*). Utilizando Agentes es capaz de soportar roles y escenarios para descubrimiento de agentes y el uso de protocolos P2P, un ejemplo es (S-APL). Otros módulos que pertenecen a la misma categoría son el *Monitoring and Inventory* el cual soporta un repositorio de dispositivos (*Device Repository*) definiendo: las clases de dispositivos, Monitor de dispositivo (*Device Monitor*) el cual gestiona los procesos e invocación de métodos, el historial y descubrimiento (*History and Discovery*) para la correlación de eventos y lleva un historial de los eventos invocados y mensajes de depuración, el repositorio de dispositivo encuentra dispositivos y servicios a través de: un modo pasivo (escuchar y notificar) y un modo activo(búsqueda dinámica). En servicios ubicuos se tiene el descubrimiento de servicios (sw-Addressing, ws-discovery, ws-metadata exchange, ws-evening).
- **Security and Privacy / Seguridad y Privacidad:** La seguridad se maneja a alto nivel entre pares y nivel de gestión de topologías de red con autenticación, permisos de acceso y protección de información de enrutamiento, también se presenta seguridad en todos los bloques funcionales o confianza entre otros.
- **Managing Data Volume / Volúmenes de datos**: Los datos producidos en exabytes requiere de métodos de datos para buscar (Indexación), recuperar (modelo de procesos) y transferir (gestión de transacciones). Otros módulos

son el módulo de gestión del almacenamiento (*Storage Manager Module*) y el módulo de apuntalamiento (*Shoring Module*) el cual recopila, filtra y finalmente el módulo de consultas (*Query Manager*).

No todos los middleware implementan todos los módulos descritos anteriormente y depende de sus objetivos, funcionalidad, arquitectura base y enfoque, tal como se presenta en los próximos apartados con diferentes puntos de vista de taxonomías de middleware.

1.5.1.2 Taxonomía de Middleware Por Funcionalidad

El Reporte de Gartner [32, 33] la empresa consultora en TIC define un middleware como "*es el desarrollo de sistemas en tiempo de ejecución que permite interacciones a nivel de aplicación entre programas en un entorno de computación distribuida*", estas interacciones pueden ser síncronas o asíncronas, los middlewares logran que las aplicaciones se ejecuten en múltiples plataformas logrando interoperabilidad entre aplicaciones, escalabilidad, portabilidad, independencia de la ubicación y oculta la heterogeneidad de hardware en ambientes distribuidos. Según Zhou [34] la consultora Gartner divide los middleware en tres clasificándolos en funcionalidades.

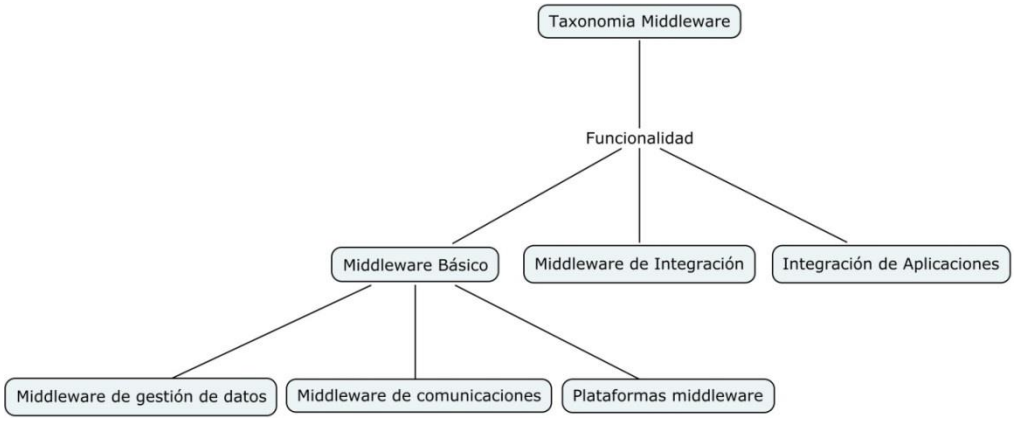


Figura 7. Taxonomía de Middleware Por Funcionalidad

Basado en "The Internet of Things in the Cloud: A Middleware Perspective," in *The Internet of Things in the Cloud,"* Zhou [34]

El Middleware Básico es la capa en la que se encuentra en la arquitectura general del IoT, este se puede subdividir en middleware de gestión de datos, comunicaciones y plataformas. En la Tabla 5 se presentan las principales características de este tipo de Middleware.

Capa	Clase	Descripción	Ejemplos
Middleware	Middleware de gestión de datos	Se gestiona la lectura y escritura de datos en bases de datos remotas abarca los	Google Files Systems, IBM GPFS, Archivos en Red. Open Database Connectivity,

		sistemas distribuidos, archivos paralelos, y acceso a bases de datos	Bibliotecas Java, IBM DB2, Oracle, Microsoft SQL Server.
	Middleware de comunicaciones	Servicios de comunicaciones por protocolo (Serial Peripherical Interface – SPI).	Propietarios como IBM WebSphere MQ/MQTT-O, Microsoft MSMQ. Estándar como ASN.1 con procedimientos remotos RPC CORBA/IIOP, Java Message, Service (JMS). Servicios web SOAP y REST.
	Plataformas middleware	Entornos de ejecución y alojamiento y servicios de gestión de recursos en tiempo de ejecución como almacenamiento en caché, multiplicación de programas, balanceo de carga tolerancia a fallos, monitoreo y procesamiento de transacciones distribuidas e interfaces para varias formas de comunicación (one-way messaging y request/reply).	Java EE o. NET Framework / COM. Procesamiento de transacciones TPM como IBM CICS, TPMS Unix distribuido. Servicios de portal como BEA Weblogic Portal, Plumtree, Vignette, y otros que están que complementan servidores web y servidores de aplicaciones.

Tabla 5. Características de Los Middleware de la IoT

Las clases Middleware de Integración no hacen parte de los middleware del IoT, tal como se presenta en la **¡Error! No se encuentra el origen de la referencia.**:

Capa	Descripción	Ejemplos
Middleware de Integra	Cuando se requiere conexiones entre aplicaciones de desarrollo independiente.	Integración (A2A) o empresarial (B2B).
Integración de Aplicaciones	No es un middleware puro sino una combinación de middleware y procesos de negocio específicos.	La sincronización de órdenes de compra entre un sistema de pedidos y un sistema de gestión de pedidos.

Tabla 6. Capa Middleware no pertenece a IoT

Estas características serán tomadas en cuenta al momento de seleccionar el middleware que se adapta mejor al modelo propuesto.

1.5.1.3 Taxonomía de Middleware por arquitectura

También se puede clasificar los middleware por la arquitectura que implementan, así en la **¡Error! No se encuentra el origen de la referencia.** se puede observar esta clasificación.

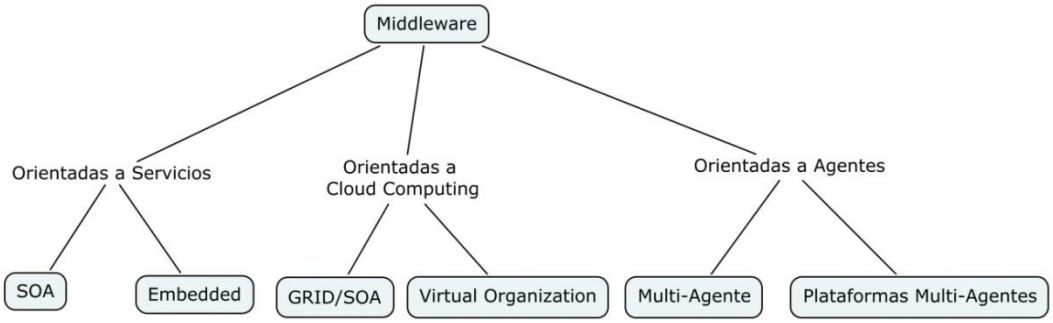


Figura 8. Taxonomía de Middleware Por Arquitectura

Basado en "The Internet of Things in the Cloud: A Middleware Perspective," in *The Internet of Things in the Cloud,"*Zhou [34]

A continuación se detalla las arquitecturas de middleware más importantes para la IoT y sus características generales.

La arquitectura orientada a servicios (*services oriented architecture* – SOA) es una de las arquitecturas que han adoptado algunos estudios como en Atzori, et al. [12]. Esta arquitectura descompone servicios complejos y monolíticos en aplicaciones simples y por componentes con interfaces comunes. El proceso de negocio requiere lenguajes de ejecución de procesos de negocio (*Business Process Execution Language* - BPEL) para crear flujos de trabajo y servicios coordinados que apoyan las interacciones entre los componentes agilizando la adaptación de las aplicaciones a los cambios del mercado, la interacción con las entidades externas requiere de lenguajes como lenguajes de descripción de servicios web (*Web Services Description Language* - WSDL) para llamar flujos de trabajo desde el interior de otros flujos de trabajo es decir anidados, todo esto facilita el desarrollo y la comunicación con otras aplicaciones.

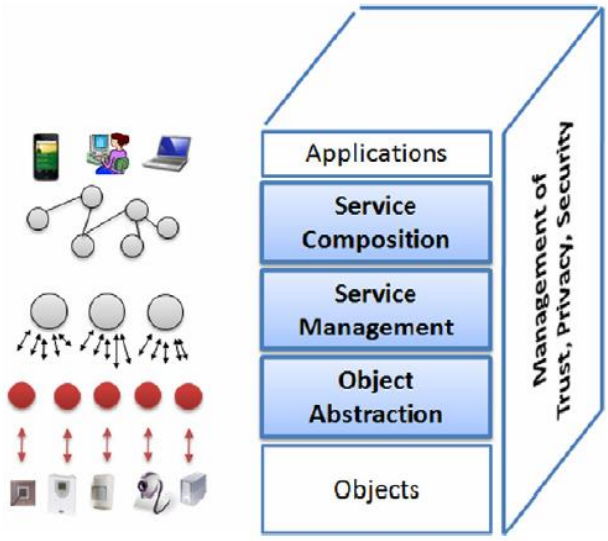


Figura 9. Arquitectuira de Middleware SOA

Recuperado de "The Internet of Things: A survey" Atzori, et al. [12]

Las características de cada capa son:

- **Aplications / Aplicaciones**: No hace parte del middleware sino que aprovecha de todas las funcionalidades prestadas por el middleware, estas funciones se utilizan por medio de protocolos estándar de servicio y tecnologías de composición de servicios.
- **Service Composition / Composición de servicios**: Crea un entorno de servicio con un repositorio de instancias de servicio de los objetos conectados, construyen servicios complejos por composición de servicios, los cuales soportan procesos complejos en secuencias de acciones que son coordinadas por componentes software individuales.
- **Service Management / Gestión de servicios**: Ofrecen funciones a la capa superior, funciones para los objetos como descubrimiento, monitoreo, configuración, gestión de calidad (QoS) de tolerancia a fallos, gestión de bloqueo y funciones semánticas como administración del contexto.
- **Object Abstraction / Objeto Abstracto**: Los objetos heterogéneos implementa funciones en su propio lenguaje por tal motivo es necesaria una abstracción por medio de una capa de envoltura que implemente una capa de interfaces y otra de comunicación, la capa de interfaz web implementa (operaciones de mensajería - outcoming) y la capa de comunicación es la lógica o los comandos para comunicarse con los objetos por ejemplo las pilas TCP/IP como TinyTCP, MIP, IwIP y los servidores web embebidos que son una abstracción de los objetos.
- **Management of Trust, Privacy, Security / Gestión de confianza, privacidad, seguridad**: La comunicación automática implica un peligro inherente de seguridad la cual requiere de mecanismos de vigilancia por medio de middleware que implementen servicios seguros para el IoT. El papel de la seguridad es primordial para el desarrollo comercial de IoT un ejemplo el (Fosstrak) que implementa tecnologías RFID que establece aplicaciones destino, un conjunto especifico de objetos y pone limitaciones geográficas, otro ejemplo pero que no tiene arquitectura SOA como (e-SENSE) que implementa tecnología de WSN y establece protocolos de control de entidades, un último ejemplo es el Proyecto (UbicSec & Sens) también WSN que implementa mecanismos de seguridad como largo pazo de registro de datos con (Tiny PEDS), abstracción de memoria compartida con (Tiny DSM) y almacenamiento de memoria distribuida con el protocolo (DISC) para WSN.

Los middleware embebidos (Embedded) son capas de software entre los sistemas operativos y las aplicaciones, tales como CORBA, EJB y DCOM. El objetivo es crear una abstracción del sistema operativo a las aplicaciones a través de servicios bien definidos. Adicionalmente hacen un desarrollo especial en mecanismos de comunicación. También se pueden encontrar subtipos de middleware embebidos como: orientados a mensajes, orientados a objetos,

basados en llamadas a procedimientos remotos, de base de datos y de transacciones.

Con respecto a la arquitectura Grid/SOA el Cloud Computing [35] es la fusión de las arquitecturas SOA (SOAP, REST) y Grid Computing, el nombre Grid es una analogía de las redes eléctricas debido a consistencia (consistent), Omnipresente (permasive), Confiable (dependable) y acceso transparente. El Grid Computing y la tecnología de Grupo de estaciones de trabajo son los antepasados de Cloud Computing, las granjas de servidores de Google se basa también en esta filosofía. La IoT se parece a Cloud Computing en que están constituidas por sistemas distribuidos de redes de comunicaciones los servicios de Cloud Computing, esta no requiere que los usuarios conozcan la ubicación física de los datos y configuraciones del sistema distribuido.

El Cloud Computing encierra los siguientes conceptos de servicios: Software como servicio (*Software as Service* - SaaS), Plataforma como servicio (Platform as Service - Paas) e Infraestructura como servicio (Infrastructure as Service - IaaS).

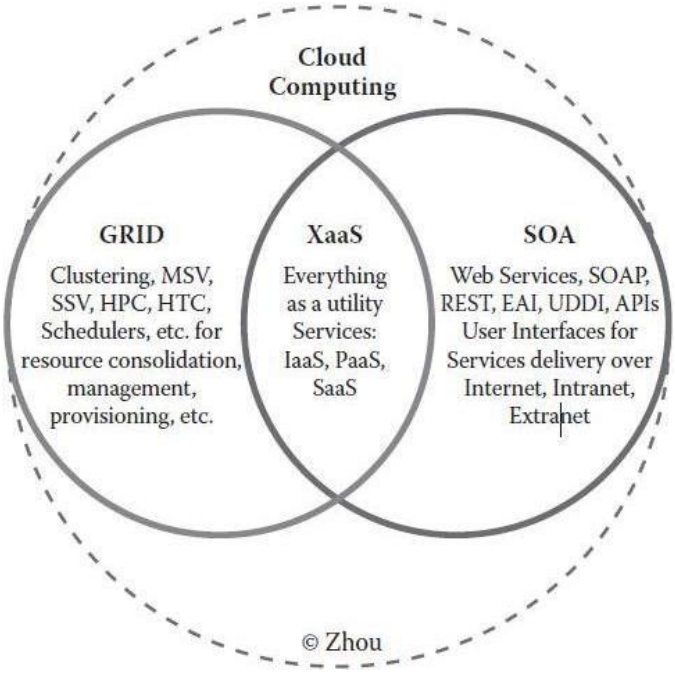


Figura 10. Arquitectura de Middleware Cloud Computing

Recuperado de "The Internet of Things in the Cloud: A Middleware Perspective," in *The Internet of Things in the Cloud,"* Zhou [34]

El *Cloud Computing* permitirá crear una gran red de objetos a nivel mundial para crear una arquitectura de Nube de las Cosas (*Cloud of Things)* que está en proceso de definición y especificación, esta especificación debe tener en cuenta los siguientes aspectos según Zhou [36] (ver Tabla 7).

Aspectos	Descripción

Conectivity / Conectividad	Conexión de objetos móviles o restringidos.
Content / Contenido	Datos masivos producidos por los objetos.
Cloud / Nube	Servicios de la nube.
Context / Contexto	Sensible al contexto para mejorar el rendimiento.
Collaboration / Colaboración	Comunicaciones cooperativas e intercambio de servicios.
Cognition / Cognición	Minería de conocimiento y mejora de sistemas autónomos.

Tabla 7. Aspectos a Tener en Cuenta Para Crear una Cloud of Things

Con respecto a las organizaciones Virtuales (*Virtual Organization* – VO) Hock Beng, et al. [37] definen que los portales de despliegue de sensores y la infraestructura de alojamiento grid son considerados como (Organizaciones virtuales - VO) y se caracterizan por tener diferentes proveedores, diferentes plataformas, conectividad de red variada y nodos de sensores programables, como ejemplos tenemos: SensorScope, Crossbow Motes, el proyecto estudio nacional de metodología (*National Weather Study Projec* - NWSP) con estaciones meteorológicas en Singapur. En la Figura 10, se presentan los tipos de middleware de este tipo de arquitectura.

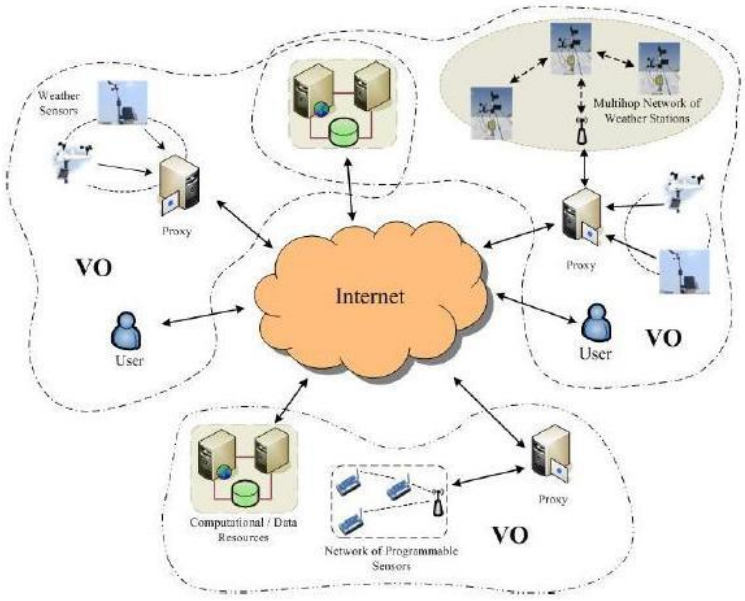


Figura 11. Tipos de Middleware en Organizaciones Virtuales
Recuperado de "A Large-Scale Service-Oriented Sensor Grid Infrastructure" Hock Beng, et al. [37]

También se puede observar en la Figura 10, como se pueden interconectar las VO para integrar servicios y compartir recursos entre diferentes VO, proporcionando interfaces de acceso web y sockets, servicios de composición de sensor y aplicación, también autenticación de usuarios. Los middleware implementan arquitecturas SOA y ontologías para sensores. Los componentes de diseño del modelo VO se detallan en la Tabla 8:

Capas	Descripción
Capa Plataforma	Programación de nodos existen productos comerciales como Crossbow Motes y Gumstix.
Capa Software	Gestiona las capacidades de detección, almacenamiento,

	encaminamiento utiliza fireware como el entorno de programación middleware Tiny OS.
Capa Semántica	Compuesto por ontología de datos: Proporcionan semántica a los datos. Ontología de procesos: Proporciona interfaces orientados a servicios de los componentes de la capa de software.

Tabla 8. Componentes de Diseño en Middleware con Arquitectura VO

Las arquitecturas de middleware orientados a multi-Agente se pueden entender como una evolución natural de los middleware en la IoT, ya que la misma es una red de objetos a los cuales se le han proporcionado capacidades de detección, procesamiento, almacenamiento y comunicación y a estos objetos se les denomina Objetos Inteligentes (Smart Thing – ST o Smart Object - SO), los cuales tienen el propósito de ser autónomos, así los middleware desarrollados con la arquitectura de agentes se ajusta perfectamente al concepto de SO, en la **¡Error! No se encuentra el origen de la referencia.** se muestran los aspectos en que se ajustan las agentes y los SO según Fortino, et al. [38].

Aspectos	Descripciones
Autonomy / Autonomía	Solución de problemas sin intervención humana o de otros agentes, controla las acciones y estados internos.
Social ability / Habilidades sociales	Interactúa cuando se requiera con humanos u otros agentes para ayudar a resolver problemas.
Responsiveness / Capacidad de respuesta	Al percibir el mundo físico, usuarios, otros agentes y responda de manera oportuna a los cambios.
Proactiveness / Productividad	Es oportunista es decir no se limita a responder al entorno sino que tomo la iniciativa por ejemplo dirigir una conducta a un objeto.

Tabla 9. Aspectos de los Objetos Inteligentes Correspondientes con los Agentes Inteligentes

La alianza IPSO [39] definió el middleware para dominios determinados de aplicación tales como *Smart SO* para conciencia del contexto, *2WEAR* portátiles SO, middleware para arquitecturas preinstaladas de servicios como Aura, Gaia e IRoom. Middleware con arquitecturas más generales como *UbiComp GAS* y *FedNet* centrado en datos.

Las Plataformas Multi-Agentes establecen un nodo coordinador el cual puede ser dispositivos con mayores capacidades como un PC, Smartphone el cual pueda realizar el proceso solicitado por nodos internos de menores capacidades como sensores y actuadores con el protocolo *Internal Smart Object Protocol* – (ISPO), el nodo coordinador establece comunicación con nodos estratégicos en la red que pueden ser remotos con el protocolo *External Smart Object Protocol* – (ESOP). Los middleware facilitan la interacción entre los actores externos como dispositivos, usuarios y servicios de internet proporcionando servicios de descubrimiento de tipo de objetos, servicios ofrecidos, ubicación y atributos estáticos y dinámicos. En la **¡Error! No se encuentra el origen de la referencia.** se muestran las capas con enfoque de agentes.

Capa	Objetivos Principales	Descripción
Arquitectura de Alto nivel	Gestor de entrada /salida	Recibe y transmite desde y para el mundo exterior datos de comando de comunicación e interrupciones.
	Gestor de comunicación	Comunicación de (SO) con el mundo exterior a través de middleware.
	Comunicación middleware	Comunicaciones con objetos coordinadores remotos.
	Módulo de descubrimiento	Servicios de descubrimiento de (SOs).
	Adaptador de base de conocimiento	Modificación y consulta de bases de conocimiento.
	Gestor de contexto	Interpretación de los datos de usuarios, (SO), internet según el contexto donde se encuentran.
	Razonador	Inferencias a la base de conocimiento y proactivo es decir que tareas ejecutar.
	Adaptador sensor y actuador	Abstracciones de sensores y actuadores, oculta el tipo y la ubicación.
	Externos sensor actuador	No pertenecen al (SO).
Middleware de Agentes	JADE Plataformas coordinador	Arquitecturas de alto nivel middleware derivados JADEX, JADEX Android, JADE LEAP.
	Mobile Agent Platform for SunSPOT - MAPAS	Descubrimiento dinámico de servicios de agente búsqueda de (SOs) con metadatos que contienen tipo, servicios, ubicación, propiedades hardware etc..
Gestión WSAN	Building Management Framework - BMF	Presenta soporte multiplataforma, interfaz de programación, rápida configuración, algoritmos de procesamiento, abstracciones dinámicas. (Signal Processing In-Node Environment - SPINE) propone bibliotecas de alto nivel en java para emitir nodos y solicitar servicios de sensores.

Tabla 10. Capas de Middleware con Enfoques de Agentes

1.5.1.4 Taxonomía de Middleware según las soluciones específicas

Henricksen and Robinson [40] plantean una taxonomía en función de los tipos de soluciones implementan los middleware a nivel de redes de sensores.

Solución	Descripción	Middleware
Database Inspired / Inspirado en dase de datos	Utiliza consultas declarativas SQL con enfoque distribuido y recursos limitados de energía.	COUGAR, TinyDB, SINA.
Tuple Space / Espacio Tupla	Almacenamiento en espacio de tuplas.	TinyLIME, Linda.
Event Based / Base en Eventos	Correlación de eventos de datos de sensores, no es compatible con el modelo Publicación /Suscripción.	MIRES.

Tabla 11. Taxonomía de Middleware según la solución que Implementa

Otros middleware poseen características específicas las cuales son importantes mencionar por separado:

Middleware	Objetivos
COUGAR, TinyDB	Recogida sencilla de datos aplicable a monitoreo ambiental utiliza arboles de enrutamiento semántico, realiza procesamiento con fórmulas aritméticas de

Middleware	Objetivos
	suma y promedio que apoya la conservación de energía, las consultas son de muestras de datos temporales.
SINA	Expresa la consulta con lenguaje SQL-like es una arquitectura completa para redes de sensores con apoyo a consulta SQL y scripting (Senser Query and Tasking Languaje - SQTL) que son primitivas de acceso a hardware, comunicación y manejo de eventos. Almacenamiento del middleware se realiza en una hoja de cálculo asociativo, soporta la movilidad del nodo estación base (sink).
TinyLIME, Linda	Modelo de comunicación Publicación/Suscripción basada en eventos, utiliza memoria compartida con un espacio de tuplas compartidas y federación de nodos.
MIRES	Sistema Publicación/Suscripción utiliza TinyOS sistema operativo para comunicación basada en mensajes, anuncia el tipo el sensor se centra en la arquitectura y problemas de red.
MiLAN	Mecanismo Plug-in para incorporar protocolos arbitrarios, las variables son los tipos de datos que requieren los sensores, utiliza un protocolo de descubrimiento de servicios disponibles en puntos en el tiempo, con sensores virtuales reúne la información de dos o más sensores, no es adecuado para sistemas con escasos recursos usa protocolos (SDP, SLP) proveniente de los sistemas tradicionales distribuidos y heterogéneos.

Tabla 12. Ejemplos de Middleware Orientados a la Solución

Existen middleware que soportan soluciones más genéricas, hardware heterogéneo y diversos mecanismos de comunicación, adicionalmente enfoques que manejan el contexto (context-aware) con consultas más sofisticadas y técnicas de fusión de datos y con el uso de ontologías permite mecanismos de razonamiento que es la extracción de información de forma explícita por ejemplo algoritmos de aprendizaje automático.

1.5.1.5 Taxonomía orientada a los Enfoques Middleware

Las funciones de los middlewares es poner los recursos de sensores a disposición de las aplicaciones, se puede diferenciar cuatro clases de acuerdo a su función. Las funciones pueden superponerse por eso las fronteras de la imagen están difuminados, las tres capas principales pueden tener subcapas dependiendo de la arquitectura de cada middleware. Según Bröring, et al. [41] presenta los cuatro enfoques que pueden abarcar varias capas.

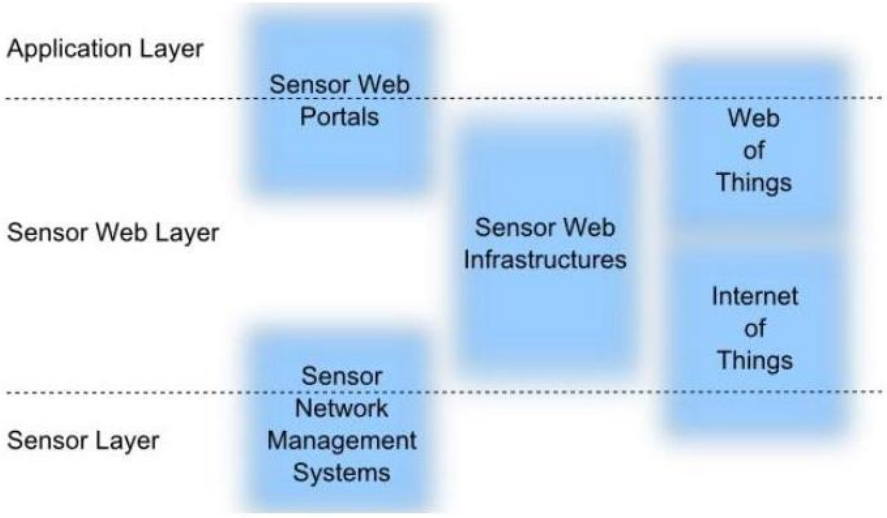


Figura 12. Tres Capas y Cuatro Enfoques de Middlewares en la IoT
Tomado de: A. Bröring, "New generation sensor web enablement"[41].

Las características de cada capa se detallan en la Tabla 13 a continuación:

Capa	Descripción
Sensor Layer / Capa sensor	En la capa de sensor se encuentran los dispositivos conectados y protocolos de comunicación que pueden ser propietario o estándares por ejemplo el protocolo WPAN y el IEEE 1451.
Sensor Web Layer / Capa web sensor	En la capa intermedia es el puente entre y aplicaciones de sensores. Esta capa implementa las representaciones digitales de los mismos y en lo posible utilizando formatos estándar e interoperables.
Application Layer / Capa aplicación	En la capa de aplicación se encuentran las interfaces de interacción con los usuarios o con las maquinas como dispositivos finales.

Tabla 13. Capas Generales de los Middlewares

Finalmente, los enfoques se detallan en la Tabla 14.

Enfoque	Funciones	Descripción
Sensor network manager system / Sistema de gestión de red de sensores	Sistemas de administración de red o bajo nivel a comunicación de red.	Protocolos de enrutamiento Optimización de comunicación Optimización de rutas de recolección Optimización de cobertura de las redes Localización de los sensores dentro de la red
Sensor web infraestructures / Infraestructuras Sensor Web	Fabricación de sensores disponibles en la red.	Acceso a nivel de aplicación Abstracción de la red No proporciona nivel de gestión de red de sensores.
Sensor web Portals / Portales web sensores	API's que presentan funcionalidades de sensores	Funciones básicas de sensores como: Registro, carga, consulta, descubrimiento. Funciones avanzadas de sensores como:

		Central de programación de sensores.
Frameworks for (Internet of Things - IoT)/(Web of Things - WoT)	*Web of Thigs* – (WoT) es como una evolución para la IoT	(WoT) utiliza protocolos de aplicación como *Web Services* – (SW). Servicios de nombres para las cosas (ONS) e identificación por códigos (RFID).

Tabla 14. Enfoques de los Middlewares

Espacio *Tuple Space* [42] es un paradigma basado en la computación de *tuplas* expresados en forma tipletas RDF donde la unidad de información tiene tres dimensiones "objeto, predicado, sujeto". Esta comunicación elimina elementos estructurales de soporte, por tal motivo puede realizarse en entornos informáticos diferentes. Las tuplas están formadas por tripletas RDF y un URI que identifica la tupla. El tuplespace permite razonamiento de RDFS y fue diseñado para la computación ubicua, pero su paradigma cliente- servidor centralizado no le permite ser instalado en dispositivos pequeños sino en máquinas de gran capacidad.

El RFID suite [43] es un middleware que se le aplico los principios de la Computación Orientada a Servicios (*Services Oriented Vision - SOC*) con débil acoplamiento y maneja la heterogeneidad. Las etiquetas RFID son capaces de almacenar pequeñas cantidades de datos, por ejemplo: abono, saldo, URL de objeto. La Arquitectura Orientada a Servicios (*Service Oriented Architecture - SOA*) se caracteriza por su bajo acoplamiento y alta flexibilidad, permitiendo una perfecta integración de plataformas y aplicaciones heterogéneas. El middleware es un sistema multicapa en cada capa en el contexto del RFID. Este middleware fue creado en la Universidad de Grenoble, Francia, es de código abierto en marco del proyecto OW2 AspireRFID1, parte de la proyecto ASPIRE FT7.

El (*Linked Stream Middleware - LSM*)[44] integra fácilmente fuentes de datos en función del tiempo, enriqueciendo los flujos de datos de los sensores con descripciones semánticas y permite complejas consultas SPARQL a través de un motor de consultas que abstrae a los usuarios de las implementaciones concretas, como limitaciones de los recursos y protocolos de acceso. Facilita la consulta y publicación datos dinámicos de los sensores para que sean tan accesibles como los recursos de la web tradicional.

Los componentes[45] tales como: las redes de sensores, los servicios web y las bases de datos se deben combinar para construir aplicaciones en el IoT. La alternativa semántica para combinar estos elementos constituye una potente solución para el manejo de las dificultades que se tienen con los flujos de datos del sensor.

Para el desarrollo de aplicaciones en el IoT, una de las aportaciones más influyentes ha sido el (*Open Geospatial Consortium - OGC*)[46] y del proyecto

(*Sensor Web Enablement - SWE*), los cuales han aportado un estándar en observaciones, mediciones, descripciones de red de sensores, transductores y transmisión de datos en el campo de los diferentes tipos de servicios generados por las fuentes de datos de objetos. El OCG ha apoyado la planificación, alerta, observación, recogida de medición y gestión de redes.

Las tendencias en las soluciones que se han presentado para la integración y fusión de datos heterogéneos utilizando técnicas semánticas como: la *Transformación del flujo de datos de sensores a (Resource Description Framework - RDF), que luego* se publican con (*Hypertext Transfer Protocol – HTTP*) mediante la identificación del sensor con (*Uniform Resource Identifier - URI's*), ampliamente utilizado en el (*Linked Open Data – LOD*). Otra técnica es la *Consulta semántica adaptadas a datos de sensores*, con lenguajes de consultas (*SPARQL Protocol and RDF Query Language – SPARQL*) que genera vistas lógicas para consultar los flujos de datos de sensores.

Finalmente el LOD [47] busca proporcionar una base de datos donde se pueda enriquecer semánticamente los datos de sensores provenientes de distintos fuentes de datos por ejemplo Linked Observation Data, Sensormap y Earthcam cuyo dominio de aplicación es el ámbito tierra y Sensormasher, publican los datos en diferentes formatos interfaces y estándares. Recopila la información de sensores los enlaza con la nube de datos vinculados (*Data Open Cloud - DOC*) que facilita el acceso de fuentes heterogéneas de datos de sensores como: (*Extensible Markup Language - XML*), (*Really Simple Syndication – RSS*) y (*JavaScript Object Notation – JSON*) o texto plano.

1.5.2 Estudio Comparativo de Middleware para el Proyecto

Teniendo en cuenta los trabajos de Molla and Ahamed [48] en redes de sensores inalámbricos, Bandyopadhyay, et al. [49] sobre el papel de los middleware en la IoT y de Al-Jaroodi and Mohamed [50] con un análisis de middleware orientados a servicios, se tomaron sólo los principales middleware creados para la IoT y que se consideraron los que contaban con más información existente para su descripción, podemos hacer un análisis comparativo de acuerdo a sus funcionalidades clave para el desarrollo de la IoT, como se presenta en la **¡Error! No se encuentra el origen de la referencia.**.

MIDDLEWARE	CARACTERISTICAS ANALIZADAS																	Totales	Porcentaje
	Administración de Dispositivos	Interoperabilidad Semántica	Portabilidad de la Plataforma	Conciencia - Contexto	Seguridad y Privacidad	Composición de Servicios	Descubrimiento e Integración	**Abstracción de la Heterogeneidad**	Integración Transparente	**Escalabilidad y Eficiencia**	Requerimientos de QoS	Fusión de Datos	**Restricciones de los recursos**	Topología dinámica	Aplicación del conocimiento	Paradigma de Programación	**Adaptabilidad**		
HYDRA	√	√	√	√	√	√	√	√	√	√			√		√	√	√	**14**	**82%**
LINKSMART	√	√	√	√	√	√	√	√	√	√		√	√	√	√	√	√	**16**	**94%**
ROS	√							√		√			√	√			√	6	35%
ECLOUDRFID	√	√	√	√	√	√	√	√		√		√	√	√	√	√	√	15	88%
ISMB	√		√															2	12%
ASPIRE	√		√					√		√		√		√			√	7	41%
UBIWARE	√		√	√		√	√			√					√			7	41%
UBISOAP	√	√	√			√	√	√	√	√	√		√	√	√	√	√	**14**	**82%**
UBIROAD	√	√	√	√	√			√	√	√		√	√	√	√		√	13	76%
GSN	√		√		√					√				√			√	6	35%
SMEPP	√		√	√	√			√	√	√	√		√	√			√	11	65%
SOCRADES	√	√	√	√	√	√	√	√	x	√	√	√	√	√	√	√	√	**16**	**94%**
SIRENA	√	√	√	√	√	√	√	√	√	√	√		√	√	√	√	√	**16**	**94%**
WHEREX	√	√	√	√				√		√			√		√		√	9	53%
(SI)2	√	√				√	√	√		√								6	35%
IMPALA*	√							√		√			√	√		√	√	7	41%
MATE*	√				√			√					√			√	√	6	35%
TINY DB*	√							√				√	√	√		√		6	35%
AGUILLA*	√							√		√		√	√	√		√	√	8	47%
TINY CUBU*	√		√					√					√	√	√	√	√	8	47%
TINY LIME*	√				√			√				√	√	√		√		7	41%
Totales:	**21**	9	14	9	10	8	8	**18**	6	16	4	8	**16**	15	10	12	**16**		
Porcentajes:	**100%**	43%	67%	43%	48%	38%	38%	**86%**	29%	**76%**	19%	38%	**76%**	71%	48%	57%	76%		

* Middleware basados de TinyOS.

√ Significa que el middleware presenta la funcionalidad o tiene como propuesta implementarla

ˣ Significa que el middleware no presenta la funcionalidad o no se encontró información suficiente para establecerlo

Tabla 15. Comparación de Middleware de la IoT

A partir del análisis de la Tabla 15 y tomando una comparación simple de frecuencia estadística combinada con el principio de Pareto[1] en la que se toma el 20% de las mayores incidencias, podemos establecer varias cosas al respecto:

- Los principales esfuerzos en la creación de middleware en la IoT, han estado enfocados fundamentalmente en su fácil adaptabilidad que permitan

[1] http://es.wikipedia.org/wiki/Principio_de_Pareto. Fue enunciado por Wifredo Pareto e indica cómo el 20% de las incidencias en un fenómeno pueden representar o repercutir en el 80% de las restantes.

la administración de objetos, aplicando abstracciones a la heterogeneidad de los mismos, teniendo en cuenta sus restricciones y buscando escalabilidad y eficiencia.

- Los Middleware que están tocando más aspectos para el desarrollo completo de su funcionalidad son los de los proyectos LinkSmart [51], Hydra [52], Socrades [53] y Ubiroad [54]. Los tres primeros han sido una evolución del mismo Middleware, utilizando los aportes de cada uno y por ello han podido abarcar más temas al respecto y generar mayor impacto. UbiRoad, es un middleware semántico al igual que los anteriores y orientado hacia el contexto, lo cual hace que cubra muchas de las características exigidas a los middleware.
- En el aspecto de la Interoperabilidad semántica, menos de la mitad de los proyectos la han incorporado al middleware, por consiguiente aspectos como contexto, aplicación de técnicas de conocimiento, integración de datos, construcción y composición de servicios son temas que hasta ahora se están trabajando. Esto es lógico debido a la evolución del área de investigación, ya que primero se presentan las arquitecturas y luego la incorporación de servicios.

En ninguno de los trabajos se encontró una mención directa a una indexación semántica, si se encontró información de aplicación de técnicas como ontologías para incluir conocimiento al middleware tanto en la parte de abstracción como en la parte de contexto y servicios. Los índices semánticos podrían aportar bastante en estas áreas de trabajo actual.

1.5.3 Servidores de Objetos como Fuentes de Datos de Sensores

Los sensores generan cantidades de información de orden de terabytes, solo se puede usar eficientemente si se logra recoger, analizar y almacenar en un entorno de información como la web. La creación de nuevas aplicaciones de sensores demanda la disponibilidad de los mismos en diferentes ambientes, para esto se ha creado servicios avanzados como un modelo para hacer seguimiento de su consumo, uno de ellos es la predicción de flujo a nivel de protocolo de red[55], el cual es un modelo que presenta una estimación estadística de los datos que el sensor está monitorizando, en estos modelos no se enfoca este trabajo ya que la perspectiva se centra en la semántica y el uso de metadatos para describir los flujos de sensores.

Uno de los problemas de los recursos de sensores, es que no presentan un formato semántico que describiera su significado, lo que limitaba el uso de los datos de sensores a unos cuantos dominios de aplicación. Una estrategia es adicionar metadatos para describir las características técnicas del sensor, su precisión, condiciones y escenarios de medición, en esta estrategia se fundamenta la Web Semántica de Sensores (*Semantic Sensor Web – SSW*), siendo la

descripción semántica muy útil para usuarios que desean consumir estos recursos y que no tienen conocimiento de la heterogénea naturaleza de los sensores conectados, ya que la semántica simplifica la utilización de servicios de sensores.

Los servidores mediadores [56] de sensores, permiten *integración automática* por medio de servicios de descubrimiento, acceso, asignación de tareas, eventos y alerta. Existen otras aplicaciones que cumplen con la calidad de mediadores de la IoT con funciones específicas como Sensorpedia, Sensormap, SensorBase y Xively. La integración automática de sensores y servicios va más allá de conectar y poner en marcha los sensores, se trata también del desarrollo de aplicaciones con un modelo de integración destinado para distintos objetivos, por ejemplo: prevención de desastres, suscribiéndonos a un servicio de Sensorweb con características del evento como derrames de crudo en el océano, estos servicios tiene identificadores y también características espaciales y temporales. La mediación se lleva a cabo por *descripciones con ontologías y emparejamiento semántico*, la *deducción de conceptos* también es una parte importante de los servicios como por ejemplo hacer conversiones de medidas de datos sensor y otro tipo de *deducciones avanzadas* útiles para dar respuesta a las consultas. Los mecanismos de mediación son el *razonamiento - subsunción* que calcula correspondencia con funciones de reclasificación taxonómica, y metodologías de mediación es la *similitud semántica* la cual ofrece información clasificada para el emparejamiento de sensores y servicios.

1.5.4 Estudio Comparativo de Servidores Mediadores de Objetos para la IoT

Con el fin de poder establecer cuan de los servidores existentes se ajustan más a los requisitos del proyecto, se hizo un estudio comparativo de los más importantes. Los resultados se pueden ver en la **¡Error! No se encuentra el origen de la referencia.**.

CARACTERISTICAS ANALIZADAS																				
PLATAFORMAS	**Soporte Desarrollo Aplicaciones**					**Cobertura**			**Servicios**											
Nº	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Uso de Estándares de metadatos	Uso de formatos de Estructura de Datos	Arquitecturas que Soporta IoT	API de desarrollo para diferentes Lenguajes	Ciclo de vida y Despliegue de Productos	Redes que da soporte	Cantidad de Objetos Conectados*	Cantidad de Empresas Conectadas	Gestión de Metadatos	Gestión Datos en Tiempo Real	Directorio de Objetos - Búsqueda	Servicios de Negocio	Servicios de Personalización y	Seguridad, Privacidad y Autorización	Usuarios en los que se focaliza	Costos	Escalabilidad	Extensa Documentación	Formato de sindicación	Trigger
AMEE	G,Tg	J	R	√	√	I	342	0,26	√	√			√	√	Ep					
Arkessa				√		I,M,St			√	√	√	√	√	√	T	O/P				
Axeda			R,So	√	√	I,M,S	1.2	150	√	√	√	√	√	√	T	P				

CARACTERISTICAS ANALIZADAS																			
PLATAFORMAS	**Soporte Desarrollo Aplicaciones**					**Cobertura**			**Servicios**										
Nº	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**	**18**	**19**	**20**
	Uso de Estándares de metadatos	Uso de formatos de Estructura de Datos	Arquitecturas que Soporta IoT	API de desarrollo para diferentes Lenguajes	Ciclo de vida y Despliegue de Productos	Redes que da soporte	Cantidad de Objetos Conectados*	Cantidad de Empresas Conectadas	Gestión de Metadatos	Gestión Datos en Tiempo Real	Directorio de Objetos - Búsqueda	Servicios de Negocio	Servicios de Personalización y	Seguridad, Privacidad y Autorización	Usuarios en los que se focaliza	Costos	Escalabilidad	Extensa Documentación	Formato de sindicación	Trigger
						t														
Bugswarm			R	√		I		300				√	√		T					
Carriots cloud		J,X	R	√		I				√				√	T,Ep	O/P		√		√
Eart Cam						I,M						√		√	E	P				
Evrythng		J	R	√		I				√		√		√	T,Ep	O		√		
GroveStreams		J	R	√		I,M				√			√	√	T,Ep	O/P		√	Rs	
Sensinode	Co,Lo				√	I,M,St				√		√	√	√	Ep	P				
Nimbits	Xp	J,X	R	√		I				√	√				T	O	√	√		√
Exosite		J		√	√	I,M,St				√	√	√	√	√	T,Ep	O/P	√	√		
Open.Sen.se		J	R	√		I			√	√					T	O		√		
Paraimpu		J	R			I									T	O				
NewAer					√	I,M,St							√		Ep	P				
SensorCloud		C,Xd	R	√		I,M,St			√	√		√	√	√	Ep	P		√		
SensorPedia	G	J	R	√		I,M,St			√	√	√		√		Ep	O	√	√	At	
Sensormap	Tg			√		I,M,St			√	√	√		√	√	T		√	√		
SenseWeb	Tg			√		I,M,St				√	√		√	√	T		√	√		
SensorBase	Em			√		I,M,St				√	√		√	√	T		√	√	Rs	
ThingSpeak		J		√		I		0.01	√	√	√		√	√	T,Ep	O/P		√	Rs	
Thingworx				√	√	I,M,St				√	√	√	√	√	Ep	P	√			
Xively Cloud Services***	G,Tg	J,X,C	R,Sk,M	√	√	I	250	55	√	√	√	√	√	√	T	O	√	√	Rs	√
Yaler		J		√		I				√				√		O			Rs	

* Millones de dispositivos.

Tabla 16. Comparación de Características de Servidores Mediadores de la IoT

Abreviatura	Definición de las abreviaturas	Abreviatura	Definición de las abreviaturas
Tipo de Arquitectura Implementada		*Tipos de Formatos de Estructura de Datos Soportados*	
R	REST (Reprsentational State Transfer)	J	JSON (JavaScript Object Notation)
Sk	Socket	X	XML (eXtensible Markup Language)
So	SOAP (Simple Object Access Protocol)	C	CVS (Comma Separated Values)
M	MQTT (Message Queuing Telemetry Transport)	Xd	XDR (eXternal Data Representation)
Tipos de Metadatos soportados		*Disponibilidad de la funcionalidad*	
G	OGC (Open Geospatial Consortium)	√	Quiere decir que si lo soporta
Em	EML (Environmental Markup Language)	-	Quiere decir que no se encontro información
Tg	Tags (Etiquetas definidas por los usuarios)	*Tipos de Usuarios*	
Xp	XMPP (Extensible Messaging and Presence Protocol)	T	Todo Tipo de Usuarios
Co	COAP (Constrained Application Protocol)	Ep	Empresriales
Lp	6LoWPAN (IPv6 over Low power Wireless Personal Area Networks)		
Tipos de Redes		*Tipos de Costos*	
I	Internet	O	Open: Abierta sin costo
M	Móviles	P	Paga: Se debe pagar por sus servicios
St	Satélite	O/P	Servicios Open y otros sonPagos
Formato de sindicación			
At	ATOM (Atom Publishing Protocol)		
Rs	RSS (Really Simple Syndication)		

Tabla 17. Convenciones de Servidores Mediadores de la IoT.

A los servidores mediadores también se les denomina portales web de objetos. Las principales conclusiones que se extraen de la Tabla 16 son:

- Los portales web permiten acceso a múltiples fuentes de datos de sensores, sobre los cuales se pueden desarrollar aplicaciones que soporten la heterogeneidad de los mismos.
- Los portales de sensores actúan como centros de datos que evitan que los usuarios tengan que llevar a cabo procesos repetitivos tales como: iniciar sesión en diferentes sistemas, adaptar las unidades de medida o comprender la semántica de diferentes archivos de registro.
- Los portales web de sensores pretenden superar el modelo monolítico (Semantic Sensor Web - SSW) por modelos más flexibles, que sean reusables y mantenidas por múltiples comunidades.
- Los portales siguen los principios (REST y Linked Data), REST da independencia de la tecnología usada y de protocolos de comunicación, mientras Linked Data es un marco semántico para publicación de observaciones de sensores.
- Todos los portales soportan soluciones de aplicaciones web de gran éxito social como: Blogs para grupos de confianza para puntos de vista relevantes y publicación de las actualizaciones para aplicaciones externas como LinkedIn, Facebook, Twitter y Youtube.

A parte de la clasificación de las características realizadas en la **¡Error! No se encuentra el origen de la referencia.** para soporte al desarrollo, cobertura y servicios, adicionalmente, se agruparon los diferentes portales web de objetos de acuerdo a sus características interrelacionadas, con el fin de poder tipificar los portales relacionados a sus funcionalidades. En la Tabla 18s se presenta esta clasificación:

Nº de característica	Tipo de Servidor

2, 9, 17	**Orientado a Gestión de Datos**: Apoya el fácil acceso a las observaciones de sensor, el reusó de los datos y el peso ligero de los archivos.
1, 2, 4, 5, 9, 10, 11, 15, 16, 18	**Orientados al Desarrollo**: Apoyan el suministro de datos por medio de mecanismos sencillos, desarrollo rápido de aplicaciones para el usuario y bajo costo de implementaciones en las plataformas abiertas.
1, 3, 11, 14, 19, 20	**Orientados a Compartir Datos**: Apoya las mejores prácticas y las soluciones comprobadas provenientes de la web 2.0 para vinculación y control del contenido.
6, 7, 8, 12, 13	**Orientados a Usuarios**: Apoyo al desarrollo automático de productos que beneficie el reusó y aplicación de los datos.

Tabla 18. Tipos de Servidores Mediadores de la IoT

Aunque los servidores o portales de objetos pueden cumplir varias funcionalidades, es posible establecer cómo algunos privilegian un tipo de servidor particular.

Dado que el proyecto utiliza como fuente de datos los servidores mediadores, es necesario contar con uno o varios que den un buen soporte para el desarrollo de aplicaciones. Se decidió seleccionar los candidatos de acuerdo a los siguientes criterios esenciales: seleccionar servidores que ofrezcan un API de desarrollo con acceso a datos y metadatos, gratuitos, usen formatos estándar, ofrezcan la posibilidad de directorio de objetos para su consulta y arquitectura REST. Además de unas características preferentes que podrían facilitar el trabajo como: enfocado a todo tipo de usuarios, basados en estándares de IoT y que posean extensa documentación. Así nos quedan por eliminación los siguientes: SesnsorMap y Xively Cloud Servises (antes Cosm y PachuBe), como los servidores más adecuados para realizar los casos de estudio.

Para poder obtener el mayor provecho a los metadatos encontrados en la IoT, es necesario tener en cuenta la reutilización de estándares de metadatos y almacenado en estructuras reutilizables y compartibles. Los trabajos previos han creado ontologías para almacenar los metadatos representando los objetos, interrelaciones, servicios, entornos y contexto en la IoT [21, 52, 57, 58]. Estas ontologías son una fuente de información importante que permite construir la ontología del presente proyecto.

Las ontologías [59] se pueden ver como estructuras de almacenamiento de conocimiento formalizado y compartido, las cuales permiten no sólo crear procesos de indexación semántica, sino también procesos de razonamiento sobre las mismas. Para este proyecto, se destaca el trabajo presentado por el grupo denominado “W3C Semantic Sensor Network Incubator group (the SSN-XG)”[60], los cuales han creado una ontología llamada *SSN ontology sensors* (http://purl.oclc.org/NET/ssnx/ssn), la cual permite describir las capacidades de los sensores, el proceso de medición y las observaciones resultantes en una red de sensores semánticos. Esta ontología se puede alinear con otras ontologías con el fin de realizar un trabajo conjunto en la semántica de los objetos. Esta ontología ha reutilizado los demás estándares existentes en la representación de sensores,

tales como SensorML y O&M del Open Geospatial Con-sortium's (OGC) [61], y de servicios web como SensorWeb Enablement (SWE) standards[62].

Finalmente, teniendo en cuenta los datos y metadatos almacenados en los servidores mediadores de la IoT, la clasificación de Mathew, los metadatos de contexto, los estándares de objetos y la ontología SSN, se creó una ontología que unifica todas estas fuentes, permitiendo indexar semánticamente los objetos de la IoT con el máximo enriquecimiento semántico.

1.5.5 Protocolos de Comunicación en el IoT

Los protocolos IP de en las redes *ad hoc* no son eficientes para redes de sensores que son muy cambiantes y de numerosos nodos, por este motivo las arquitecturas de los protocolos deben utilizar técnicas para dividir las redes en grupos y para integración de datos por medio de algoritmos de fusión de datos, apoyando las capacidades limitadas de energía, ancho de banda, procesamiento y, almacenamiento de los dispositivos.

Comparado con las redes inalámbricas tradicionales, tales como: Red móvil ad hoc (MANET), y la red de sistemas celulares, algunas características que se deben tener en cuenta en el diseño de redes de sensores inalámbricos según Al-Karaki and Kamal [63] son:

Factores	Descripción
Node deployment / Despliegue de nodo	***Determinista***: Se colocan manualmente los sensores y las rutas están predeterminadas. ***Aleatorio***: El despliegue al azar de manera no uniforme ni óptimo tendrá múltiples rutas y es necesario un corto intervalo de transmisión debido a las limitaciones de energía y ancho de banda.
Energy consumption without losing accuracy / Consumo de energía sin perder precisión en los nodos	En redes de sensores multi-hop cada nodo tiene un doble papel como (remitente y enrutador) un fallo en las baterías de alimentación causan cambios en la topología de la red y por tal motivos cambios en las rutas de los paquetes.
Data Reporting Model / Modelo de reporte de datos	Detecciones y notificaciones de red dependen del momento de reporte, se clasifican en (time-driven, event-driven, query-driven, and hybrid) para cambios periódicos en los nodos utiliza los reportes periódico y para nodos con cambios drásticos los reportes impulsado por eventos o consultas.
Node Heterogeneity / Heterogeneidad del nodo	Diferencias en las capacidades de proceso, comunicación y energía de los nodos de la red, un nodo de sensores puede tener sus propias funciones específicas, lectura de datos, modelo de informes, limitaciones o calidad del servicio. El funcionamiento de un protocolo depende de las capacidades de los nodos de la red.
Fault Tolerance / Tolerancia a fallos	Con el fallo de los nodos por distintos motivos como falta de energía, condiciones ambientales o daño físico impone a la red la gestión del consumo de energía disminuyéndolo cambiando de

Factores	Descripción
	ruta en nodos con más energía o con un uso adecuado de la redundancia manteniendo la señalización en los nodos existentes.
Scalability / Escalabilidad	El orden del número de nodos es del orden de cientos de miles además los nodos deben estar en estado de sueño hasta que se produzca el evento y los pocos activos deben proporcionar una amplia calidad.
Network Dynamics / Redes dinámicas	La mayoría de las redes asumen nodos estacionarios pero con la movilidad de los nodos de sensores, se dificulta la transferencia desde o hacia los nodos, la estabilidad de las rutas y el consumo de energía se deben gestionar y además el evento monitoreado puede ser dinámico.
Transmission Media/ Transmisión media	Las redes de sensores en su mayoría se fían de la comunicación inalámbrica y por ende de los problemas como el alto error y velocidad de señal apropiada. El control de acceso por (MAC)[64] con protocolos basados en conservación de energía como (TDMA), es comparado con protocolos de contención como (CSMA), un ejemplo es el IEE 802.11 y Bluethoot.
Connectivity / Conectividad	Aunque la red de nodos tenga una alta densidad, la red sigue siendo variable y pierde tamaño debido a los fallos, la conectividad depende de la distribución de los nodos, por tal motivo seria aleatoria, no tendría un orden establecido.
Coverage / Cobertura	Un nodo de sensores posee un rango y precisión que deben ser tomados en cuenta en el diseño de redes y la adopción del protocolo de enrutamiento.
Data Aggregation / Agregación de datos	Para reducir la redundancia generada, los nodos se pueden agregar, combinando los datos de fuentes de nodos diferentes. Las técnicas utilizadas son *Funciones de agregación*: Por ejemplo duplicar represión, mínimos, máximos y promedio. *Fusión de datos*[65]: Un nodo da precisión a la señal combinando las entradas y reduciendo el ruido de la señales.
Calidad de Servicio QoS / Calidad del servicio	Los datos son útiles en cierto periodo de tiempo por lo tanto los retardos o latencia es otra restricción para las aplicaciones. En algunas aplicaciones la conservación de la energía es más importante que la calidad del servicio. Con menos energía en la red es necesario reducir la calidad de los datos para así conservar la energía. Por este motivo en este requisito se tiene en cuenta la energía de la red.

Tabla 19. Restricciones en el Diseño de Redes de Sensores

Para analizar los protocolos, los autores proponen los siguientes enfoques para describir la arquitectura y así poder escoger una infraestructura y sin el ánimo de ser extensivo en todos los aspectos se pueden analizar las siguientes categorías según Akkaya and Younis [66] y Singh, et al. [67]:

Categorías	Descripción
Data-centric / Centrado en datos	Difiere de los protocolos centrados en direcciones, la forma de enviar los datos en los protocolos centrados en direcciones los sensores de origen envía sus datos de forma independiente y en los protocolos centrados en datos, los sensores intermedios realizan agregación de datos para ahorrar energía.
Hierarchical /	Son protocolos de energía eficiente. Agrupa la jerárquicamente la red

Categorías	Descripción
Jerarquia	(Clustering), los grupos (Clusters) son gestionados por un nodo, el nodo cabeza de grupo (cluster head). Este protocolo divide la red en capas y el salto entre capas abarca una gran distancia, si la red fuera de una sola capa, la sobrecarga genera latencia o retardos y no sería una red escalable ni podría comunicarse a grandes distancias. No se recomienda en redes que abarquen superficies extensas presenta agregación de los nodos cabeza de grupo (cluster head) para reducción de datos.
Location-base / Localización	Se direcciona los nodos dependiendo de la localización, el direccionamiento no es un esquema como el IP y se calcula la distancia entre nodos para determina el consumo de energía, se eliminan el número de transmisiones realizando consulta en una región en particular. No es aplicable a redes de sensores debido a las limitaciones de energía.
QoS / Calidad del modelo de servicios	Minimizar el consumo de energía y aumentar la calidad del servicio (QoS) como los retardos, fiabilidad o adaptación a los recursos y tolerancias a fallos, los protocolos deben encontrar un equilibrio entre consumo de energía y calidad del servicio. La secuencia asignación de ruta (Sequential Assigment Routing - SAR) tiene como métrica la energía de la ruta de acceso y la prioridad de los paquetes con el mantenimiento de las tablas crea sobrecarga. La Conciencia de energía (energy-aware) establece como ruta más eficiente la de menor costo de energía, el algoritmo de costo tiene en cuenta la reserva de energía, la energía de transmisión, las tasas de error y los parámetros de comunicación, el protocolo realiza gestión de colas pero no proporciona ajuste flexible de ancho de banda.
Network-flow / Flujo de red	Centrados en la configuración de la ruta y resuelve problemas del flujo de red.

Tabla 20. Categorías de Arquitecturas de Redes de sensores

Se han desarrollado categorías que apoyan la movilidad de las redes de sensores (mobility-based) con protocolos de energía eficiente que garantice la entrega de datos desde los nodos hasta las estaciones base (sink) móviles, también categorías que apoyan múltiples rutas (multipath-based), con protocolos que distribuyen uniformemente la energía en las rutas, también se ha desarrollado una categoría que apoya la transmisión de los datos de entidad de nodo (entity-aware)[68], los cuales no usan los dos campos para transmitir la (entidad y los datos del nodo) sino un vector de tamaño fijo que puede ser combinado por los nodos intermedios en un solo vector y así economizar energía en la red. Recientemente se están planteando protocolos de calidad del servicio (QoS-aware)[69] para red de sensores inalámbricos a nivel de capa de red y para middleware a nivel de capas superiores que median entre la solicitud del usuario y la red de sensores.

Relación de protocolos con sus características diferenciables en función de las taxonomías establecidas según Akkaya and Younis [66]:

Protocolo	Data-centric	Hierarchical	Location-based	QoS	Network-flow	Data Aggregation
SPIN	√					√
Directed Diffusion	√					√
Rumor	√					√
Shah and Rabaey	√		√			
GBR	√					√
CADR	√					
COUGAR	√					√
ACQUIRE	√					
LEACH		√				√
TEEN y APTEEN	√	√				√
PEGASIS		√				√
M. Younis		√	√			
Subramanian and Katz		√				√
MeCN y SMECN			√			
GAF		√	√			
GEAR			√			
Chan y Tessiulas		√			√	
Kalpakis			√		√	
SAR				√		
Akkaya		√		√		
SPEED			√	√		

Tabla 21. Relación de protocolos con sus características diferenciables en función de las taxonomías

Los protocolos de enrutamiento a nivel de red utilizan estrategias para el descubrimiento de la topología de red como: Inundaciones (Flooding), es la difusión de mensajes a todos los vecinos y la versión mejorada es el (Gossiping), que selecciona al azar el los vecinos a los que le envía mensajes para establecimiento de rutas.

Para los conceptos a nivel de red se han creado abstracciones a nivel de aplicación, desarrollando protocolos que permiten interoperabilidad entre

dispositivos, lenguajes y aplicaciones. Una relación de protocolos utilizados en dispositivos del dominio del hogar en Hoang [70] se puede ver la Tabla 22.

Protocolo de Interfaz	Objetivos
Lonwork, X10	Protocolos de bajo nivel, es el enfoque de normas eléctricas también conocido como LonTalk ANSI / EIA 709.1 para control de red eléctrica, sigue el modelo (Open Systems Interconnection - OSI). Introduce el concepto de variable de red como abstracción por ejemplo de temperatura o estados de un interruptor.
Plug and Play (UPnP)	Estándar fácil de usar y flexible a los consumidores, es un conjunto de protocolos y servicios para permitir control simultaneo de electrodomésticos, permite la creación de APIs propias, es independiente del sistema operativo, lenguaje y hardware. Utiliza TCP/IP. Unión dinámica de red (AutoIP) y aprende de las capacidades de los dispositivos conectados. Los servicios se describen en formato XML y a través de un puente se puede comunicar con otras no (UPnP), aprovecha muchos protocolos estándares como el TCP/IP. Con servicios como (Simple Service Discovery Protocol – SSDP) localiza recursos y anuncia disponibilidad, (Generic Event Notification Architecture – GENA) envía y recibe notificaciones, (Simple Object Protocolo de Acceso – SOAP) llama a procedimientos remotos.
Jini	Plataforma para sistemas distribuidos complejos basados en java (*Java Remote Method Invocation - RMI*), está compuesto por servicios jini, clientes jini y servicios de búsqueda jini, que es equivalente a un directorio o índice de servicios disponibles, es considerada una plataforma adecuada para la integración de protocolos a bajo nivel y mecanismos de descubrimiento.
OSGi	Proporciona estándares para programación de dispositivos remotamente, es compatible con LonWorks, Jini, UPnP, sobre una puerta de enlace de servicio permite múltiples servicios, los servicios tienen una interfaz java y el modelo arquitectónico asegura interoperabilidad entre middleware heterogéneos usando las APIs, los usuarios cargan los servicios bajo demanda independiente del middleware proveedor. La especificación no es lo suficientemente flexible para manejar los complejos requisitos de seguridad entre módulos. Hay paquetes que se pueden descargar de uso libre, por ejemplo el servicio http y los servicios registro.
oBIX - Open Building Information Xchange	Permite la integración de las aplicaciones empresariales, utiliza estándares XML y SW de servicios web y el estándar REST, es decir sus servicios son accesibles a través de la web y compatible con otras tecnologías, por ello es considerado el middleware de próxima generación para la automatización del hogar.

Tabla 22. Protocolos Utilizados en Redes de Sensores del Hogar

De los protocolos anteriores podemos decir que no son completamente compatibles, además de complejos para su implementación, con una comunidad reducida de desarrolladores.

En cuanto a protocolos de los objetos inteligentes (Smarth Things), se han desarrollado por cuenta de las empresas de productos electrónicos de

entretenimiento y hogar incorporando procesadores y sistemas de almacenamiento con conexión a internet, que son capaces de ejecutar sistemas operativos ligeros e implementar tecnologías como: *Device Profile for Web Services* - DPWS o *Digital Living Network Alliance* -DNLA, ambos implementaciones de la tecnología UpnP, para la gestión, descubrimiento y control multimedia. También ofrecen los APIs de desarrollo de manera abierta con el fin de generar aplicaciones para sus dispositivos.
El DPWS define una arquitectura en la cual los dispositivos corren dos tipos de servicios: servicios de almacenamiento (*hosting services*) y servicios almacenados (*hosted services*). El primero está asociado al dispositivo, jugando un papel importante en el proceso de descubrimiento, el segundo está asociado a la funcionalidad ofrecida y depende el primero.

El DLNA o Alianza para el estilo de vida digital en red, es una asociación de fabricantes de electrónica e informática sin ánimo de lucro fundada por Sony en junio de 2003. Su objetivo es definir directrices de interoperabilidad que permitan compartir medios digitales entre dispositivos de consumo como ordenadores, impresoras, cámaras, teléfono móviles y otros dispositivos multimedia. Estas directrices toman como base estándares públicos ya existentes, pero sólo pueden obtenerse previo pago.

Todos estos protocolos juegan un papel importante al momento de elegir un middleware apropiado para implementar la solución.

1.6 Aplicaciones del Internet de Objetos IoT

Las aplicaciones [12] del IoT podrían mejorar la calidad de vida de las personas. Los objetos equipados con inteligencia primitiva como: autos, electrodomésticos, dispositivos médicos, máquinas para el trabajo y el deporte están generando servicios útiles para sus usuarios, si se les da la capacidad de razonar y comunicarse entre ellos, permitirá generar más escenarios de aplicación, en la cual la interacción multiplicaría los servicios provistos a sus usuarios. En la Tabla 23 se presentan ejemplos de aplicaciones del IoT.

Escenarios	Descripción
Transporte	**Logística**: Wal-Mart y Metro trabaja el inventario con seguridad, debido al seguimiento en tiempo real que realiza a la cadena de suministros con tecnologías RFID y de campo de comunicación cercana (*Near Field Comunication – NFC*) en compra de materias primas, producción y transporte, almacenamiento, distribución y comercialización. Con un software *Enterprise Resource Planning* – (ERP) en tiempo real, informa a los clientes sobre la disponibilidad de productos. **Conducción asistida**: Los vehículos equipados con sensores, actuadores y capacidad de procesamiento, pueden proporcionar información importante al conductor y pasajeros. Sistemas para evitar accidentes, información sobre el

Escenarios	Descripción
	estado de los productos transportados. **Entradas y los celulares**: Los carteles y paneles informativos pueden ser equipados con etiquetas (NFC) o marcadores visuales (*Quick Response Code* - QR Code), por medio del celular se recoge la información de los servicios asociados. **Parámetros ambientales**: El seguimiento de productos perecederos como frutas carnes verduras, monitoreando el estado de conservación y control de temperatura y humedad. **Aumentar los mapas**: Identificar un mapa con un (NFC) y relacionarlo con un servicio web en el cual se entrega información sobre carreteras, hoteles, restaurantes, monumentos y eventos. Las técnicas (*Physical mobile interaction* - PMI)[71] pueden emplearse para aumentar la información de etiquetas.
Cuidado Médico	**Seguimiento**: Es la autenticación de los pacientes y objetos, además de la monitorización de sus signos vitales. El rastreo está destinado a la identificación de una persona u objeto en movimiento. **Identificación y autenticación**: Es usado para cumplir los requisitos y procedimientos de seguridad y evitar los robos y pérdidas. **Recolección**: Es adquisición de la información de los pacientes como historial clínico, automatización en el procesamiento de formularios atención, gestión de inventarios y procesos de auditoria. **Sintiendo**: Es el diagnostico de las condiciones del paciente, un ejemplo es la telemedicina que supervisa el cumplimiento de los pacientes con la toma de medicamentos, otro ejemplo es la atención ambulatoria por medio de acceso remoto y por último monitorizar la movilidad de los pacientes.
Entornos Inteligentes	La inteligencia contenida en los objetos permitirá facilitar el trabajo en la oficina, la industria, el hogar o para la recreación. **Viviendas y oficinas confortables**: Es poder hacer la vida más cómoda por medio de sensores y actuadores, por ejemplo: la calefacción puede adaptarse a nuestras preferencias y clima, la iluminación de la habitación puede variar de acuerdo a los momentos del día. Los accidentes pueden evitarse con sistemas de seguimiento y alarma, además se puede ahorrar energía con sistemas apagado automático. **Industria**: Se etiquetan elementos, a la etiqueta se le asigna un evento que es activado cuando se le inalámbricamente por la unidad de procesamiento maquina o robot, así mismo se puede controlar la calidad del proceso por medio de sensores, por ejemplo de vibración, el cual se conecta a una pieza y dependiendo del umbral de vibración se activa el evento para detener el proceso. Seguimiento en tiempo real el ERP de pérdidas y el progreso de la producción. **Museos y gimnasios inteligentes**: En los museos se pueden interactuar con la información con mayor calidad en exposiciones y en el gimnasio a las maquinas se les puede programar la rutina de las personas.
Personal y Social	**Dominio Personal y social**: El establecimiento de relaciones sociales por medio de mecanismos de interacción, como programar mensajes a los amigos a través de etiquetas RFID, pueden generar la creación de eventos con personas y lugares de manera intuitiva. **Redes Sociales**: Actualización automática de la información de las redes sociales a través de eventos programados por el usuario utilizando etiquetas RFID, puede actualizar la información en tiempo real. **Consultas históricas**: Para realizar consultas sobre los datos y eventos históricos de los objetos permitiendo desarrollar aplicaciones que soportan

Escenarios	Descripción
	actividades a largo plazo, como una agenda digital en Google calendar, donde el usuario puede consultar en que ha gastado su tiempo. También generando lugares con API grafica de Google, que hace seguimiento de sitios y fechas. **Pérdidas y Robos**: la utilidad puede ser el seguimiento de objetos que no recordamos donde ha quedado por medio de aplicaciones de búsqueda.
Futuristas	En el campo de comunicaciones se pueden desarrollar sistemas de detección, desarrollo de materiales y tecnologías de procesos industriales, un ejemplo es el proyecto SENSEI[72]. **Taxi Robo**: Ideado para las ciudades del futuro donde la prestación de este servicio sea de manera eficiente, reduciendo la congestión de tráfico y sin la necesidad de un conductor humano, y el usuario se ubica por GPS. **Modelo de información de ciudad (*City Information Model - CIM*)**: Es una aplicación donde se sigue el estado y rendimiento de cada uno de los elementos de una ciudad como edificios, calles, cloacas, vías férreas, estaciones de autobuses y pasos peatonales. **Sala de juego**: Mejorar la sala de juego equipándolas con una cantidad de dispositivos para detectar la ubicación, movimiento, aceleración, información visual, ruido, voz, temperatura y humedad.

Tabla 23. Aplicaciones del IoT.

La presente propuesta realiza un caso de estudio en el campo de la contaminación medioambiental en dos regiones de Colombia, para lo cual pretende aplicar el modelo conceptual aportado y un buscador semántico que retorne información relevante.

1.7 Recuperación de la Información

Recuperación de la Información (*Information Retrieval - IR*): Es una de las áreas más antiguas en investigación en las ciencias de la información. La (IR) busca y recupera los documentos más relevantes requeridos por las necesidades de los usuarios. En la Tabla 24 se presentan las principales características.

Aspectos de la (IR)	Descripción
Interfaces	El diseño de la interfaz conlleva un compromiso a mejorar la usabilidad, interfaces simples para consultas ambiguas e interfaces complejas para consultas detalladas o precisas. Por ejemplo las interfaces basadas en palabra clave.
Procesador de Consultas	La consulta primero debe interpretarse antes de proceder a la búsqueda, en este proceso contempla varias tareas por ejemplo buscar las palabra vacías o derivadas.
Indexación	La primera tarea de la indexación es optimizar la consulta y los tiempos almacenando la ocurrencia de cada término en una estructura invertida de funcionalidad eficiente.
Búsqueda	En el índice invertido se buscan todos los términos que contienen ocurrencias.
Clasificación	Se otorga una puntuación a los documentos recuperados en la fase de

Aspectos de la (IR)	Descripción
	búsqueda, en función de la calidad de correspondencia de los términos de la consulta y los documentos. Este proceso es dependiente del modelo de recuperación utilizado.

Tabla 24. Principales características de la Recuperación de la Información.

La elección de la métrica de evaluación es fundamental para puntuar los objetos clasificados y seleccionar los que se presentaran como primeros en la información de respuesta.

Por otro lado la recuperación semántica de la información (SIR), busca extender el campo de la Recuperación de la Información (*Information Retrieval* - IR) al no poder recuperar el significado de la información que se transporta, un problema que se presenta es el nivel polisémico de las palabras, el significado que lleva una palabra en diferentes oraciones y sus relaciones con otras palabras de la frase, el proceso de encontrar el significado correcto de las palabras en la frase se conoce como desambiguación (*Word Sense Disambiguation* - *WSD*), existen técnicas estadísticas o modelado de dominio, el modelo de dominio que describa los conceptos relevantes y un modelo estocástico que aprovecha la información lingüística eliminando la ambigüedad de las palabras[73].

1.7.1 Modelos de recuperación de información

Los modelos de representación de documentos y consulta tienen gran importancia en el rendimiento y la recuperación, se centra el trabajo en los modelos clásicos que comprenden los modelos Booleano, Vectorial y probabilístico, aunque existen modelos basados en lógica formal, lógica fuzzy e inteligencia artificial (ver Tabla 25)

Modelos de la (IR)	Descripción
Booleano	Un conjunto de términos y las consultas representan a los documentos son expresiones booleanas, se requiere de una coincidencia exacta para la recuperación.
Espacio Vector (VSM)	Vectores de términos ponderados representan los documentos y las consultas, la recuperación se realiza según el grado se similitud con vector consulta y tiene la capacidad de recuperar los documentos parcialmente coincidentes.
Probabilístico	La recuperación de los documentos se realiza teniendo en cuenta las probabilidades de pertenencia al conjunto.

Tabla 25. Aspectos de Recuperación de la Información.

Existen más modelos, basados en técnicas de inteligencia artificial y otras áreas importantes para el descubrimiento de conocimiento, sin embargo para el alcance del presente proyecto, se utilizará uno de los modelos básicos en los datos recuperados de los sensores para el proceso de indexación y en la medida que se

vaya avanzando en el modelo se irán colocando particularidades propias de la adaptación en la IoT.

1.7.2 Estructuras de indexación

Las estructuras de indexación registran los datos con cierto orden creando un índice, los motores de búsqueda semántica indexan recursos automáticamente usando bases de conocimiento ontológicas que recuperan la información utilizando conceptos o significados. Existen tres técnicas de búsqueda de documentos: semi – estructurados el esquema de Índice Nodo que utiliza el etiquetado de nodos, el esquema Índice Grafico que utiliza índices secundarios o resúmenes y el esquema Índice de estructura secuencial que requiere de técnicas de cadenas coincidentes para responder a las consultas.

Para el proyecto se debe plantear un modelo de búsqueda de alto rendimiento que sea aplicable a grandes colecciones de datos en la web, las unidades básicas que se desean recuperar están compuestas por un conjunto de atributos, valores y relaciones entre entidades, y el esquema está compuesto por un índice invertido con forma de árbol que realiza consultas sobre grafos RDF con empleo de palabras clave [74].

1.7.3 Expansión de consultas

Las consultas y los índices pueden ser expandidos utilizando la semántica de las palabras es decir, sus sinónimos y sentido de una palabra. Se ha demostrado que la expansión de los índices o *Synsets* puede mejorar el rendimiento cuando se le aplica un método de desambiguación WSD y se sabe que el rendimiento depende da la calidad de la WSD.

1.7.4 Evaluación en recuperación de información

Existen algunos índices o medidas, que son utilizados para evaluar la calidad de la información recuperada, a continuación se nombran algunas de las más usadas [75]:

- **Exhaustividad**: consiste en recuperar todos los documentos que puedan ser relevantes de una colección, de acuerdo a los parámetros sugeridos por el usuario.
- **Precisión**: Esta medida representa el porcentaje de los documentos recuperados, que son relevantes de acuerdo a determinada consulta. Se calcula así: documentos relevantes recuperados, sobre, total de documentos recuperados.

- **Relevancia**: Mide la importancia que el documento recuperado representa para el usuario, según sus necesidades de información.

El presente proyecto reutilizará estos indicadores, los cuales serna detallados más en la sección de evaluación y propone otros indicadores adaptados a las características de la IoT.

1.8 Web Semántica

Es la web de datos vinculados, donde se ha desarrollado un conjunto de herramientas y tecnologías para modelar, anotar, buscar e integrar los datos. La web es un conjunto de documentos interconectados o hipertexto y lo que busca es que se pueda fácilmente navegar a través de las conexiones, tarea fácil para un humano, pero no para una maquina como un agente autónomo, ya que una página es solo un archivo html, por tal motivo la semántica le da sentido al documento debe ser identificada. Las entidades de un documento pueden ser identificadas de forma única y poseen un conjunto de propiedades, las relaciones entre las entidades permiten la integración de datos, su reutilización y la automatización.

1.8.1 Metadatos

Una de las piedras angulares sobre la cual se basa la Web Semántica son los metadatos, ya que con ellos los programas de la Web semántica pueden obtener el significado y el contexto del recurso recuperado. Los metadatos corresponden a un conjunto de propiedades o atributos necesarios para etiquetar, catalogar, describir y clasificar[2] información de una fuente o recurso[76]. Un ejemplo de lo anterior en un sentido amplio son las fichas bibliográficas, los registros de auto descripción de las bases de datos o simplemente las propiedades de los archivos digitales, pero en un sentido estricto con respecto al contexto de la Web Semántica, Berners-Lee y la W3C los definen como: “Los metadatos son información inteligible para el ordenador sobre recursos Web u otras cosas”.

De los diversos modelos de metadatos existentes se puede establecer que en general presentan información del contenido, especificaciones formales (tamaño, formato, idioma, etcétera), derechos de autor, autenticación y finalmente el contexto. La forma de almacenarlos depende de las necesidades, es posible almacenarlos con el mismo recurso o en estructuras separadas.

---

[2] Metadatos. María Jesús Lamarca Lapuente. Hipertexto: El nuevo concepto de documento en la cultura de la imagen. http://www.hipertexto.info/documentos/metadatos.htm. (27/01/10).

Los metadatos pueden ser utilizados por los motores de búsqueda Web con el fin de mejorar la precisión de los resultados devueltos a los usuarios. El gran problema es que actualmente existe un porcentaje muy bajo de recursos con metadatos, lo cual hace que los motores deban aplicar técnicas de recuperación de la información directamente del contenido de los mismos recursos.

Existe una organización internacional denominada Iniciativa Dublin Core (DCMI)[77], la cual ha marcado desde 1995 la pauta en la definición de estándares internacionales para describir los recursos Web y facilitar su recuperación. Inicialmente se propusieron un pequeño grupo de descriptores que permitieron estandarizar metadatos hacia los recursos bibliográficos. Posteriormente debido al gran interés despertado por otros sectores como la educación, las empresas y demás proveedores de información, se amplió el conjunto de metadatos[3] y se estableció una forma estándar para describirlos[4].

Sin embargo, existen varias iniciativas para definir metadatos en otras áreas del conocimiento particular, dado que la especificación DCMI no puede cubrir todos los requisitos específicos, es así como se encuentran metadatos en humanidades y lingüística a través de TEI[5], biología[6], ciencias geoespaciales[7] y recursos educativos[8] entre otras. Una lista completa de especificaciones de metadatos se puede encontrar en la página Web de Lamarca[78].

La forma de escribir los metadatos se puede de varias formas:

1. Definición de una propiedad y de un valor para la misma. Ésto se puede hacer de dos maneras:
    a. Meta – etiquetas HTML: Las últimas especificaciones HTML han definido una etiqueta ***meta***, la cual permite fácilmente incluir con el mismo documento la información de atributos. Ej. *<META name="description" content="Esta es la descripción general de la página">*.

---

[3] Términos de metadatos Dublin Core (DCMI Metadata Terms) http://dublincore.org/documents/dcmi-terms/.

[4] Codificación Dublin Core en HTML (IETF RFC 2731): http://www.ietf.org/rfc/rfc2731.txt. . (27/01/10).

[5] TEI: Text Encoding Initiative. http://www.tei-c.org/index.xml. (27/01/10).

[6] National Biological Information Infraestructure. Página Principal de Metadatos. http://www.nbii.gov/portal/community/Communities/Toolkit/Metadata/. (27/01/10).

[7] Federal Geographic Data Committe. Página Princioal de Metadatos Geoespaciales. http://www.fgdc.gov/metadata. (27/01/10).

[8] IEEE Learning Technology Standards Committe. Learning Object Metadata (LOM) Working Group 12. http://www.ieeeltsc.org:8080/Plone/working-group/learning-object-metadata-working-group-12/learning-object-metadata-lom-working-group-12. y IMS Learning Resource Meta-data. http://www.imsproject.org/metadata/. (27/01/10).

   b. Fuera del Documento con un Enlace: Con este método se utiliza una etiqueta *<LINK>* en el *<HEAD>,* la cual permite especificar el documento en el cual se encuentran los metadatos.
2. Definición de un perfil en el cual se define la propiedad y sus valores permitidos. Para esto, se usa el atributo <PROFILE> en el elemento <HEAD>. Adicionalmente, proporciona un medio para establecer diferentes contextos dependiendo de los objetivos del usuario.
3. Utilización del Marco de Descripción de Recursos (RDF): Es una especificación del W3C, la cual define estructuras más complejas, flexibles y estandarizadas para escribir los metadatos de los recursos Web. Un ejemplo de ella es la especificación Dublin Core. Utilizan archivos externos basados en XML para describir los metadatos. Es importante mencionar que el RDF no sólo define datos acerca de los datos, sino que también permite definir clases y reglas de inferencia, pero no negaciones y disyunciones. Adicionalmente, es posible agregar varias descripciones creadas en diferentes momentos y con propósitos diferentes.

Para poder obtener el mayor provecho a los metadatos es necesario tener en cuenta la reutilización de estándares de metadatos ya escritos y aceptados ampliamente como Dublim Core. Por otro lado, es importante en las aplicaciones de recuperación de información como los buscadores semánticos, deben tener a la mano vocabularios controlados[9], taxonomías[10] y/o tesauros[11], los cuales permiten que se defina una semántica compartida por una comunidad, de tal forma que su interpretación sea igual para muchos usuarios.

Finalmente, como se puede observar, los metadatos deben fundamentarse en repositorios de conocimiento para poder utilizarlos de una forma productiva para todos, es por ello que desde la Inteligencia Artificial se dió una respuesta con el concepto de Ontologías, las cuales permiten organizar los términos de los lenguajes humanos y organizarlos de tal forma que podamos establecer relaciones de jerarquía y semántica en un dominio del conocimiento, dando la posibilidad también de realizar operaciones de raciocinio basados en la lógica. Así dotaremos a los ordenadores de las capacidades para interpretar semánticamente las solicitudes de los usuarios y obtener una respuesta más inteligente de la

[9] Un Vocabulario Controlado en la Web puede verse como una lista cerrada de términos, que se usan para clasificar, ya que hacen referencia de manera univoca a un solo sujeto. Ej. http://www.wikipedia.com.

[10] Una Taxonomía se puede entender como un vocabulario controlado al cual se le ha establecido una jerarquía semántica. Ej. En Hipertext.net. http://www.hipertext.net/web/pag264.htm. (27/01/10), presenta una documentación relacionada con la construcción de taxonomía para clasificar información en sitios Web.

[11] Un Tesauro se entiende como una Taxonomía a la cual se le han añadido enlaces para definir relaciones entre los diferentes conceptos del vocabulario controlado tomado como base. Ej. http://www.dmoz.org. (27/01/10).

información recuperada ya que los metadatos de los recursos se comparan con éstas ontologías y los contextos de los usuarios.

Para el presente proyecto, los metadatos y sus tecnologías son una fuente importante de información a tener en cuenta para el logro de los objetivos, ya que los servicios Web semánticos deben autodescribirse y los algoritmos de búsqueda toman sus metadatos como base para la clasificación y recuperación de la información.

En el siguiente apartado se profundizara un poco más en las Ontologías, ya que se considera como una de las estructuras más importantes para crear aplicaciones de la Web Semántica.

1.8.2 Ontologías

1.8.2.1 Historia del Término Ontología

La palabra Ontología proviene del griego, idioma en el que significa “estudio del ser”[79] de los fundamentos y explicaciones de todas las cosas. Tales de Mileto y Anaxímenes de Mileto (se preocupaban por buscar el origen de todo), Pitágoras (las cosas, fenómenos y la vida real eran explicadas o podían ser interpretadas mediante los números o expresiones matemáticas). Desde entonces, varios filósofos y científicos acuñaron el término adecuándolo a la disciplina que trabajaban[59, 80].

Tiempo después Christian Wolff popularizó en círculos filosóficos la palabra “ontología”, la cual es usada como un método demostrativo racional y deductivo. Siguiendo a Wolff, Alexander Baumgarten definió ontología “como la ciencia de los predicados más generales y abstractos de todo”, de forma que a ella pertenecen los principios cognitivos fundamentales del pensamiento humano[80].

Así, muchos más ontólogos filósofos intentaron revelar los aspectos comunes de las cosas para entenderlas de manera tal que el término ontología ingresó a otros campos del conocimiento, como la inteligencia artificial (IA) y la informática[59].

En los últimos años se ha aplicado el término ontología en otros campos del conocimiento donde es considerada algo más que una herramienta para el análisis superficial de algún fenómeno, a través de la cual es posible extraer conocimiento difícil de ver desde el exterior. Razón por la cual los científicos en Inteligencia Artificial han incluido a las ontologías como herramienta para ayudar a cumplir con la inmensa tarea de imitar el comportamiento y razonamiento humano.

En la informática las ontologías aparecieron como herramienta que permite y facilita el proceso de compartir y reutilizar el conocimiento. Los estudios más relevantes son realizados en áreas como la ingeniería del conocimiento, procesamiento del lenguaje natural o representación del conocimiento. Más recientemente, la noción de ontología se ha popularizado en campos como integración inteligente de información, sistemas cooperativos de información, recuperación de información, comercio electrónico y gestión de conocimiento. La razón por la cual las ontologías son ahora tan populares es, en gran medida, debido a lo que prometen: una comprensión compartida y común de algún dominio que puede ser comunicado entre individuos y aplicaciones[59, 80].

La Web Semántica se presenta como una evolución de la Web actual, la cual está dotada de mecanismos que le permiten realizar operaciones de información de manera inteligente, permitiendo a los usuarios obtener respuestas rápidas y precisas, gracias a una definición adecuada de la información, la cual esta dotada con significado compartido y entendido por máquinas y humanos. Ésto último, hizo que los investigadores vieran en la Ontologías una de las piedras angulares del desarrollo de la Web Semántica, dado que es necesario un mecanismo que permita organizar y procesar el conocimiento, tareas propias de las Ontologías.

1.8.2.2 Concepto de Ontología

Existen varios conceptos de Ontologías, como vimos, desde la filosofía hasta su aplicación en la Informática en diversos campos, pero el concepto que más se acerca al trabajo que se realiza en la presente es la propuesta por Borst (1997): "*Una ontología es una especificación formal de una conceptualización compartida*". En este contexto, "conceptualización" se refiere a un modelo abstracto de algún fenómeno en el mundo, a través de la identificación de los conceptos (términos) relevantes de dicho fenómeno. "Explícita" significa que el tipo de conceptos y restricciones usados se definen sin ambigüedades; "formal" representa el hecho de que la ontología debería ser entendible por las máquinas y "compartida" refleja la noción de que una ontología captura conocimiento consensual; esto se relaciona con que es aceptado por un grupo de expertos[59].

En esencia, las ontologías son muy similares a las técnicas de modelado conceptual existentes, como el Modelo Entidad Relación o el Lenguaje Unificado de Modelado UML[81]. Sin embargo, las ontologías difieren de estos métodos y tecnologías existentes en tres aspectos: Primero, el objetivo principal de las ontologías es permitir el consenso sobre el significado de un vocabulario específico de términos y, de esta forma, facilitar la integración de información entre aplicaciones; Segundo, las ontologías son formalizadas en lenguajes de representación basados en lógica, aspecto que permite que su semántica sea especificada sin ambigüedades; Tercero, los lenguajes de representación permiten

realizar consultas y especialmente razonamientos a través de cálculos en tiempo de ejecución

1.8.2.3 Elementos de las Ontologías

La conceptualización que pretende representar una ontología se especifica generalmente usando cinco tipos de componentes: conceptos, relaciones, funciones, axiomas e instancias[80], tal como se muestra en la Tabla 26.

Elemento de una Ontología	Definición
Conceptos o clases	Son las ideas básicas que se intentan representar. Los conceptos pueden ser clases de objetos, métodos, planes, estrategias, procesos de razonamiento. En la Web Semántica los conceptos pueden ser términos claves de los documentos que se analizan en las búsquedas.
Relaciones	Representan la interacción y enlace entre los conceptos de un dominio. En algunos casos suelen formar la taxonomía del dominio. Por ejemplo: subclase-de, parte-de, parte-exhaustiva-de, conectado-a, etc.
Funciones	Son un tipo concreto de relación donde se identifica un elemento mediante el cálculo de una función que considera varios elementos de la ontología. Por ejemplo, pueden aparecer funciones como: asignar-fecha, categorizar-clase, etc.
Reglas de restricción	Los axiomas son expresiones que son siempre ciertas. Pueden incluirse en una ontología con muchos propósitos, tales como definir el significado de los componentes ontológicos, definir restricciones complejas sobre los valores de los atributos, argumentos de relaciones, etc. Las ontologías que poseen una gran cantidad de axiomas se denominan ontologías pesadas, en contraste con las ontologías ligeras que no incluyen axiomas.
Instancias	Representan una manifestación concreta de una clase (un objeto con valores determinados).

Tabla 26. Elementos de una Ontología

Actualmente se han desarrollado muchas Ontologías de Dominio como WordNET [82] y CyC [83], las cuales están a disposición de los usuarios de Internet para intégralas o interoperarlas con las aplicaciones como los Buscadores Semánticos[12], los cuales pretenden mejorar la precisión de las consultas, ya sea refinando los resultados de los buscadores tradicionales o creando un buscador propio.

[12] En esta Web se encuentra una Lista de los principales Buscadores Semánticos existentes http://www.javi.it/semantic.html.

1.8.2.4 Aplicaciones y Ventajas del Uso de Ontologías

Las principales aplicaciones de las Ontologías [59, 84-86] se resumen en la **¡Error! No se encuentra el origen de la referencia.**:

Área de aplicación	Definición
Integración de Información	Un campo de aplicación prometedor de las ontologías es su uso en la integración de fuentes de información heterogéneas a nivel de esquema. Esto debido a que en muchas ocasiones, bases de datos diferentes almacenan el mismo tipo de información empleando diferentes modelos de datos. En este escenario, una ontología puede emplearse para mediar entre los esquemas de las bases de datos, permitiendo integrar la información de diferentes fuentes e interpretar los datos de una fuente bajo el esquema de otra. De allí que el Comercio electrónico sea una de las áreas más beneficiadas de estas tecnologías.
Recuperación de Información	La recuperación de información desde documentos es el mayor campo de aplicación para las ontologías. La idea detrás de la recuperación de información basada en ontologías es incrementar la precisión de los resultados obtenidos teniendo en cuenta la semántica de la información contenida dentro de las consultas como en los documentos, pasando con esto, de palabras clave a conceptos y relaciones ontológicas
Administración de conocimiento	Al interior de compañías u otras asociaciones organizadas, el conocimiento individual puede verse como un recurso estratégico que es susceptible de compartirse y mantenerse sistemáticamente dentro de la denominada gestión de conocimiento. En este aspecto, las ontologías proporcionan un modelo conceptual compartido de un dominio, que permite unificar la administración de conocimiento, conectando los sistemas técnicos para la navegación, almacenamiento búsqueda e intercambio del mismo
Sistemas expertos	Este tipo de aplicaciones permite formalizar el dominio de conocimiento de un experto, como por ejemplo el de diagnóstico médico o el de asesoramiento legal, mediante una ontología de dominio sobre la cual se presentan interrogantes sofisticados que son resueltos empleando mecanismos de razonamiento
Aplicaciones de Web Semántica	Una de las aplicaciones que más fuerza ha tomado últimamente son las Ontologías para la Web Semántica. Se están desarrollando Ontologías para compartir conocimiento entre redes sociales en línea hasta el “Ontology Learning”, que se conoce como la construcción automática de Ontologías de Dominio para reutilizar su conocimiento obtenido de cualquier recurso en la Web.

Tabla 27. Aplicaciones de las Ontologías

Los beneficios de utilizar ontologías se pueden resumir de la siguiente forma:

- Proporcionan una forma de representar y compartir el conocimiento utilizando un vocabulario común.
- Proporcionan un protocolo específico de comunicación.
- Permiten una reutilización del conocimiento.

Sin embargo, debido a los diferentes usos de las Ontologías, formas y ámbitos de conocimiento, es necesario establecer claramente los diferentes tipos de Ontologías, tal como lo presenta el siguiente apartado.

1.8.2.5 Clasificación de las Ontologías

Actualmente existen varios criterios que permiten clasificar los diferentes tipos ontologías con base en algunos parámetros[59]. Por esta razón, a continuación se presentarán algunos de los más relevantes encontrados en la literatura.

Tipos de Clasificación	Descripción
Clasificación según el ***campo y el uso*** de la conceptualización	Esta clasificación está determinada por el nivel de complejidad empleado para describir los conceptos y las relaciones entre ellos[80, 86]: • **Ontologías terminológicas, lingüísticas**: Especifican los términos usados para representar el vocabulario de un dominio. • **Ontologías de información**: Especifican la estructura de los registros de la base de datos. Los esquemas de bases de datos serían un ejemplo. • **Ontologías para modelar conocimiento**: Especifican conceptualizaciones de conocimiento. Estas ontologías tienen una estructura interna mucho más rica que los anteriores tipos de ontologías y éstas son las ontologías que interesan a los desarrolladores de sistemas basados en conocimiento. Otra clasificación por complejidad es la siguiente: • **Ontologías del dominio:** Corresponden a ontologías que contienen todos los conceptos asociados a un dominio particular. • **Ontologías de tarea**: Establecen la forma en la cual se puede usar el conocimiento del dominio para realizar tareas específicas. De esta forma, una aplicación podría realizar búsquedas de información mientras otra podría gestionar la asignación de bloques de memoria libre. • **Ontologías generales:** Contienen descripciones generales sobre objetos, eventos, relaciones temporales, relaciones causales, modelos de comportamiento y funcionalidades.
Clasificación por ***motivación[59, 80]***	Dentro de esta clasificación se distinguen las siguientes categorías: • **Ontologías para la representación de conocimiento**: Permiten explicar las conceptualizaciones que subyacen de los formalismos de representación de conocimiento • **Ontologías genéricas**: Definen conceptos considerados genéricos en diferentes áreas. Ejemplos de tales conceptos serían componente, subclase, proceso, estado, etc. Estas ontologías son reutilizables en diferentes dominios. Se llaman también ontologías abstractas o super teorías porque permiten definir conceptos abstractos. • **Ontologías del dominio**: Definen conceptualizaciones específicas del dominio. Las metodologías actuales de

Tipos de Clasificación	Descripción
	adquisición de conocimiento distinguen entre ontologías y conocimiento del dominio; el último describe situaciones actuales del dominio, mientras que las ontologías imponen descripciones sobre la estructura y contenido del conocimiento del dominio. • **Ontologías de aplicación**: Están ligadas al desarrollo de una aplicación concreta. Tales ontologías cubren los aspectos relacionados con aplicaciones particulares. Típicamente, estas ontologías toman conceptos de ontologías del dominio y genéricas, así como métodos específicos para realizar la tarea, por lo que no son muy adecuadas para ser reutilizadas.
Clasificación según el ***ámbito de conocimiento[59, 80]***	• **Ontologías generales**: Son las ontologías de nivel más alto ya que describen conceptos generales (espacio, tiempo, materia, objeto, etc.). • **Ontologías de dominio:** Describen el vocabulario de un dominio concreto del conocimiento. • **Ontologías específicas:** Son ontologías especializadas que describen los conceptos para un campo limitado del conocimiento o una aplicación concreta.
Clasificación según el ***tipo de Agente***	Permite la clasificación de ontologías teniendo en cuenta el agente (software o humano) al cual vayan destinadas: • **Ontologías lingüísticas:** Se vinculan a aspectos lingüísticos, esto es, a aspectos gramáticos, semánticos y sintácticos destinados a su utilización por los seres humanos. • **Ontologías no lingüísticas**: Destinadas a ser utilizadas por robots y agentes inteligentes. • **Ontologías mixtas**: Combinan las características de las anteriores.
Clasificación según el ***nivel de abstracción y razonamiento lógico***	Esta clasificación permite diferenciar las ontologías considerando el nivel de abstracción y el grado de razonamiento que permitan realizar a partir de la forma de descripción empleada: • **Ontologías descriptivas:** Incluyen descripciones, taxonomías de conceptos, relaciones entre los conceptos y propiedades, pero no permiten inferencias lógicas. • **Ontologías lógicas**: Permiten inferencias lógicas mediante la utilización de una serie de componentes como las axiomas formales.

Tabla 28. Tablas de Clasificación de las Ontologías

De acuerdo con lo anterior, para el presente trabajo se utilizarán en mayor medida las Ontologías de Dominio con el fin de capturar los conceptos básicos y sus relaciones de dominios de conocimiento particulares. Por otro lado, es posible que se necesiten las Ontologías lingüísticas, ya que muchos de los procesos de la Web Semántica requieren un análisis desde el texto, según la fuente que se trabaje. Finalmente, se debe incorporar algunos procesos de las Ontologías de Conocimiento o de raciocinio lógico, ya que se requiere que la Ontología permita dar respuestas a consultas de clientes o agentes externos.

1.8.2.6 Metodologías para la Construcción de Ontologías

Similar a cualquier componente software, la construcción de una ontología puede ser mejorada mediante la aplicación de algún tipo de metodología. El propósito de emplear una metodología es procurar un óptimo resultado siguiendo un conjunto de pasos, los cuales usualmente están basados en procedimientos cuyo éxito ha sido comprobado en la práctica. Las ontologías no son la excepción a esta regla, y de esta manera la mayoría de metodologías destinadas a su construcción están basadas en la experiencia de personas o grupos involucradas(os) en su desarrollo [59]. En varios casos, estas metodologías son extraídas de la manera en la cual ha sido construida una ontología particular, de la cual han heredado su nombre.

Las metodologías propuestas en la literatura se pueden clasificar en dos grupos: (1) las destinadas a la creación de ontologías desde cero, y (2) las destinadas a la reutilización de ontologías disponibles[85].

Para el presente proyecto, es importante estudiar las diferentes metodologías con el fin de establecer si las existentes se adecuan a los objetivos previstos, o por contrario, si es necesario realizar una propuesta en este sentido. En la Tabla 29 se presenta un resumen de las principales metodologías encontradas[87, 88].

Metodología	Descripción
CYC	Esta metodología fue propuesta a partir del proyecto desarrollado en el área de la Inteligencia artificial, que tenia por objetivo habilitar el razonamiento humano. La metodología Cyc recomienda los siguientes pasos: 1. **Codificación manual de conocimiento**: Consiste en la extracción de conocimiento a partir de documentos, sin la utilización de sistemas de procesamiento de lenguaje natural o de aprendizaje. Este paso se realiza de forma manual, debido a que dichos sistemas no cuentan con el suficiente vocabulario común que les permita efectuar una adecuada búsqueda de conocimiento. 2. **Codificación de conocimiento asistido por herramientas software**: Consiste en la utilización parcial de herramientas de procesamiento de lenguaje natural y de aprendizaje, con el propósito de descubrir nuevo conocimiento común a partir del conocimiento obtenido en el paso anterior. 3. **Codificación de conocimiento realizado por herramientas software:** En este paso el descubrimiento de nuevo conocimiento es realizado principalmente por las herramientas. La función del desarrollador consiste en recomendar las fuentes de conocimiento y explicar algunos fragmentos de texto.
USCHOLD Y KING	Esta metodología nace de la experiencia obtenida en el desarrollo de "*Enterprise Ontology*", una ontología para modelar los procesos empresariales. Los pasos propuestos son los siguientes: 1. **Identificar el propósito de la ontología:** Consiste en determinar el porqué de la construcción de la ontología y cuáles serán los usos de la misma. 2. **Construir la ontología**: Comprende el desarrollo de la ontología a través de tres actividades. La primera consiste en capturar la ontología

	a través de la identificación de los conceptos y relaciones entre conceptos, así como los términos utilizados para referirse a dichos elementos. La segunda actividad consiste en codificar la ontología empleando un lenguaje formal. La tercera actividad consiste en integrar la ontología obtenida con alguna ontología existente. 3. **Evaluar la ontología:** Consiste en realizar un juicio técnico de la ontología, entorno software asociado y documentación con respecto a un marco de referencia (especificación de requerimientos, preguntas de competencia, etc.).
GRÜNINGER Y FOX	Esta metodología surgió del desarrollo del proyecto TOVE (Toronto Virtual Enterprise), dentro del cual se desarrollaron un conjunto de ontologías para el dominio de los procesos de negocios. Esta metodología propone los siguientes seis pasos: 1. **Identificar los escenarios de motivación**: Aquí se reconocen las posibles aplicaciones en las cuales podrá emplearse la ontología. 2. **Elaborar preguntas informales de competencia**: Las cuales son realizadas en lenguaje natural y son empleadas con la finalidad de determinar el alcance de la ontología. 3. **Especificar la terminología utilizando lógica de primer orden**: Se realiza la formalización de la terminología extraída a partir de las preguntas de competencia realizadas en el paso anterior. 4. **Especificar preguntas formales de competencia:** Consiste en formalizar las preguntas de competencia, para lo cual las preguntas empleadas en el paso dos son re-escritas en lógica de primer orden. 5. **Especificar los axiomas empleando lógica de primer orden:** Esta metodología propone hacer uso de axiomas formales para definir los términos y restringir su interpretación dentro de la ontología. 6. **Establecer las condiciones para calificar la integridad de la ontología:** Consiste en definir las condiciones bajo las cuales la ontología puede dar respuesta a las preguntas de competencia que han sido formalizadas.
Metodología KACTUS	La metodología KACTUS basa la construcción de una ontología (que representa el conocimiento requerido por una aplicación) en la reutilización de conceptos definidos en otras ontologías[88]. De esta manera el proceso de construcción está condicionado al análisis y adaptación de conceptos de otras ontologías. Comprende las siguientes etapas: 1. **Especificación de la aplicación**: Se identifica el contexto de la aplicación a través de una lista de términos relacionados del dominio. 2. **Diseño preliminar de la ontología:** Comprende la evaluación de las ontologías superiores disponibles con el fin de seleccionar sus categorías relevantes, las cuales posteriormente son redefinidas y/o extendidas para finalmente incluirlas en la nueva ontología. 3. **Refinamiento y estructuración:** Se descompone la ontología para reorganizar su estructura con el fin de obtener un diseño modular.
Methontology	Es una metodología creada en el Laboratorio de Inteligencia Artificial de la Universidad Técnica de Madrid. Define un conjunto de actividades para su creación: 1. **Actividades de administración del proyecto:** Estas actividades involucran la planeación, seguimiento de tareas y control de la calidad con el fin de obtener una ontología de calidad. 2. **Actividades orientadas al desarrollo:** Dentro de las cuales se encuentran**:** • **Especificación:** La cual consiste en definir el alcance y la

	granularidad de la ontología. • **Conceptualización:** Aquí se realiza la organización y estructuración del conocimiento adquirido a través de un medio de representación (tablas, diagramas UML, etc.) que sea independiente del formalismo y lenguaje de implementación en que vaya a ser formalizada o implementada la ontología. • **Implementación:** Consiste en trasformar el modelo conceptual obtenido en un modelo formalizado, el cual es implementado a través de un lenguaje de ontologías como OWL. • **Mantenimiento:** Define el ciclo de vida de la Ontología. 3. **Actividades de soporte:** Incluyen la recopilación de conocimiento, evaluación de la ontología, reutilización de la ontología y documentación.
Sensus	Esta metodología permite derivar ontologías de dominio a partir de ontologías de grandes proporciones a través de un enfoque *top-down* (de lo general a lo particular). Para realizar esto, los autores proponen identificar un conjunto de términos semilla que son relevantes dentro de un dominio particular. Una vez obtenidos dichos términos, éstos se enlazan manualmente a una ontología de grandes proporciones, que en este caso corresponde a la ontología Sensus (contiene más de 70,000 conceptos). A continuación se ejecuta un algoritmo que retorna todos los términos que se encuentran entre la trayectoria de los términos semilla y la raíz de Sensus. Si por algún motivo se considera que falta algún término que sea relevante dentro del dominio, entonces se procede a enlazar el término a la ontología y posteriormente ejecutar nuevamente el algoritmo de búsqueda de términos
On-To-Knowledge	La metodología On-To-Knowledge (OTK) está enfocada hacia la construcción de sistemas basados en conocimiento, en los que se empleen las ontologías como parte fundamental. OTK define las siguientes etapas para la creación de una ontología: 1. **Estudio de factibilidad:** OTK sigue la metodología CommonKADS con el propósito de establecer si se justifica la construcción de la ontología. 2. **Inicio:** Se determina el alcance y el objetivo de la ontología. Además de esto se identifican las entidades más importantes así como las fuentes de conocimiento y los expertos humanos que puedan ser consultadas(os). 3. **Refinamiento:** En esta etapa primero se extrae el conocimiento relevante desde las fuentes y expertos humanos identificados en la etapa anterior, para luego proceder a su formalización. 4. **Evaluación:** Permite verificar la utilidad de la ontología desarrollada y de su entorno software asociado. 5. **Mantenimiento:** Se establece quién es el responsable de la labor de mantenimiento y cómo ésta debería llevarse a cabo.
Terminae[89]	Terminae aporta tanto una metodología como una herramienta para la construcción de ontologías a partir de textos. Se basa en un análisis lingüístico de los textos, el cual se realiza mediante la aplicación de diferentes herramientas para el procesamiento del lenguaje natural. En particular se usan dos herramientas: (1) Syntex para identificar términos y relaciones; y (2) Camaleón para identificar roles o relaciones. Estas herramientas se basan en la misma hipótesis lingüística: el significado de las frases y las palabras es específico para un dominio y puede ser inferido de la observación de regularidades en documentos. La metodología funciona como sigue: Mediante la aplicación de Syntex obtenemos una lista de posibles palabras y frases del texto y algunas dependencias sintácticas y gramaticales entre ellas. Estos datos se usan como entrada para el proceso de modelado junto con el texto original. De esta forma,

	la identificación de conocimiento se basa en dos tareas que se realizan alternativamente: 1. Explorar los resultados Syntex para identificar conocimiento importante o decidir cómo representar alguna información de acuerdo con el uso de las palabras en el texto. 2. Extraer sistemáticamente del texto tanto conocimiento como sea posible. Cada pieza de conocimiento puede ser representada en el modelo de conocimiento de Terminae, cuyo lenguaje de representación de conocimiento posee las siguientes primitivas: fichero terminológico (términos), conceptos genéricos (clases), conceptos primitivos (instancias) y roles (relaciones). El siguiente paso es normalizar el conocimiento para obtener una ontología bien estructurada, donde cada concepto quede justificado por sus relaciones con otros conceptos. Esta metodología sugiere aplicar criterios diferenciadores para hacer explícitas las propiedades comunes y diferentes de un concepto con sus respectivos conceptos padre y hermanos debidas a sus roles. La última etapa es la formalización de la ontología en el lenguaje formal Terminae, que es un tipo de lógica descriptiva. Una función de clasificación sirve para comprobar la corrección de las definiciones de conceptos genéricos, ya que sólo pueden ser definidos si tienen roles diferenciados

Tabla 29: Metodologías para Crear Ontologías

Después de analizar las principales metodologías en la construcción de Ontologías se puede decir que el proyecto actual podría utilizar como referente las metodologías *KACTUS y Sensus*, ya que presentan la forma de reutilizar Ontologías existentes para crear nuevas Ontologías. Adicionalmente, *Methontology y Terminae*, se convierte en un referente importante ya que se presenta como una metodología que utiliza herramientas software ya creadas que permiten extraer de manera automática términos y relaciones, elementos importantes en el Ontology Learning, que es el área en la cual se propone el presente trabajo y se profundizara más en los siguientes apartados.

1.8.3 Anotación Semántica

Aunque podemos entender la anotación semántica como un mecanismo por el cual dotamos de metadatos a los recursos Web, esto es posible a través de mecanismos manuales o semi-automáticos, en los cuales interviene un experto, pero también se debe entender en un contexto automático y controlado, como una estrategia para inferir conocimientos a partir del texto, utilizando como base las Ontologías y así poder entregar al usuario conocimiento relacionado al objetivo inicial.

En el contexto anterior, aparece la teoría de la Semántica Ontológica[90] propuesta por Nirenburg, la cual estudia el significado del lenguaje humano con el fin de aproximarlo al procesamiento de lenguaje natural (PLN), para lo cual utiliza las Ontologías como estructuras para extraer y representar el significado de textos en lenguaje natural, al razonar con el conocimiento que se deriva a partir de

textos. Así, esta teoría se convierte en la base para la Anotación Semántica de los recursos Web.

De lo anterior, podemos definir la Anotación Semántica Automática como: “una información sobre las entidades o conceptos de una ontología que aparecen en un texto y su situación en el mismo, o también las referencias que hay en un texto sobre un repositorio semántico en el que hay más conocimiento”[76]

En la se presenta un ejemplo de Anotación Semántica. Se puede observar como los términos del documento son enlazados a las clases (conceptos) de una Ontología existente en un repositorio de información, permitiendo posteriormente inferir más información de los términos del documento y posiblemente relacionarlos con términos diferentes a los consultados por el usuario, pero relacionados. Por ejemplo saber que XYZ es una compañía radicada en Londres y que Bulgaria es un País.

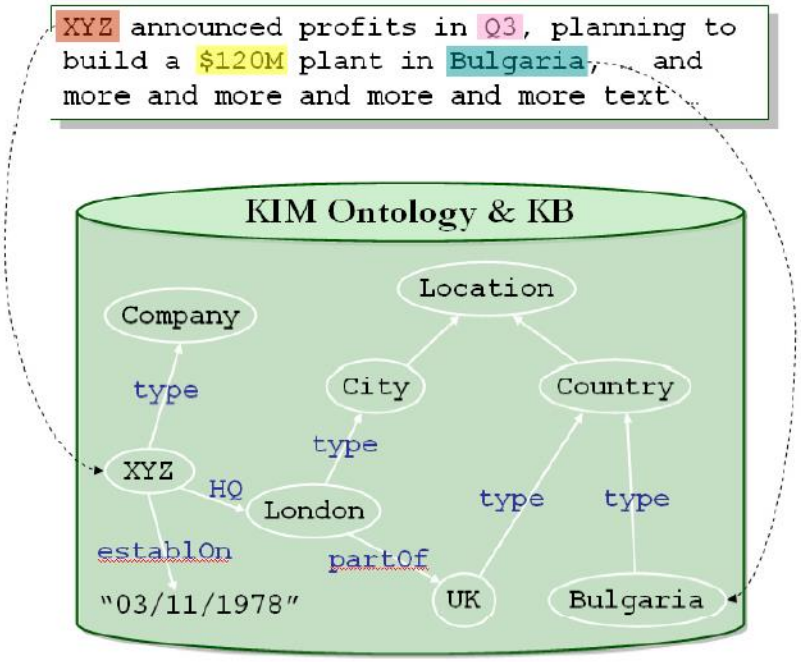


Figura 13. Ejemplo de Anotación Semántica
REF: Tomado de Kiryakov[91], página 2.

El proceso que se llevan a cabo para realizar la Anotación Semántica Automática en general se puede resumir en los siguientes pasos[91]:

1. ***Extracción***: Es uno de los pasos más complejos, ya que la información textual encontrada en los recursos Web no están estructurados y generalmente responden a semánticas particulares y formatos propios de cada organización o propietario del recurso. Esto puede ser resuelto

parcialmente con la aplicación de Ontologías, pero depende de que tan genérica o aceptada sea la misma y el contexto del mismo recurso.

2. ***Indexación y Recuperación***: Este proceso es más complicado de lo que parece, ya que no basta con sólo con encontrar la relación directa de pregunta respuesta. Los términos utilizados pueden cambiar su semántica dependiendo del contexto en que se utilicen, además de tener que asumir que no hay problemas sintácticos y que los usuarios se están refiriendo a las cosas con los términos adecuados, entre muchos otros problemas. Es así como este campo tiene aún mucho trabajo por recorrer.

Existen entornos (Framework) de trabajo de anotación semántica[92], que pueden ser utilizados para realizar este proceso a los recursos Web. Entre ellos mencionamos los principales:

- ***Annotea***[13]: Es un proyecto en el cual se comparte la información anotada a través de unos servidores. Las fuentes son documento HTML o XML. Utiliza el lenguaje RDF para los metadatos y Xpointer para localizar las anotaciones en un documento anotado. Algunos clientes que implementan este framework son el editor y navegador *Amaya*, el plugin *Annozilla* y *Vannotea* un anotador se videos.
- ***CREAM***[14] (CREAting Metadata for the Semantic Web): Usa RDF o DAML + OIL como esquemas de anotación y XPointer como estándar de localización. Este framework ofrece la posibilidad de anotar bases de datos relacionales.
- ***Cerno***[15]: Es un sistema de anotación que usa el lenguaje de programación TXL[16] que está especialmente diseñado para soportar análisis por computadora y tareas de transformación, es decir, que a través de una gramática de entrada y el documento a anotar se obtiene una estructura árbol con la cual se pueden generar reglas de anotación. Aquí no se usa una arquitectura tan compleja como las anteriores, y aún necesita la intervención del hombre quien aporta la gramática.

Finalmente, existen herramientas que apoyan la anotación semántica, éstas se pueden clasificar en dos grupos[76]:

---

[13] Sitio Web del Proyecto Annotea http://www.w3.org/2001/Annotea/. (27/01/10).

[14] Artículo Web de CREAM. Siegfried Handschuh, Steffen Staab. "Authoring and Annotation of Web Pages in CREAM". http://www2002.org/CDROM/refereed/506/. (27/01/10).

[15] Artículo: [93] N. Kiyavitskaya, N. Zeni, J. R. Cordy, L. Mich, and J. Mylopoulos, "Cerno: Light-weight tool support for semantic annotation of textual documents," *Data & Knowledge Engineering*, 2009. http://research.cs.queensu.ca/~cordy/Papers/KZCMM_DKE_Cerno.pdf. (27/01/10).

[16] Sitio Web: http://www.txl.ca/. (27/01/10).

1. Herramientas de Anotación Externa: Permiten crear metadatos de documentos HTML utilizando RDF. Almacenando la información en repositorios externos para su posterior uso.
2. Herramientas de Autor: Permiten incluir información estructura en los mismos recursos mediante XML o RDF.

En ambos casos, las anotaciones se almacenan con respecto a una Ontología que clasifica los términos. El estudio de Uren[92], clasifica la mayoría de las herramientas, aportando las fortalezas y debilidades de cada una, sin embargo entre las herramientas más conocidas podemos mencionar:

- ***Amaya***[17]: Es un editor Web que permite desarrollar sitios adicionando metadatos a las páginas Web que se están construyendo.
- ***OpenCalais***[18]: Utiliza técnicas de PLN y de aprendizaje de máquina para reconocer conceptos, relaciones y hechos. Se puede utilizar un servicio Web para utilizar su funcionalidad.

La Anotación Semántica es unas de las técnicas que debe contemplar el presente proyecto ya que se basa en la utilización de Ontologías que deben estar relacionadas con metadatos, para posteriormente crear los servicios Web semánticos que permitirán el acceso a la información. Se debe escoger cuidadosamente las herramientas y los métodos a utilizar para esto.

1.8.4 Ontology Learning

Existe un concepto en el desarrollo de la Web Semántica: el Aprendizaje Ontológico -"Ontology Learning"[94], el cual consiste en la extracción de conceptos de diversas fuentes y su posterior organización (creación, modificación) en una ontología. Lo anterior se logra con el uso de métodos como: Procesamiento de Lenguaje Natural - "Natural Language Processing" (NLP), Inteligencia Artificial – "Artificial Intelligence" (AI) y Aprendizaje de Máquina – "Machine Learning" – (ML).

Se han creado diversas técnicas[95] desde las áreas descritas anteriormente o una combinación de ellas, para extraer conceptos clave de las diferentes fuentes de información en la Web. Sin embargo, es necesario establecer la efectividad de las técnicas creadas, con el fin de garantizar la evolución correcta de ésta área de trabajo.

---

[17] Sitio Web Amaya: http://www.w3.org/Amaya/. (27/01/10).

[18] Sitio Web OpenCalais: http://www.opencalais.com/. (27/01/10).

Actualmente se están construyendo plataformas de procesos de Aprendizaje Ontológico[94] (Frameworks of Ontology Learning Proceses), con el fin de evaluar que técnicas o combinación de ellas son más efectivas para recuperar conceptos relevantes a partir de un documento Web al cual se le aplican diversas técnicas tales como: Adquisición de conocimiento, aprendizaje de máquina, recuperación de la información, procesamiento de lenguaje natural, razonamiento de inteligencia artificial y administración de bases de datos [96-102].

1.8.5 Semantic Web Services

Por otro lado, se está aprovechando la tecnología de los servicios Web para crear lo que hoy se conoce como Servicios Web Semánticos o "Semantic Web Services"[103]. Los servicios Web permiten acceder a los recursos de las organizaciones sin tener que conocer de la lógica del negocio de las mismas, utilizan lenguajes estándar de comunicación entre máquinas como el XML[104] y bajo una arquitectura orientada a servicios, denominada SOA[105], la cual permite que un cliente pueda consumir servicios e interoperar conjuntamente. Los servicios Web se vislumbran como la solución para resolver las necesidades de las aplicaciones, mediante la integración de diversas soluciones en la Web.

Para poder usar un servicio es necesario que las aplicaciones puedan descubrir, describir e invocarlos, para esto se han creado protocolos estándar como WSDL, UDDI y SOAP[106]. Dado lo anterior, actualmente es posible consumir un servicio Web, pero existen limitaciones para expresar las capacidades y requerimientos de los mismos. Esta falencia en la representación semántica de los servicios impide que se pueda lograr una integración automática de las aplicaciones de servicios Web. La solución al problema anterior es la creación de Servicios Web Semánticos, los cuales deben permitir que las aplicaciones puedan no solo descubrirlos y consumirlos, sino entender su alcance y posibilidades con el fin de que puedan decidir su utilidad para lograr los objetivos de procesamiento e integración automática[103].

Para la creación de Servicios Web Semánticos se han definido nuevos estándares para su creación e interoperación como OWL-S, WSMO y WSDL-S[107], los cuales nuevamente utilizan los conceptos de las Ontologías para definir la semántica de los mismos.

Finalmente existen herramientas[107] que están siendo probadas para crear y manipular Servicios Web Semánticos, tomando como base el estándar WSMO. Se han desarrollado componentes de infraestructura (parsers), librerías estándar y ambientes de ejecución.

Con respecto a la infraestructura tenemos[107]:

- WSMO API: la cual es una interfaz de programa de aplicación escrito en Java que provee una interfaz genérica para acceder de manera programática a una descripción con WSMO, soporta Ontologías WSMO, Servicios Web, Objetivos y Mediadores, adicionalmente incluye una implementación por defecto llamada WSMO4J, la cual implementa todas las interfaces del API, éstos proveen una base para ser utilizados por editores y razonadores.
- WSML Rasoner Framework[19]: el cual agrupa diferentes técnicas y formas de razonamiento para ser utilizadas por las aplicaciones.
- WSML Validator[20]: Es un servicio en línea para establecer la validez de una especificación Web Service Modeling Language (WSML), verificando errores de sintaxis y sugerencias de arreglo.
- WSML reasoning service[21]: Otro servicio en línea que soporta consultas de respuesta de una especificación WSML.

Con respecto a las herramientas de diseño tenemos:

- Plataforma ECLIPSE: Provee un entorno de desarrollo que puede integrar API de diferentes para el desarrollo de software en Java. Este entorno puede integrar el WSMO API y así reconfigurar sus herramientas para trabajar los editores de servicios Web que posee ECLIPSE.
- Web Service Modeling Toolkit (WSMT): Es un proyecto del Digital Enterprise Research Institute (DERI). Combina un conjunto de editores y visores. Soporta modelado, ejecución y monitoreo de Servicios Web. Esta herramienta está integrada a ECLIPSE.

Integrated Reasoning Support o WSML2Reasoner framework: Provee una interfaz que le permite al usuario ejecutar consultas sobre una Ontología. Se encuentra integrada a ECLIPSE.

- Mediación de Datos o Data Mediation: Este concepto lo ha implementado el WSMT. El editor gráfico hace una aproximación de cómo se podría transformar instancias de una Ontología Origen a una Destino, el usuario puede cambiar las reglas de transformación si lo requiere.
- WSMO Studio: Es una colección de herramientas para editar descripciones WSMO, desarrollado para ECLIPSE por el OntoText Labs[22]. Soporta Ontologias, Servicios Web, Objetivos y mediadores, permitiendo manjar subtipos de esas entidades.

[19] http://kaon2.semanticweb.org/.

[20] http://tools.deri.org/wsml/validator/.

[21] http://tools.deri.org/wsml/rule-reasoner/.

[22] http://www.ontotext.com.

- DOME: El DERI Ontology Management Environment (DOME)[23], es un conjunto de herramientas para administrar documentos WSML desarrollados por el Ontology Management Working Group[24]. Puede ser utilizado para editar, navegar, direccionar y combinar Ontologías, manejando también sus versiones.

Con respecto a los ambientes de ejecución tenemos:

- Web Service Execution Environment (WSMX): Es una implementación de WSMO y es un entorno de ejecución en el cual las descripciones semánticas de los objetivos de los usuarios y los proveedores de servicio pueden utilizar para descubrir, componer, mediar, seleccionar e invocar Servicios Web que coinciden con los requerimientos de los usuarios. El WSMX tiene una arquitectura basada en componentes bien definidos para ser utilizados fácilmente por cualquier aplicación, ya sea cliente o proveedor de servicio[107].
- Internet Reasoning Service IRS-III[108]: Es un framework y una plataforma de implementación que actúa como un intermediario (broker) entre los objetivos de un usuario y los Servicios Web implementados. Soporta capacidad para invocar Servicios Web, fácil de usar, oculta los procesos internos de interoperabilidad, capacidad para realizar solicitudes externas a Servicios Web, además de una documentación semántica de los servicios construidos.

Finalmente, las aplicaciones que tiene esta tecnología son muy interesantes, aunque por ser el WSMO relativamente nuevo no hay muchos casos de éxito. Por un lado está el Comercio Electrónico. Como ejemplo tenemos un proyecto denominado “WSMO Description of the ECS Web Service”, el cual implementa la semántica de los Servicios Web de AMAZON[25] permitiendo automatizar los requerimientos de los productos pedidos por los usuarios y las información que suple los mismos, dejando que los usuarios o terceros administren los carros de compras e informaciones complementarias, pero haciendo los pedidos directamente a la compañía. Por otro lado, tenemos aplicaciones en B2B, lo cual permite que las empresas realicen sus transacciones de manera automática sin intervención humana, mediante un conjunto bien definido de Servicios Web Semánticos de parte y parte, con el fin de compartir la información y transformarla en los formatos particulares de cada empresa. Otra área importante es e-gobierno, en la cual se pueden automatizar todos los procesos que un gobierno

[23] http://dome.sourceforge.net.

[24] http://www.omwg.org

[25] http://www.Amazon.com

realiza con sus ciudadanos a través de servicios bien definidos en portales Web. Finalmente en bancos, cuya funcionalidad seria similar a los ejemplos anteriores.

1.8.6 Servicios Web en la Recuperación de Información

Hasta ahora los servicios Web, normalmente han sido utilizados para soportar el desarrollo de aplicaciones distribuidas, principalmente comercio electrónico, gobierno y salud[103]. Sin embargo, con las tecnologías de la Web Semántica algunos investigadores han visto la posibilidad de combinarlos con técnicas de RI, con el fin de mejorar sus resultados cuando sean utilizados por las aplicaciones.

Una de las aplicaciones de la combinación de estas dos técnicas se encuentra en las Bibliotecas digitales[109, 110], las cuales enfocan sus funciones de clasificación y consulta basados en servicios Web, la diferencia radica en que utilizan algoritmos de RI para obtener la información de las bases de datos bibliográficas y por intermedio del servicio Web exponer la funcionalidad de las mismas. Lo que se logra es una respuesta a los usuarios con niveles de precisión y relevancia mayor que con un buscador normal.

Por otro lado también se ha encontrado un proyecto propuesto por Sotolongo[111], el cual propone el uso de un repositorio lexicográfico como WordNET[82], con el fin de comparar la información semántica contenida en el WSDL-S con los términos del repositorio y así poder jerarquizar y publicar los Servicios Web en el buscador de servicios para un descubrimiento más acertado a los requerimientos de los desarrolladores de aplicaciones distribuidas.

Otra aplicación de los Servicios Web y la RI la podemos encontrar en la búsqueda de música. El trabajo de Zadel[112] presenta una aplicación que utiliza las bases de datos de Google y de Amazon.com para generar grupos (cluster) de artistas relacionados basados en metadatos. Los resultados preliminares muestran que a través de un pre-procesamiento adecuado de los resultados devueltos por los servicios Web, es posible mejorar la experiencia del usuario en cuanto a la búsqueda de temas relacionados con la música.

Sin embargo, aparte de las ventajas de los servicios Web, no todo es bueno, también es necesario saber que su uso puede aumentar los problemas de tráfico en la Red y los tiempos de latencia pueden aumentar para dar al usuario una respuesta en un tiempo razonable tal como lo plantea Chourmouziadis[113] en su estudio.

De todo lo descrito anteriormente en este apartado, podemos tomar como elementos importantes hacia el presente proyecto lo siguiente:

- Los servicios Web pueden implementar técnicas de Recuperación de la Información y de Web Semántica internas a su negocio, con el fin de hacer un procesamiento adecuado del conocimiento a retornar a las peticiones de las aflicciones que los usen.
- Hay que tener cuidado al momento de implementar una arquitectura basada en Servicios Web Semánticos para la Recuperación de la Información, ya que los mismos podrían aumentar los tiempos de respuesta a los usuarios.
- Hasta ahora no se ha encontrado un estudio que enfoque sus esfuerzos a la construcción de Servicios Web Semánticos soportados en Ontology Learning, como soporte para recuperar datos e información de cualquier recurso Web, con la finalidad de mejorar las búsquedas de información de los usuarios.

1.8.7 Vistas Semánticas

Las vistas semánticas es un concepto propuesto por Fernandez[114] en su tesis de doctorado. Lo define como: "*una estructura de información formado por una o varias interpretaciones que proporcionan diferentes visiones de acuerdo a diferentes ontologías de un mismo contenido*". Así podemos obtener diferentes interpretaciones de una misma página Web, así una página Web se asocia a diferentes vistas semánticas que pueden ser utilizadas por los buscadores semánticos dependiendo del contexto del usuario o el objetivo de búsqueda.

Uno de los elementos importantes de esta implementación, es que permite convertir las páginas HTML en páginas Semánticas, las cuales hacen referencias mediante URIs a las diferentes vistas semánticas, pero sin interferir con las operaciones normales de los mismos sitios Web de donde se recupera la información.

Finalmente el proyecto propone una herramienta software llamada **sw2sws,** que permite transformar un sitio Web en un sitio Web Semántico mediante un sistema de cooperación de usuarios activos. El proceso se realiza mediante tres pasos: Identificación Ontológica, Extracción y Análisis e Interpretación, que es dónde se crean las vistas semánticas.

Este concepto de vistas semántica puede ser un elemento importante para reutilizar en el presente proyecto, ya que podemos utilizar las herramientas y el proceso definido para crear las vistas semánticas de los recursos Web y finalmente intentar convertir las posibilidades de información con una fachada de servicios Web Semánticos a ser utilizados por un buscador semántico. Sin embargo, como se verá más adelante en el modelo el concepto es utilizado para adaptar el índice al contexto de uso, intercambiado la ontología de dominio a utilizar, más no el proceso en sí mismo definido por Fernández.

1.9 Indexación semántica en la IoT

El aumento exponencial de los dispositivos conectados a Internet, plantea desafíos en recuperación y procesamiento de datos, debido a que los datos deben ser tratados en tiempo real y las redes son distribuidas, las gran tamaño en la cantidades de datos son difíciles de tratar y usualmente hay que utiliza herramientas para análisis de grandes cantidades de datos.

Una arquitectura escalable para consultar en la IoT con palabras claves es la indexación semántica, los datos semánticos pueden ser los documentos RDF están almacenados como bases de conocimiento de forma estructurada para consultas por palabras clave.

La integración de la web sensor con la web semántica, se puede lograr por medio de un esquema de codificación de los tripletes RDF, que permite la integración de conjuntos de datos heterogéneos que utiliza las normas (SOS). El razonamiento proporciona significado a los datos del sensor, anotando conceptos significativos que se puedan manejar con un motor de inferencia y entrega la capacidad de inferir recursos de sensores a una aplicación que no esté diseñada para este fin, también presta el servicio de *semántica compartida* evitando a las aplicaciones hacer sus propias traducciones de los datos de sensores. Por medio de ontologías se puede obtener interpretaciones propias y se puede beneficiar de una *semántica común de datos de sensores* lo que conduce a la interoperabilidad de la IoT[46].

El rastreo, indexación y recuperación de datos de cosas, lugares y personas es extremadamente difícil debido al gran tamaño del IoT, el problema de *escalabilidad* de búsqueda efectiva e indexación es un gran desafío con el fin de lograr que los motores de búsqueda puedan localizar los datos y servicios para el uso de otras aplicaciones.

Otra técnica que se puede utilizar es el (Latent Semantic Indexing – LSI) [115], el cual es un sistema de publicación de datos del sensor que sigue los principios de datos vinculados (Linked Data), sus elementos son el enriquecimiento semántico, la publicación de componentes y el almacenamiento de los datos de sensores haciendo uso de base de datos relacionales. El acceso a redes y bases de información de sensores se acceden a través de protocolos estándar e interfaces de aplicación. La salida del sistema es un conjunto de descripciones semánticas de sensores y mediciones en tiempo real, es decir se presentan como una descripción estándar en RDF, se utilizan los principios (Linked Open Data - LOD) esta quiere decir que son accesibles en la web. La interacción con el usuario propone palabras relevantes y el sistema le entrega una lista de los conceptos de los repositorios de sensores en línea, y partir de estos conceptos semánticos, realiza mapeo de datos y metadatos estructurados sensor, se apoya de una ontología estándar para mediación de diferentes descripciones de los sensores.

1.9.1 Normas y Estándares semánticos en el IoT

Las aplicaciones iniciales debían ser diseñadas conociendo perfectamente el sistema de sensores y su entorno, para solucionar esta dificultad se desarrollaron estándares y normas para soportar la integración y reutilización de los datos de sensores heterogéneos, por ejemplo las normas IEEE 1451 para transductores, las normas ANSI N42 para la radiación, adicionalmente métodos para acceder a los datos del sensor con paginas HTTP el cual se usó para el sistema GSN, y estándares sintácticos XDS para integración de datos sensor usado en el sistema Xively y Sensorbase, también se desarrollaron estándares semánticos como (Open Geospatial Consortium - OGC) y metadatos basados en RDF con ontologías OWL que representan el conocimiento de dominio, también tenemos protocolos para el intercambio de datos de sensores como él (Extended Environments Markup Languaje – EEML), usados para implementar conexiones entre sensores por medio de servicios web, también se propuso RDFa para anotar conceptos ontológicos y propiedades de (Sensor Web Enablement - SWE) utilizando Xlink para gestión y apoyo de funcionalidades semánticas.

Un aspecto importante propuesto, es añadir semántica al flujo de datos sensor para hacer consultas, se hicieron descripciones usando Prolog y representaciones de flujo con OWL, por ultimo llegamos a la descripción de consultas usando metadatos de flujo que utiliza aplicaciones para publicar las descripciones de datos de sensores como (Linked Stream Middleware - LSM) otro ejemplo es Sersormasher que posee un motor mashups visual para la publicación de datos de sensores parecida a linked data en la web semántica, otro ejemplo es Senseweb que se basa en metadatos sensor para seleccionar las fuentes más apropiadas de sensores, y un último ejemplo es el Sensorgrid4env que utiliza anotaciones semánticas para apoyar las interacciones entre sensores heterogéneos, cabe mencionar SPARQLstream el cual hace énfasis en la integración y no en la consulta ni en la escalabilidad de los datos sensor.

1.9.2 Estructuras de indexación en el IoT

El método de "Indexación etiqueta" se creó para dar una mejor solución a los problemas de almacenamiento e indexación, la estructura básica de almacenamiento cuenta con tres capas base y algunos mecanismos para acelerar el proceso de indexación, en la primer capa de almacenamiento encontramos la Tabla Principal de usuarios que contiene la información del usuario, la segunda capa almacena todos los Objetos que pertenecen a un usuario en particular y almacena todas las propiedades de los objetos. Hay tres maneras de indexación que se benefician de la estructura de almacenamiento; Primera: la indexación de

Interfaz Visual se utiliza para construir un diagrama 3D de objetos que es un mapa de objetos; Segunda: la indexación Etiqueta Orientada a Objetos que encuentra los objetos por medio de etiquetas Orientada a Objetos o Etiquetas Categoría; Tercera: indexación de Palabras Clave[116].

1.9.3 Modelos estadísticos de Objetos en el IoT

Las consultas en SensorNet están basadas en el paradigma de consultas declarativas sobre los datos de los sensores, los datos de los sensores no representan exactamente información del mundo real sino se debe aplicar un proceso para representar la realidad física por medio de la *Modelización Estadística,* necesaria para complementar las lecturas de sensores y así poder entregar respuestas más significativas a los usuarios, estos modelos también benefician los costos de adquisición de datos en la red y gasto de energía. El modelo se crea con algoritmos de optimización y un polinomio heurístico de tiempo para ver qué solución se adapta mejor a la práctica. El desarrollo de estas consultas permite a los programadores realizar búsquedas de datos de sensores sin preocuparse por la programación de los nodos de sensores. El trabajo de un buscador de bases de datos es responder correctamente a una consulta pero los sensores generan un conjunto continuo de datos que corresponde a una propiedad física o fenómeno que se produce en el tiempo y en un espacio al que se le denomina (dinamismo en los datos), con datos continuos los datos relevantes serian infinitos, por tal motivo es necesario tomar muestras discretas de información con determinados puntos en el espacio y tiempo, también se pueden crear modelos basado en los datos históricos con el uso de algoritmos convencionales.

Motor para procesamiento de consulta declarativa que utiliza el Modelo Probabilístico para establecer el estado de la red de sensores inalámbrico, desarrolla un protocolo de red llamado BBQ en base a un modelo específico de tiempo multi-variable Gaussiano. El modelo se utiliza para estimar la lectura de los sensores en un determinado periodo tiempo siendo esta la respuesta a la consulta[117].

ANEXO 2. EVOLUCIÓN CONCEPTUAL DE LA WOT

Este apartado se presenta la conceptualización, como una evolución de conceptos relacionados que llevan a la construcción de la Web semántica de objetos, estableciendo así el entorno y elementos en los cuales se desarrollará la investigación, relacionando a su vez investigaciones previas e importantes en ese aspecto.

El presente proyecto aporta a la interoperabilidad semántica en la IoT, enfocándose en la interacción entre los objetos de la WoT, para a partir de esta interacción, permitir construir otros modelos de interoperabilidad semántica con aplicaciones y personas. La interoperabilidad es un concepto que en general busca intercambio de información y conocimiento entre sistemas, a través de las tecnologías de la información y la comunicación existentes, en diferentes niveles: técnica, sintáctica, semántica y organizacional [1, 2].

Los objetos tienen unas características especiales como: diferentes tipos de datos [3] o señales, capacidad de almacenamiento, capacidad de procesamiento, comunicación, autonomía energética, localización, origen, estado, utilización y una capacidad de identificación única [4]. Para esto es necesario iniciar con una taxonomía de las características y propiedades de los objetos actuales, con el fin de servir como base para su conceptualización. Así el trabajo de Mathew, et al. [5], permite caracterizar los objetos basado en tres dimensiones: comunicación, procesamiento y almacenamiento, además de una identidad la cual permite establecer una jerarquización de los mismos. Esta clasificación será utilizada en el modelo conceptual a desarrollar.

Una vez que se ha establecido las características de los objetos, con el fin de aprovechar su información, es necesario definir un mecanismo que permita etiquetar los mismos [6], para posteriormente ser consultados y actualizados, esto se hace a través de metadatos. La mayoría de proyectos que proponen incorporación de técnicas de RI y de Web semántica [7-12], en general, proponen utilizar tecnologías basadas en XML y estándares de metadatos. El proyecto actual utilizará estas mismas anotaciones semánticas para almacenar la información contextual de los objetos.

Para gestionar los datos de los objetos, podemos encontrar cuatro tendencias de implementaciones de interoperabilidad en la WoT:

1. Existen varios trabajos que proponen la creación de middlewares [12-24], que realizan las abstracciones necesarias para conectar los diferentes tipos de objetos a la Web, independientemente de su tecnología y capacidades,

permitiendo a las aplicaciones de la Web consumir los datos de los objetos, pero con poca expresividad semántica, ya que están más orientados a la interoperabilidad sintáctica, sin embargo, han aparecido middlewares semánticos [10, 25, 26] que han incorporado técnicas semánticas para ofrecer un mejor soporte al conocimiento de los objetos. Estos desarrollos en su mayoría se enfocan a dominios particulares de desarrollo y pocos facilitan sus API de forma abierta para el desarrollo por parte de terceros, entre los más importantes están los proyectos middleware: LinkSmart [10], Hydra [27], Socrades [17] y Ubiroad [25].

2. Existen en el Web varios servidores mediadores de objetos como: SensorMap y Xively Cloud Servises (antes Cosm y PachuBe), que han surgido como nuevos servicios Web que pueden ser usados por diferentes tipos de usuarios para: conectar, compartir, interactuar e implementar aplicaciones con los datos de objetos. Estos servidores han implementado arquitecturas, como: REST - Representational Estate Tranfer y SOA - Simple Object Access Protocol, para asegurar la escalabilidad de la WoT y tecnologías como: MQTT - Message Queuing Telemetry Transport, que implementan protocolos de comunicación ligeros con los dispositivos, también exponen la información de datos y metadatos utilizando formatos estandarizados como: JSON - JavaScript Object Notation, XML - eXtensible Markup Language, CVS - Comma Separated Values, XDR - eXternal Data Representation. A diferencia de los middleware la gran mayoría ofrecen APIs de desarrollo abiertos, algunos se enfocan a dominios específicos y otros de propósito general. Adicionalmente tienen un enfoque de una sola base de datos centralizada para manejar los objetos y no distribuida como en los middleware.
3. Por cuenta de las empresas de productos electrónicos de entretenimiento y hogar se ha trabajado en la incorporación de procesadores y sistemas de almacenamiento con conexión a internet, que son capaces de correr un sistemas operativos ligeros e implementar tecnologías como: “Device Profile for Web Services” - DPWS o Digital Living Network Alliance – DNLA, ambos implementaciones de la tecnología Universal “Plug and Play” – UpnP, para la gestión, descubrimiento y control multimedia. También ofrecen los APIs de desarrollo de manera abierta con el fin de generar aplicaciones para sus dispositivos.
4. Cloud Computing: Esta área de trabajo se espera que permita jugar un papel importante en el desarrollo de la IoT ya que su diseño está más orientado al manejo del “*Big Data*”, un área de trabajo en la IoT que propone nuevos retos para manejar la gran cantidad de datos que se genera a partir de los sensores. Entre las principales plataformas existentes tenemos a Sensor-Cloud [28], la cual usa SensorML para describir los metadatos de los sensores físicos, empaquetando los mismos en una representación digital que los usuarios pueden combinar. SenaaS es otro servidor que apunta a lo mismo ofreciendo los sensores como servicios.

Aún existen diferencias en que tecnologías son mejores para la interoperabilidad y como utilizarlas adecuadamente. En general, los autores apuntan a modelos que utilicen la arquitectura REST y SOA, además de protocolos ligeros para la telemetría. Cada opción tiene aún retos por resolver y la transparencia en la misma está lejos de ser una realidad para un usuario no técnico de la Web. El presente proyecto tendrá en cuenta las mejores prácticas en estos sentidos con el fin de incorporarlos al modelo conceptual.

Teniendo ya el acceso a los datos de los sensores y anotados semánticamente, es el momento de construir conocimiento alrededor de los objetos, para esto se ha propuesto la creación de repositorios de conocimiento basados en ontologías [6, 10, 25, 29-32], implementando diferentes estrategias para almacenar y procesar el conocimiento proveniente de los sensores de la IoT. Las soluciones van desde la propuesta de modelos ontológicos como el Semantic Model Driven Architecture (SeMDA) [27], hasta el desarrollo de ontologías para el manejo de los eventos, de los dispositivos, de estimación y de contexto. Para este proyecto, se destaca el trabajo presentado por el grupo denominado “W3C Semantic Sensor Network Incubator group (the SSN-XG)” [33] los cuales han creado una ontología llamada *SSN ontology sensors* (http://purl.oclc.org/NET/ssnx/ssn), la cual permite describir las capacidades de los sensores, el proceso de medición y las observaciones resultantes en una red de sensores semánticos. Esta ontología se puede alinear con otras ontologías con el fin de realizar un trabajo conjunto en la semántica de los objetos. Esta ontología ha reutilizado los demás estándares existentes en la representación de sensores, tales como SensorML y O&M del Open Geospatial Con-sortium’s (OGC) [34], y de servicios Web como SensorWeb Enablement (SWE) standards [35]. . El presente proyecto reutilizará y generará su propia propuesta de ontologías a ser incorporadas en el modelo conceptual.

Finalmente, las últimas propuestas de servicios Web exploran lo que se ha empezado a llamar el “Social Web of Things” [36, 37], como una estrategia para compartir información entre objetos y humanos, acerca de los datos de los sensores, además del contexto de operación y otros servicios de alto nivel. Sin embargo la mayoría de los trabajos hasta ahora están explorando como comentar los objetos a las redes sociales existentes [38-48] y algunos ya empiezan a proponer arquitecturas [36]. Estos últimos temas son más complejos, ya que tienen que involucrar además del conocimiento, las capacidades de raciocinio en los objetos para permitirles tomar decisiones o comportamientos.

El presente proyecto se sitúa en esta última capa de interacción social entre los objetos. Sin embargo, se orienta explícitamente a lograr comprender a través de un modelo conceptual que integre las tecnologías necesarias para implementar interacción entre objetos, bajo la premisa de que cuando los objetos adopten un comportamiento colaborativo entre ellos, los servicios a los humanos pueden enriquecerse más fácilmente.

ANEXO 3. ANÁLISIS DE TRABAJOS PREVIOS EN EL DESARROLLO DE LA SWOT

Adicionalmente, se presentan los puntos concretos (números de 1-3) en los cuales se hace una síntesis y brechas existentes por cada área. Finalmente se puede evidenciar el aporte de la presente investigación. En la **¡Error! No se encuentra el origen de la referencia.** se explica cada punto temporal (PT):

PT*	Investigaciones Previas y Detalle
Arquitecturas IoT	
A	BROLL, G., RUKZIO, E., & PAOLUCCI, M. Perci: Pervasive service interaction with the internet of things. En: IEEE Internet Computing (2009). [50] • ***Aporte***: presenta un caso de estudio particular de la IoT y es una propuesta arquitectónica en la que interactúan los dispositivos móviles con etiquetas Radio Frecuency IDentification - RFID, para generar nuevos servicios en la Web. Proponen una nueva especificación llamada *Service User Interface Annotation (SUIA)* basada en Ontologías, con el fin de integrar servicios Web con *Physical Mobile Interaction (PMI)*. Para el enriquecimiento de los servicios Web provee unos *widgets con ontologías,* que permiten una mejor interacción con el usuario. • ***Análisis***: Los resultados obtenidos en la interacción con los usuarios humanos, permiten que la presente investigación tenga en cuenta los posibles problemas y estrategias al momento de desarrollar interfaces para móviles. La principal falencia es que su arquitectura está planeada para un entorno muy particular de la IoT como lo es los dispositivos móviles con objetos etiquetados con RFID, dejando por fuera otros dispositivos como sensores. Los servicios Web semánticos se utilizan como opción de interoperabilidad, en este caso agregando especificaciones. Las ontologías usadas no se presentan en detalle y se utilizan en el nivel de los servicios. SCIOSCIA, F., & RUTA, M. Building a Semantic Web of Things: issues and perspectives in information compression. En: Semantic Computing, 2009. ICSC'09. IEEE International Conference (2009). [51] • ***Aporte***: centra su atención en la *anotación semántica* que se debe realizar para la Web semántica de los objetos. Pone de manifiesto, los principales problemas para generar esta Web y que se relacionan a las características de los mismos objetos: muy poco espacio de almacenamiento, poca o nula capacidad de procesamiento y enlaces inalámbricos de bajo rendimiento. Propone como alternativa de solución un enfoque de compresión de datos y su consulta eficiente, adicionalmente, propone una arquitectura de IoT cimentada en un modelo que utiliza una *base de conocimiento ubicua - uKB*, con acceso a ontologías y Servicios Web Semánticos. en la IoT. • ***Análisis***: las propuestas de anotación semántica pueden ser utilizadas como base para su reutilización. Este primer artículo hace más énfasis en el modelo propuesto para la compresión de la información que en la uKB y las ontologías, Aún debe ser probado en más escenarios, para establecer si efectivamente no crean un fuerte impacto en el rendimiento para el uso de información proveniente de los objetos.
B	RUTA, M., NOIA, T. D., SCIASCIO, E. D., SCIOSCIA, F., & TINELLI, E. A ubiquitous knowledge-based system to enable RFID object discovery in smart environments. En: n Proceedings of the 2nd International Workshop on RFID Technology - Concepts, Applications, Challenges (2010); 87 - 100. [7] • ***Aporte***: Es una continuación del trabajo anterior, pero en este hace más énfasis en la arquitectura y sobre todo en la *base de conocimiento ubicua – uKB*, para la cual explica elementos importantes de implementación y un caso de estudio particular. • ***Análisis***: Es importante la forma en que almacena anotaciones en las mismas etiquetas RFID, lo único es que necesita cierto tipo de etiquetas que serían más costosa para un despliegue de la aplicación. Los procesos de descubrimiento implican el desarrollo de

PT*	Investigaciones Previas y Detalle
	algoritmos en la red EPCGlobal que es privada. El trabajo se enfoca en un tipo particular de objetos etiquetados con RFID dejando de lado los demás tipos existentes. GUINARD, D., TRIFA, V., & WILDE, E. A resource oriented architecture for the Web of things. En: Internet of Things (IOT), 2010 (2010). [52] GUINARD, D., TRIFA, V., KARNOUSKOS, S., SPIESS, P., & SAVIO, D. Interacting with the SOA-Based Internet of Things: Discovery, Query, Selection, and On-Demand Provisioning of Web Services. En: IEEE Transactions on Services Computing (2010); 223-235. 1939-1374. [53] • ***Aporte***: Plantea una arquitectura en la cual puede hacer uso de estándares como Device Product Web Services - DPWS y los principios REST para acceder a la capa de objetos y conectarlos adecuadamente a la capa de servicios, a través de los "physical mashups". En un segundo artículo lo complementan con la arquitectura orientada a recursos – ROA y computación en la nube, como un medio para lograr la interoperabilidad en esta arquitectura. • ***Análisis***: Los "*physical mashups*", sugieren muy buenas ideas para la composición de los servicios de información. En las evaluaciones preliminares de su arquitectura establecen cómo los sistemas fuertemente acoplados, son benéficos para los requisitos del alto rendimiento en tiempo real en la comunicación, esto implica que la idea de encapsular las soluciones a través de un servicio Web, podría ser una buena solución en cuanto al reúso de estas tecnologías con poco detrimento del rendimiento en la RI en la IoT. En la última publicación, adicionalmente incorporan la computación en la nube. El principal vacío que tiene es que no plantean una solución a la interacción entre objetos. ROALTER, L., KRANZ, M., & MÖLLER, A. A middleware for intelligent environments and the internet of things. En: Ubiquitous Intelligence and Computing (2010); 14. [24] • ***Aporte***: Establece los retos de los middlewares en la IoT y sus características. Finalmente escoge uno de la robótica llamado Robot Operating System - ROS, para luego crear una oficina inteligente a partir de sensores / actuadores que le permitían interactuar a través de ROS. Establece de manera práctica unos servicios sencillos de contexto que se publican en una red social a manera de microbloging. El modelo arquitectónico utilizado es el de publicador / suscriptor y demuestra como un middleware en robótica puede ser adaptado para la IoT. • ***Análisis***: Explica en detalle el montaje de la infraestructura de sensores desplegados para la oficina inteligente, algo que puede servir de guía para montajes similares en el presente proyecto. Los servicios de contexto son fácilmente replicables, sin embargo deja el vacío de cómo filtrar y fusionar adecuadamente la información proveniente de los diferentes dispositivos desplegados. Los datos publicados en la red social deben ser interpretados por los humanos adecuadamente.
C	KOSMATOS, E., & TSELIKAS, N. Integrating RFIDs and Smart Objects into a Unified Internet of Things Architecture: 2011. Series. [54] • ***Aporte***: Integra los objetos RFID y los Smart Objects en una sola arquitectura basada en SOA a manera de middleware, para ello definen una arquitectura orientada por la semántica (Semantic Model Driven Architecture – MDA). Establecen conceptos interesantes de un internet que llamaron "Social Internet of Things", proponiendo una capa de abstracción de objetos, otra de administración de servicios y otra de composición de servicios, las cuales están entre los objetos y las aplicaciones. • ***Análisis***: El aporte es conceptual y no está evaluada en términos de escalabilidad y adaptabilidad. Habla de un modelo semántico, pero no presenta un ejemplo claro de su creación y uso, solo habla de los elementos a tener en cuenta y las tecnologías que se podrían usar. Adicionalmente, no tiene en cuenta el tipo de interacciones humano/objeto que se presentaría en la red social propuesta.

PT*	Investigaciones Previas y Detalle
	GUINARD, D., FLOERKEMEIER, C., & SARMA, S. Cloud computing, rest and mashups to simplify rfid application development and deployment. En: Proceedings of the 2nd International Workshop on the Web of Things (WoT 2011) (2011). [55] • ***Aporte***: Integra los patrones y modelos Web como REST con las iniciativas de servicios IoT Cloud. Para esto, desplegaron un caso de estudio en la red EPCglobal con dispositivos móviles, estableciendo problemas de implementación en dicha red y como se podían resolver a través de tecnologías disponibles como JAVA e implementaciones en la red, orientándola a consumo de recursos en tiempo real. Presentan un editor de mashup que facilita el despliegue de servicios desde los sensores RFID. • ***Análisis***: El trabajo permite valorar más la opción del uso de REST como alternativa para la creación y presentación de servicios de la IoT. Adicionalmente, el uso de servidores Cloud son una buena opción para el acceso a los datos de los dispositivos, con unas adaptaciones sencillas basadas en prototipos que utilizan lenguajes de desarrollo Web. Así el proyecto puede optar por utilizar estos servicios e implementaciones de prototipos para probar dinámicamente conceptos en el desarrollo de los servicios. Finalmente los mashups son un concepto muy poderoso para componer servicios más complejos y de manera transparente, los cuales se pueden tener en cuenta en el modelo conceptual. HERNÁNDEZ-MUÑOZ, J., VERCHER, J., & MUÑOZ, L. Smart Cities at the Forefront of the Future Internet. En: The Future Internet (2011). [56] • ***Aporte***: Esta propuesta incursiona en el concepto de la Smart Cities. Establecen los problemas actuales de los sistemas de información aislados de las ciudades y como éstos deben evolucionar para crear una plataforma escalable de servicios de información, teniendo en cuenta los nuevos paradigmas del internet del futuro: la IoT y la Internet de Servicios – IoS. Proponen una arquitectura que tiene en cuenta éstos conceptos como capas, junto con lo que denominaron la Internet de Personas, para esto, definen una red de sensores ubicuos “Ubiquitous Sensor Network” – USN, basándose en principios de plataformas abiertas, federadas y confiables. Establecen tres funcionalidades base: abstracción de las comunicaciones urbanas, modelos unificados de información urbana y desarrollo de servicios urbanos abiertos. Finalmente introduce el proyecto SmartSantander [57], como una instalación a manera de ciudad con sensores, con la finalidad se servir como repositorio de experimentación (“testbed”) de modelos, aplicaciones, metodologías y proyectos hacia las ciudades inteligentes. • ***Análisis***: La propuesta arquitectónica está a nivel conceptual y aún no se han desarrollado casos de estudio de la misma, sin embargo la parte importante es la identificación de la necesidad de modelos de descubrimiento, almacenamiento y gestión de información provistas por los sensores, con el fin de unificar información y servicios, algo en lo que podría aportar la presente propuesta con su modelo conceptual. Otro elemento importante es la identificación de instalaciones con más de 20.000 sensores en cuatro ciudades que ha creado el proyecto SmartSantander, con la posibilidad de utilizar sus testbeds para experimentar el modelo.
D	VEGA-BARBAS, M., CASADO-MANSILLA, D., VALERO, M. A., LOPEZ-DE-IPINA, D., BRAVO, J., & FLOREZ, F. (2012, 4-6 July 2012). Smart Spaces and Smart Objects Interoperability Architecture (S3OiA). Paper presented at the Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2012 Sixth International Conference on. [11, 58]. Aunque el primer aporte lo hacen en un primer artículo en 2011 [11], se decide colocarlo en 2012, ya que este artículo incluye el primero y está más detallado. • ***Aporte***: Esta propuesta plantea una arquitectura orientada a servicios dinámicos, muy completa para la interoperabilidad semántica en el IoT. Aplica conceptos de una arquitectura centrada en el usuario, quien asume un rol de supervisor e interactúa con

PT*	Investigaciones Previas y Detalle
	interfaces de usuario multidispositivo. Establece cinco módulos bien definidos que permiten capacidades de abstracción de los objetos y las aplicaciones a diferentes niveles. Un elemento importante es que enfoca los esfuerzos a desarrollar aplicaciones hacia los objetos locales a través de los *gateways residenciales*, como elemento de abstracción física que puede permitir conectar objetos con pocas capacidades de procesamiento y almacenamiento. por otro lado utilizan conceptos de espacios de tupla y espacios de triplete, refiriéndose al uso de semántica RDF para encontrar *Smart spaces*. Todo soportado en un repositorio de servicios y un modelo publicador / suscriptor. • ***Análisis***: Hasta ahora no se han presentado implementaciones reales de dicha propuesta, pero el trabajo aporta una idea interesante para el uso de objetos con características limitadas en los gateways residenciales, el cual se puede retomar para el presente trabajo. El concepto de smart spaces es interesante pero no suficientemente desarrollado como para incorporarlo directamente al modelo conceptual. ♦ HEESUK, S., SEUNGWOOK, H., & DONGMAN, L. (2012, 22-25 Aug. 2012). Contextual Information Provision on Augmented Reality with IoT-Based Semantic Communication. Paper presented at the Ubiquitous Virtual Reality (ISUVR), 2012 International Symposium on. [59] • ***Aporte***: Aporta una arquitectura en la IoT centrando su atención a la información del contexto en ambientes ubicuos y la inclusión de la realidad aumentada "Augmented Reality" – AR, como elemento semántico importante para la interacción en la IoT. Establece las limitaciones de las interfaces actuales de AR, para ser utilizadas en los datos provistos por los objetos, ya que dependen de los contextos y aún más importante proponen una plataforma semántica para sintetizar la información de los sensores u presentarlas en la interfaces AR. Define cinco tecnologías elementales para el desarrollo de interfaces de usuario AR entre ellas: comunicación semántica, confiabilidad, red centrada en el contexto, razonamiento del contexto y auto-evolución. • ***Análisis***: El aporte conceptual es muy bueno para tenerlo en cuenta en el modelo a desarrollar, ya que para lograr presentar información contextual en interfaces AR, es necesario que exista una comunicación semántica entre objetos, el cual es el primer elemento de la arquitectura propuesta y lo relacionan a un middleware que desarrollaron para tal fin, sin embargo, el middleware permite abstraer los objetos e información semántica relacionada a los mismos, pero no les permite la interoperabilidad completa, ya que se necesita establecer los intereses de los objetos, ponerlos de acuerdo en un objetivo y establecer mecanismos que disparen las acciones de su colaboración. Esto debe basarse en raciocinios del contexto. Esto último lo establecen como un vacío por resolver y es interesante que la propuesta conceptual pueda aportar en este punto. Las tecnologías que define el artículo deben ser analizadas con detalle en el modelo a presentar, ya que todas se relacionan con la interacción entre objetos. Falta desarrollar más los conceptos de contexto y como lograr cada una de las tecnologías a desarrollar quedando todos como fuentes de aporte. ♦ ZHANG, C., CHENG, C., & JI, Y. Architecture design for social Web of things. En: Proceedings of the 1st International Workshop on Context Discovery and Data Mining, Beijing, China.(2012) [36] • ***Aporte***: La arquitectura que propone centra su atención a la incorporación de una red social de Objetos. Establece la información del contexto, como un elemento fundamental para el desarrollo de esta plataforma, definen su arquitectura teniendo en cuenta que los usuarios están interesados inicialmente en la entidades (cosas, lugares, personas) y posteriormente por las cualidades o estados de esas cosas (vacío, libre, sentado, caminando, etc.) y no en sensores individuales y sus datos de salida. Por tal

PT*	Investigaciones Previas y Detalle
	razón proponen la “Social Web of Things” como una plataforma para compartir información entre máquinas y personas, para esto plantea una plataforma RESTful con servicios de redes sociales utilizando ontologías y lenguaje natural. Como tecnologías clave define: la identificación (con cuatro tipos: recurso, contexto, dispositivo e información social), el modelado del contexto (ontología de contexto), Ontología en lenguaje natural (cambiar el lenguaje máquina a lenguaje natural). Finalmente presentan un ejemplo a manera de prototipo funcional llamado MagicHome. • ***Análisis***: Todos los conceptos presentados son un insumo muy importante a ser analizados y su posible incorporación al modelo conceptual. El caso de estudio es más un escenario de motivación y por ende queda por probar aún en escenarios reales la arquitectura propuesta. Otro elemento importante que se puede incorporar al modelo conceptual, es la idea de crear una ontología que traduzca el lenguaje de los objetos con el lenguaje natural, para poder implementar interoperabilidad semántica entre máquinas y personas, aunque inicialmente hay que manejarlo con cuidado porque estos procesos de razonamiento pueden ir en el detrimento de la interacción en tiempo real, la cual es necesaria para un modelo semántico en la IoT.
1	Síntesis y brechas por resolver en las arquitecturas de trabajo de la IoT • ***Síntesis***: Con respecto a las arquitecturas de trabajo del IoT hay un consenso en la creación de capas bien definidas, en general son: la capa de objetos, capa de representación semántica, capa de servicios y capa de aplicaciones, utilizando diferentes estilos arquitectónicos como: SOA (Service-Oriented Architecture) y RESTful (Representational State Transfer), también tecnologías como: bases de conocimiento ubicuo, servicios Cloud y compresión de datos. En la capa de objetos se proponen protocolos ligeros de comunicación como: 6LOWpan y MQTT y en la capa de servicios se habla de arquitecturas REST. Por otro lado, plantean el uso de ontologías de contextos, para aumentar la semántica de los objetos, los servicios Web como herramienta para la interoperabilidad. Con respecto a interfaces se habla de tecnologías Web como: HTML5, AJAX y Mashups físicos, pero ya se está incursionando en la integración de la realidad aumentada como elemento importante de interacción. Finalmente, introducen el concepto de redes sociales de objetos para la interacción con los humanos. • Brechas: o En el desarrollo de la SWoT, se han identificado tecnologías de comunicación apropiadas hacia los objetos y otras, hacia las personas, pero aún no existe un consenso en cual privilegiar o si hacer un híbrido de las dos. Se incursionan soluciones con protocolos y ontologías mediadoras. o Existe una falencia de un correcto desarrollo de fusión de datos y desarrollo de interfaces de los servicios de los sensores con las personas, de tal forma que permitan proveer más información (contextual) que los simples flujos de datos medidos por los objetos. Aún se deben proveer modelos conceptuales (ontológicos) que permitan vislumbrar soluciones adecuadas para este fin. o las propuestas arquitectónicas en la interacción de objetos son escazas y exploratorias, dejando la posibilidad de proponer modelos formales para este tipo de interacción. Los trabajos de Heesuk, et al. [59] y Zhang, et al. [36] establecen conceptos muy importantes para el desarrollo de redes sociales de objetos y se convierten en los principales referentes para el modelo conceptual de interacción entre objetos que desea proponer el presente trabajo. • ***Posibles Aportes***: El modelo conceptual proveerá una ontología que puede utilizarse para mediar semánticamente entre la información de los objetos y otros objetos. Así se aporta a la fusión de datos que pueden ser utilizados en servicios a las personas. Adicionalmente, las propuestas de redes sociales de objetos pueden utilizar este modelo para generar el modelo de interacción en dicha capa.

PT*	Investigaciones Previas y Detalle
Integración Semántica a la WoT	
E	♦ RÖMER, K., OSTERMAIER, B., MATTERN, F., FAHRMAIR, M., & KELLERER, W. Real-Time Search for Real-World Entities: A Survey. En: (2010). [8] OSTERMAIER, B., RÖMER, K., MATTERN, F., FAHRMAIR, M., & KELLERER, W. A real-time search engine for the Web of things. En: (2010). [60] • ***Aporte***: presenta un motor de búsqueda de información de tiempo real para la IoT llamado Dyser, el cual debe superar retos diferentes a los de un buscador tradicional, especialmente el tamaño de la información a procesar y en tiempo real. Establecen dos aproximaciones para crear un motor de búsqueda en la IoT: el enfoque de inserción (*push*) y el enfoque de extracción (*pull*). Propone un algoritmo de clasificación de sensores (*Sensor Ranking*), con el fin de establecer, dada un consulta, cual es la probabilidad de que cierto sensor produzca la salida buscada, para esto proponen tres modelos de predicción de datos: (aggregated prediction model (APM), single period prediction model (SPPM), multi-period prediction model (MPPM)) y proponen una arquitectura para éste tipo de motores, estableciendo nuevos conceptos como páginas de sensor, páginas de entidad en HTML y anotadas con microformatos junto con el trabajo simultáneo de un indexador y un resolver para manejar diferentes tiempos de respuesta. Los modelos son implementados en "sensor gateways", es decir en sensores con capacidades de procesamiento y almacenamiento. • ***Análisis***: los conceptos y problemas encontrados para la creación del motor de búsqueda sirven de precedentes para este estudio en el sentido de capturar la información semántica de los objetos la cual será comunicada a otros objetos. El motor de búsqueda y el modelo establecido, fue probado midiendo la sobrecarga y los tiempos de latencia, indicadores importantes que se podrían incorporar al modelo conceptual, sin embargo es necesario que se haga medición de relevancia con los indicadores tradicionales de RI, con el fin de establecer si cumple con los requerimientos de búsqueda de los usuarios. Aunque no plantean conceptos en interacción de objetos, los elementos semánticos introducidos permiten vislumbrar las ventajas de separar las consultas de los usuarios en dos tiempos: unas que se pueden resolver sin conexión a los objetos y otras en tiempo real, teniendo que decidir cuál realizar o hacer una combinación de las mismas. La interacción de objetos debe ser disparada por un evento interno o externo, uno de esos eventos podría ser una consulta de usuario. GUINARD, D., FISCHER, M., & TRIFA, V. (2010, March 29 2010-April 2 2010). Sharing using social networks in a composable Web of Things. Paper presented at the Pervasive Computing and Communications Workshops (PERCOM Workshops), 2010 8th IEEE International Conference. [48] • ***Aporte***: Utiliza las redes sociales como Twitter para compartir los objetos con otras personas, utilizando patrones Web como REST. La principal contribución es una plataforma llamada "Social Access Controller", la cual presenta tres funcionalidades: proxy para autenticar y compartir objetos inteligentes (smart things) entre los usuarios, control de acceso basado en la red social existente, publicita el objeto compartido en la misma red social. Implementa módulos orientados a recursos en los Smart gateways para que actúen como proxy de los sensores sin capacidad de procesamiento y así poder incluirlos como parte de un smart thing. La plataforma desarrollada permite que los usuarios (propietarios o amigos) interactúen con el objeto en dos vías y no sólo obtengan sus datos. • ***Análisis***: La solución presentada se enfoca a compartir e interactuar objetos con personas, más que interacción entre objetos. Sin embargo, la utilización de las redes sociales existentes como plataformas para compartir los objetos, resuelven el problema de autenticación y confianza que heredan los objetos a través de los usuarios de la red

PT*	Investigaciones Previas y Detalle
	social, aunque deben profundizar más los aspectos de seguridad de datos. Los autores, plantean como reto, la necesidad de encontrar un marco de trabajo que permita describir integralmente los objetos inteligentes, para así poder construir un ecosistema de objetos adecuadamente interrelacionados, reto al cual podría aportar el modelo conceptual que se desarrollará en la presente investigación. ZHEXUAN, S., CÁRDENAS, A. A., & MASUOKA, R. (2010, Nov. 29 2010-Dec. 1 2010). Semantic middleware for the Internet of Things. Paper presented at the Internet of Things (IOT), 2010. [61] • ***Aporte***: Implementa un middleware semántico que permite integrar las tecnologías de sensores existentes que se conectan a los dispositivos, funcionando como una capa intermediaria para aplicaciones que requieran el uso de dispositivos con tecnologías heterogéneas. Los principales objetivos del middleware son: crear interoperabilidad entre dispositivos y la información, manejo del contexto para las aplicaciones (descubrimiento de servicios y composición) y finalmente información significativa para los usuarios que desean construir servicios con funcionalidades de seguridad y privacidad. Presentan unas interfaces intuitivas para el uso de los servicios Web semánticos que logran la interoperabilidad deseada. El enriquecimiento semántico de los dispositivos la realizan con OWL-S y recopilando la información necesaria para ejecutar sus servicios, se crea un esquema por cada dispositivo y se almacena en un repositorio. Posteriormente definieron una extensión al OWL-S para ejecutarlos directamente sin tener que hacer un OWSDL. Para el aprovechamiento de los usuarios desarrollaron una interfaz en que conjuntamente componen los servicios. • ***Análisis***: Aunque los autores plantean como ventaja la idea de no crear un único estándar, sino de interpretar los existentes y unificarlos en una misma capa de servicios, existe una desventaja y es que la escalabilidad de nuevos dispositivos con nuevos estándares implicaría rescribir el código del middleware con el fin de incorporarlos, así mismo sucedería cuando el estándar cambie o evolucione. La principal dificultad visualizada es como sacar la semántica de los sensores, ya que dependiendo del protocolo de comunicación se deben hacer implementaciones a medida, lo cual conlleva un esfuerzo muy grande. Se puede aprovechar la identificación de problemas encontrados para crear servicios Web semánticos de los sensores, pero no se puede reutilizar lo que se propone, porque no presentan implementaciones reales de lo que se supone hace el middleware, por lo tanto el aporte es más teórico. Para el presente proyecto, se ha pensado en la posibilidad de usar los servicios Web semánticos como elementos de interoperabilidad semántica, sin embargo, no nos centramos en la interoperabilidad de red, dejando este trabajo los middleware, nos enfocamos en la interoperabilidad de los objetos ya digitalizados o Web, a los cuales se les aplicarán los métodos semánticos con el fin de obtener información suficiente para su interacción, esto trae la dificultad de depender del middleware a utilizar como capa de abstracción y por ello la primera etapa del modelo tiene que hacer un proceso adecuado de selección del mismo. Finalmente no se encontró información de la interfaces intuitivas.
F	AMARAL, L. A., HESSEL, F. P., BEZERRA, E. A., CORRÊA, J. C., LONGHI, O. B., & DIAS, T. F. O. eCloudRFID – A mobile software framework architecture for pervasive RFID-based applications. En: Journal of Network and Computer Applications, 34(3) (2011); 972-979. 1084-8045. [62] • ***Aporte***: Presenta un framework adaptativo para desarrollar aplicaciones RFID en dispositivos móviles (Mobile RFID services) llamado "*CloudRFID Framework*". Permite la integración de aplicaciones embebidas con instancias de la red EPC global. • ***Análisis***: Este proyecto se relaciona a la presente investigación en el sentido que las etiquetas RFID son sensores que hacen parte del al IoT. Teniendo en cuenta que el

PT*	Investigaciones Previas y Detalle
	modelo conceptual debe en lo posible abarcar la gran mayoría de sensores, es necesario establecer las tecnologías disponibles para su integración a la solución. El trabajo analizado pone de manifiesto los problemas que se tienen al momento de enlazar la información de los RFID con los eventos de negocio, proponiendo una solución para los mismos. Sin embargo, las soluciones están enfocadas sobre la reutilización de redes privadas como la EPC global y software específico, lo cual pone impedimentos para incorporarlo a una solución abierta. En este punto se mantiene en nuestro proyecto el enfoque de utilizar las etiquetas RFID como parte de un dispositivo más inteligente y que normalmente se programaría en los smart gateways. KOSTELNÍK, P., SARNOVSKÝ, M., & FURDÍK, K. The Semantic Middleware for Networked Embedded Systems Applied in the Internet of Things and Services Domain. En: Scalable Computing: Practice and Experience, 12(3) (2011). 1895-1767. [10] • ***Aporte***: presenta una plataforma de middleware, denominada LinkSmart enfocada a ofrecer servicios desde la IoT. La plataforma fue diseñada para apoyar la interoperabilidad y la integración de varios dispositivos externos, sensores, y los servicios en los sistemas empresariales principales. LinkSmart reutiliza los resultados de dos proyectos HYDRA [27] y EBBITS [63], utilizando conceptos como: la arquitectura modular, la arquitectura orientada a servicios, las redes peer-to-peer, las tecnologías de los servicios Web semánticos, la construcción de ontologías subyacentes y los mecanismos de inferencia basadas en el conocimiento. Por otro lado, hace una propuesta arquitectónica de la orquestación de servicios, manejo de eventos complejos, modelado de procesos de negocio y procesamiento de flujo de trabajo. Presenta un modelo semántico conducido por la arquitectura (*Semantic Model Driven Architecture - SeMDA*), que fue diseñado para facilitar el desarrollo de aplicaciones y para promover la interoperabilidad semántica para los servicios en línea y los dispositivos de tipo inalámbrico o por cable. Adicionalmente presenta el concepto de "*dispositivos semánticos*", ya que utilizan una ontología para su representación y anotación. Se presenta una arquitectura que integra todos los conceptos y algunos aspectos de implementación, actualmente están trabajando en dos casos de estudio uno en fabricación de automóviles para reducir el consumo de energía y otro en la industria de producción de alimentos para soportar el ciclo de trazabilidad en todo el proceso de fabricación. • ***Análisis***: A pesar de la gran cantidad de información y documentos desarrollados alrededor del proyecto sobre el cual basa todo su desarrollo, aún no se conocen los resultados de estos casos de estudio, que permitan establecer la viabilidad real de la solución propuesta. Por otro lado, no se habla de la interacción semántica entre objetos, sino de aplicaciones que pueden utilizar los dispositivos semánticos. Con respecto al modelo ontológico propuesto y los resultados de sus investigaciones acerca de la apropiación semántica de los dispositivos se pueden en cuenta al momento de crear el modelo conceptual propuesto en este proyecto. ♦ HACHEM, S., TEIXEIRA, T., & ISSARNY, V. Ontologies for the Internet of Things. En: Proceedings of the 8th Middleware Doctoral Symposium (2011). [6] TEIXEIRA, T., HACHEM, S., ISSARNY, V., & GEORGANTAS, N. (2011, October 26-28, 2011). Service oriented middleware for the internet of things: A perspective. Paper presented at the Towards a Service-Based Internet 4th European Conference, ServiceWave 2011, Poznan, Poland. [15] • ***Aporte***: El artículo establece de primera mano los retos que debe sortear la IoT para poder utilizar sus potenciales servicios (escalabilidad, heterogeneidad profunda, topología desconocida, disponibilidad desconocida de punto de datos, datos incompletos o no precisos y resolución de conflictos). Proponen un middleware

PT*	Investigaciones Previas y Detalle
	orientado a Servicios con cinco módulos (descubrimiento, expansión, mapeo, optimización y ejecución), soportando sus funcionalidades en tres ontologías (ontología de dominio, ontología de estimación, ontología de dispositivo) que son tratadas como una base de conocimiento ("*Knowledge Base*" - KB). • ***Análisis***: La propuesta conceptual es muy interesante, en el sentido que los módulos propuestos se comportan como grandes procesos de indexación semántica, un enfoque que el presente proyecto pretende abordar como estrategia de enriquecimiento semántico de los objetos que van a interactuar. Las ideas de una ontología de estimación y de un proceso de descubrimiento pueden ser adicionadas a la idea original del presente proyecto, como elementos valiosos para incorporar. Lamentablemente, no se presentan casos de estudio o implementaciones y por ello no se puede evaluar la viabilidad de la propuesta
G	KOTIS, K., & KATASONOV, A. (2012, 4-6 July 2012). Semantic Interoperability on the Web of Things: The Semantic Smart Gateway Framework. Paper presented at the Complex, Intelligent and Software Intensive Systems (CISIS), 2012 Sixth International Conference. [31] • ***Aporte***: Proponen un marco de trabajo basado en el concepto de una "*Semantic Smart Gateway* - SSGF" para la SWoT. Establecen los retos y requerimientos que debe resolver para implementar el SSGF como: un registro semántico (Ontología IoT), aprendizaje de ontologías y alineación automática de ontologías. • ***Análisis***: La propuesta de trabajo presentada esta en la primera fase y en el sitio Web del proyecto se puede acceder a la ontología IoT con la que planean hacer el registro semántico. Los demás modelos están en planificación y aún deben implementarse para valorar correctamente su impacto. Sin embargo el plan es muy ambicioso y toca tecnologías que son muy difíciles de desarrollar o que están en sus inicios como el área del *Ontology Learning*. El aporte para el presente trabajo es la verificación de sus ontologías y como se podría alinear con el modelo conceptual que se está desarrollando. ♦ WEI, W., DE, S., TOENJES, R., REETZ, E., & MOESSNER, K. (2012, 25-27 June 2012). A Comprehensive Ontology for Knowledge Representation in the Internet of Things. Paper presented at the IEEE 11th International Conference onTrust, Security and Privacy in Computing and Communications (TrustCom-2012). [32] • ***Aporte***: Los autores presentan una ontología en la cual modelan los aspectos de la IoT con el fin de almacenar conocimiento en ella. Presentan unos principios de diseño de la misma y posteriormente definen varios módulos que pretenden abarque todo el conocimiento del área. Realizan alineaciones con la ontología DUL y la SSN-XG y los servicios con el estándar OWL-S. • ***Análisis***: Los principios de diseño presentados pueden tenerse en cuenta para la construcción de las ontologías de la presente propuesta, adicionalmente algunos de los módulos presentan conceptos interesantes que nos permiten abstraer algunas características de la IoT, por ejemplo la definición de los servicios y tipos de servicios utilizados. Otro elemento interesante es que reutilizan las ontologías más aceptadas en este momento sobre la IoT, lo cual garantiza un buen aporte a las mismas. Sin embargo la ontología no presenta aspectos relativos a la interacción semántica, por lo cual no se modela este conocimiento, algo que el presente proyecto tiene como objetivo.
H	RAZZAK, F. The role of semantic Web technologies in smart environments. 2013, porto.polito.it. [64] • ***Aporte***: Es una tesis doctoral en la cual centra su trabajo en la propuesta de un framework para un entorno domótico. Presenta los conceptos relacionados a la Web semántica y define el concepto de "*Smart Environments*" para el cual define una serie

PT*	Investigaciones Previas y Detalle
	de mecanismos para intercambiar datos enriquecidos semánticamente. Para el enriquecimiento define una ontología denominada DogOnt y una ontología como Gateway domótico basado en OSGi (Dog), como elementos fundamentales para realizar la interoperabilidad semántica entre los dispositivos domóticos. Adicionalmente, el trabajo de revisión realizada con respecto a los objetivos de los usuarios y su interacción con los dispositivos de su entorno es importante para identificar posibles servicios que pueden generar los espacios inteligentes. • ***Análisis***: El modelo ontológico presentado puede ser una buena fuente para el diseño de la ontología que se pretende crear en este trabajo. Aunque el foco del trabajo está orientado a un entorno inteligente domótico la interacción sigue estando orientada hacía entre objetos y usuarios, más que entre objetos inteligentes. ♦ PERERA, C., ZASLAVSKY, A., CHRISTEN, P., & GEORGAKOPOULOS, D. Context Aware Computing for The Internet of Things: A Survey. En: IEEE Communications Surveys & Tutorials (2013). [65] • ***Aporte***: Hace un estudio bastante detallado sobre el análisis del contexto para la Internet de Objetos. Aporta identificando los principales problemas para desarrollar la IoT y adicionalmente establece un estudio de middleware desarrollados que implementan características de contexto. Adicionalmente, afirman que son pocos los estudios que se han enfocado en conciencia y contexto para la IoT. Finalmente establece los principales retos que deben afrontar los sistemas que establecen trabajen con el contexto para la IoT. • ***Análisis***: Es un excelente referente para tenerlo en cuenta al momento de establecer los elementos de contexto que se podrían integrar en el desarrollo de la interacción entre objetos de la presente propuesta. AGGARWAL, C., & ABDELZAHER, T. (2013). Social Sensing Managing and Mining Sensor Data.[38] • ***Aporte***: Define con detalle el concepto de “Social Sensing” en el cual describe los usos que les están dando los usuarios a los objetos y sus aplicaciones. Establece los objetivos de esta área de trabajo y los retos que aún enfrenta. Presenta también las aplicaciones que recolectan datos en tiempo real de los sensores para generar servicios (“Crowd-Sourcing”). • ***Análisis***: Es un trabajo muy interesante, el cual se puede tener en cuenta para establecer los mecanismos actuales de interacción entre usuarios – aplicaciones y objetos y cómo podemos aportar a los mismos desde la visión de interacción entre objetos.
2	Síntesis y brechas por resolver en Integración Semántica en la IoT • ***Síntesis***: Las principales conclusiones son: o El punto de partida de todos los proyectos son las anotaciones semánticas y un mecanismo adecuado para almacenarlos. Las anotaciones son hechas con formatos de intercambio de conocimiento bien conocidos en la Web como XML, RDF y OWL. o El estudio del contexto es uno de los principales temas que deben ser representados en los modelos semánticos, ya que se ha encontrado que los datos de los sensores cobran importancia cuando se relacionan a su entorno y a los objetivos de los usuarios. o Se ha encontrado una tendencia a que se creen gateways inteligentes, como puntos intermedios en la red semántica, que son capaces de recopilar la información de los objetos con limitaciones de procesamiento y comunicación. o Los middleware se han convertido en una herramienta fundamental para el

PT*	Investigaciones Previas y Detalle
	desarrollo de la Web semántica de los objetos, ya que éstos permiten resolver los problemas de heterogeneidad profunda en los últimos bordes de la red, adicionalmente presentan una imagen digital del objetos a las aplicaciones y usuarios Web con el fin de poder interactuar con los mismos. ○ Se ha encontrado como consenso el uso de ontologías adaptadas a las características de los sensores y la información del contexto como estructuras capaces de almacenar y permitir razonar sobre la información semántica de la WoT. Los modelos son diversos y aún no hay un consenso en que modelo aporta más información. ○ La búsqueda Web se ha relacionado más con el descubrimiento de servicios de los dispositivos de la IoT. Sin embargo, es necesario explorar la posibilidad de indexación de toda esta información con el fin de proveer servicios de información semántica en la WoT. • ***Brechas***: Existen varias retos por resolver, pero las principales brechas identificadas y relacionadas con el proyecto son: ○ Se ha incrementado la necesidad de crear modelos formales semánticos, con el fin de encontrar una forma adecuada de aprovechar la información y el conocimiento inferido de los objetos. Existen varias propuestas pero muy pocas en resolver en tema particular de interacción semántica entre objetos. ○ El modelamiento del contexto y de la conciencia es un campo que tienen aún mucho por proponer. Se tienen identificados elementos de contexto, pero hasta ahora se está estudiando cómo implementarlo. ○ La búsqueda y descubrimiento en la WoT se ha centrado en el enriquecimiento semántico y muy poco en la indexación semántica, éste enfoque podría ser una buen punto de partida para implementar esta funcionalidad. • ***Posibles Aportes***: El modelo conceptual proveerá diferentes niveles de abstracción de los objetos, los cuales dependerán de los niveles de interrelación que se produzcan entre los mismos, para esto se adoptaran conceptos de contexto y conciencia entre los objetos con el fin de darles elementos para su interacción. La solución plantea un enfoca de indexación semántica con el fin de realizar los procesos de búsqueda y descubrimiento entre los objetos.
Comportamiento Inteligente y Social de la WoT	
I	♦ PINTUS, A., CARBONI, D., & PIRAS, A. The anatomy of a large scale social Web for internet enabled objects. En: Proceedings of the Second International Workshop on Web of Things, San Francisco, California.(2011) [42] PINTUS, A., CARBONI, D., & PIRAS, A. Paraimpu: a platform for a social Web of things. En: Proceedings of the 21st international conference companion on World Wide Web, Lyon, France.(2012). [39] • ***Aporte***: Proponen una plataforma Web en la que permiten registrar usuarios con sus objetos (sensores y actuadores). Reutilizan servidores Web que soportan una gran cantidad elevada de carga de transacciones, motores de base de datos noSQL y un proceso de manejo de eventos y mensajes a través de RESTful. Un aporte interesante es como conectan la salida de un sensor a un actuador a través de su plataforma, utilizando el concepto de “filter and mapping”. • ***Análisis***: La plataforma presentada permite un proceso básico de interacción entre objetos, la cual es adaptar la salida de un sensor a la entrada de un actuador, mediado por el usuario, quien es el que define la forma de conexión, incluso en actuadores de dispositivos embebidos debe adaptar un código fuente para su correcta utilización. La propuesta es interesante en la forma en que identifica los problemas iniciales que tiene la interacción entre objetos, pero no presenta una solución para un proceso de interacción más inteligente o al menos autónoma, ya que exige que los usuarios

PT*	Investigaciones Previas y Detalle
	adapten los diferentes dispositivos. Las interacciones sociales están medidas en la posibilidad de compartir los datos de los dispositivos conectados con los amigos en redes sociales existentes. Este proyecto explora el mismo objeto de estudio que el presente proyecto, sin embargo el presente busca ampliar la capacidad de interacción con el fin de superar los servicios básicos de una única conexión de entradas y salidas. El concepto introducido de "filter and mapping" es muy interesante y podría ser reutilizado por el presente proyecto.
J	♦ BIAMINO, G. A Semantic Model for Socially Aware Objects. En: Advances in Internet of Things (2012). [66] • ***Aporte***: Plantea un entorno en el cual interactúan los objetos y las personas, para lo cual estable en concepto de "Social Aware Objects", propone un modelo de conocimiento de contexto-conciencia social y razonamiento basado en ontologías para modelar el contexto, adicionalmente un modelo de usuario para implementarlo en las redes sociales. Define tres ontologías para el modelo de conocimiento (ontologías de contexto, ontología de objetivos sociales y ontologías de objetos), más un modelo básico de preferencias de usuario. • ***Análisis***: La propuesta se percibe como el primer acercamiento a la definición de un modelo conceptual para el manejo de lo que llamaron una comunidad inteligente "Smart comunity", definida por el autor como una comunidad de objetos físicos y personas compartiendo servicios ubicuos de manera social e inteligente. Aún falta mucho por definir y resolver, ya que establece los propósitos y características generales de las ontologías usadas y no las presenta para su correcto análisis, además la evaluación es realizada teniendo en cuenta sólo la interacción entre las ontologías desarrolladas y las personas en una red social existente, con el propósito de evaluar la capacidad de detección de contextos de la misma ontología, es decir, no hay interacción entre objetos y personas, mucho menos entre los mismos objetos. Por otro lado las preferencias de usuario exponen que son adquiridas con las API de la red social, pero en el experimento lo hacen a través de un formulario dirigido que llenan los participantes, estos participantes fueron seleccionados por los evaluadores y no en un modelo aleatorio, afectando esto la reducción de la validez de los resultados. Sin embargo, las propuestas realizadas con respecto al enfoque de construcción de las ontologías y la forma incipiente de indicadores de medición de detección de contextos son valiosos y permiten orientar la construcción de las ontologías de esta propuesta. ♦ YAO, L. (2012). A Propagation Model for Integrating Web of Things and Social Networks. In G. Pallis, M. Jmaiel, A. Charfi, S. Graupner, Y. Karabulut, S. Guinea, F. Rosenberg, Q. Sheng, C. Pautasso & S. Mokhtar (Eds.), *Service-Oriented Computing - ICSOC 2011 Workshops* (Vol. 7221, pp. 233-238): Springer Berlin Heidelberg.[47] • ***Aporte***: Define un modelo de recomendación de amigos en las redes sociales, teniendo en cuenta las interacciones entre las personas y las cosas de la WoT. Establecen tres tipos de interacciones: "*people to people*", "*people to things*", y "*things to things*". Para relacionar lo anterior definen un modelo gráfico de variable latente, es decir un modelo de conexiones entre las interacciones definidas y trabajando la técnica de filtrado colaborativo (interacciones pasadas entre usuarios y objetos de la vecindad), buscando que el modelo pueda sugerir que servicios de un objeto puede utilizar un usuario de la red social encontrando la similaridad de la interacción. Así, proponen un modelo matemático de cada uno de los tipos de interacciones definidas. El modelo matemático de personas y objetos trabaja con variables semánticas de las personas, de sus intereses y de los objetos, los cuales se calculan en un promedio ponderado de pesos de las similaridades estas variables. El modelo de la interacción persona con persona utilizan la formalización matemática "random walk" y la interacción entre objetos la realizan con un modelo de similaridades graficas no funcionales como la proximidad.

PT*	Investigaciones Previas y Detalle
	• ***Análisis***: Es uno de los pocos artículos en el cual que definen un modelo específico en la interacción entre objetos. Sin embargo, los mismos autores reconocen las falencias de su propuesta y está en su mismo enfoque, un modelo gráfico de variables latentes introduce el problema de la incapacidad para reconocer los parámetros comunes y las variables latentes de las relaciones integradas, es decir, en el modelo en un momento existen ciertas relaciones entre objetos - personas y a siguiente tiempo podrían ya no existir estas mismas relaciones, esto no es lo que generalmente sucede en la realidad. Adicionalmente, el enfoque es simulado y no se vislumbra como conectar este modelo en un entorno real, en el cual las variables que usan son dinámicas y pueden cambiar en muy poco tiempo como para trabajarlas como condiciones iniciales. Se puede rescatar el modelado matemático propuesto para definir las interacciones, las cuales se pueden integrar al presente trabajo. ♦ GUO, B., ZHANG, D., YU, Z., LIANG, Y., WANG, Z., & ZHOU, X. From the internet of things to embedded intelligence. En: World Wide Web (2012); 1-22. 1386-145X. [67] • ***Aporte***: Explora las tecnologías para incrustar inteligencia en los objetos de la IoT o W2T (Wisdom Web of Things). Plantea conceptos de conciencia en niveles de usuario, ambiente y social. Establece varios retos por resolver como: medir la participación humana, colección de datos y representación, estándares para la comunicación y representación del conocimiento, manejo de la Incertidumbre, aprendizaje de complejidad y selección de modelos y privacidad y economía. Finalmente presenta una arquitectura orientada a la W2T, con un ejemplo práctico como caso de estudio. • ***Análisis***: Los conceptos introducidos acerca de la inteligencia embebida y como los diferentes tipos de objetos pueden interactuar con los humanos para capturar información sobre su contexto son interesantes y se podrían tener en cuenta al momento de establecer el modelo conceptual. Sin embargo, el estudio no plantea una interacción entre objetos, ya que su perspectiva está orientada por la interacción con los humanos explícitamente. En nuestro caso la interacción con los humanos podría ser una fuente de información semántica, pero no la única para propiciar una interacción con otro objeto. Nuestra propuesta podría complementarse con los planteamientos del proyecto analizado. Adicionalmente, entre los retos clave se plantea la creación de modelo conceptuales que representen adecuadamente los datos que los sensores miden, algo que van en los posibles aporte de este proyecto.
3	Síntesis y brechas por resolver en Comportamiento Inteligente y Social de la IoT • ***Síntesis***: Los conceptos de redes sociales de objetos están definidos más a nivel conceptual, que implementados y se enfocan principalmente a interacción con redes sociales actuales. Muy pocos estudios abordan las características que debe tener la interacción M2M en una red social de objetos que se colaboran. Sólo se pudo encontrar un sólo trabajo que presenta desde el punto de vista conceptual una implementación básica la creación de la Web Inteligente de los Objetos. • ***Brechas***: Los retos por resolver en este ítem son de alta complejidad: o En cuando a las redes sociales de objetos aún no se logra una forma transparente y efectiva que permita compartir los objetos e interactuar con ellos. o Los modelos presentados trabajan ontologías y modelos matemáticos. Es necesario analizar bien sus aportes con el fin de obtener el mejor provecho de dichas visiones. o El razonamiento y la inteligencia de los objetos aun esta por desarrollar, ya que hasta ahora se está buscando como almacenar el conocimiento adecuadamente para su correcto uso. o La W2T necesita del aporte de muchos trabajos y de diferentes disciplinas con el fin de encontrar un modelo escalable. Adicionalmente, es necesario el avance de las capacidades de procesamiento con el fin de embeber más objetos con

PT*	Investigaciones Previas y Detalle
	inteligencia • ***Posibles Aportes***: La posibilidad de tener un modelo conceptual de interacción semántica entre los objetos, puede servir como base para crear entornos más inteligentes y apoyar procesos de descubrimiento y composición de servicios a otras aplicaciones que se centren más en la inteligencia de las máquinas y sus interacciones con los humanos.
•	

Figura 14. Análisis de Trabajos Previos en el desarrollo de la SWoT

* Punto Temporal con respecto a la Figura 2, ♦ Es uno de los trabajos más importantes para la investigación

Finalmente, todos los estudios anteriores han permitido reafirmar las principales tecnologías que se usarán en la IoT (anotación semántica, algoritmos de RI, ontologías, estándares, protocolos y uso servicios Web semánticos como elemento de interoperabilidad) para la construcción de entornos inteligentes del IoT. Sin embargo, no se ha profundizado alrededor de establecer la manera de *implementar cooperación inteligente entre objetos*, es decir mecanismos eficientes de interacción que permita ser tomado como base para la construcción de dichos entornos inteligentes de objetos.

ANEXO 4. MEJORES PRÁCTICAS PARA CONSTRUIR ESCENARIOS DE INTERACCIÓN SEMÁNTICA EN LA WoT

Con el fin de incorporar los conceptos y mejores prácticas determinadas en el campo de la interacción semántica de la WoT se presenta un enfoque de desarrollo de aplicaciones para la SWoT, el cual no pretende ser una camisa de fuerza, pero si un referente metodológico que ayudaría a soportar este tipo de aplicaciones para la WoT. Para ahondar en el tema, remitirse al Anexo 2.

1. Clasificación de Sensores IoT: Es necesario categorizar los objetos de acuerdo a sus capacidades con el fin de definir cuáles de ellos pueden interactuar y en que niveles. Para esto se recomienda usar la propuesta de Mathew, et al. [18], la cual clasifica los dispositivos con respecto a sus capacidades y así se puede definir marcos de trabajo para cada tipo de objeto de la WoT.

2. Descripción de los Sensores IoT: Teniendo en cuenta que existen diversas formas de describir y estructurar la información de los dispositivos que contienen los sensores de la IoT, se recomienda adoptar una de las especificaciones estándar existentes y que están siendo utilizadas por varios proyectos. Así se recomienda:

- **Formato de descripción (Metadatos y Datos)**: Para definir las características del sensor, plataforma, sistema y despliegue, restricciones de operación, datos medidos y proceso de observación, se recomienda el uso de la ontología SSN-XG [48]. Además de estar en formato OWL que es interoperable, permite procesos de razonamiento sobre las descripciones y relaciones del dispositivo. Esta especificación ha retomado el trabajo de otros estándares y los ha incorporado a su solución por lo cual es más completa que las otras existentes.
- **Formato para la Comunicación entre dispositivos**: Ya que el sistema se enfoca a la interacción entre dispositivos, se recomienda seguir las especificaciones definidas por el Open Group Standard con el “Open Data Format” (O-DF) [62] y el “Open Messaging Interface” (O-MI) [63], los cuales permiten estandarizar los mecanismos de comunicación entre dispositivos de la IoT sin importar la tecnología de comunicación (MQTT, XMPP o CoAP), ya que puede proveer soporte a arquitecturas REST o SOA.
- **Formato para la Comunicación con Middlewares y Servidores IoT**: Ya que la mayoría de los dispositivos de la IoT se encuentran alojados en estos servidores mediadores, es importante verificar que utilicen formatos compartible y estándar para transmitir los datos y metadatos, tales como: XML, JSON y CSV.

3. Selección de Middleware: Es necesario evaluar las plataformas middleware más adecuadas para obtener datos de los dispositivos de la IoT con relación a los requisitos de la aplicación. En general se relacionan las siguientes opciones:

Crear un Middleware Propio: No se recomienda crear un middleware propio, ya que las soluciones existentes llevan más de 10 años implementando este servicio y mejorando su soporte y eficiencia, si es necesario crear uno por razones de

requerimientos del proyecto, se aconseja utilizar uno de código abierto, soporte de API a múltiples lenguajes de programación y soporte de comunidad de desarrollo, en lo posible que implemente arquitecturas REST o SOA.

- **Usar un Middleware existente**: Si se usa una plataforma existente hay que evaluar bien las restricciones para acceso gratuito y los costos para acceso completo. Algunas restricciones son en ventana de datos históricos, ancho de banda, número de consultas y resultados, etc. que pueden afectar de otra forma la aplicación a desarrollar.

- **No usar un middleware**: El hecho de no utilizar un middleware lleva a que la solución será hecha a la medida y en este caso se debe crear software que capture la información de los dispositivos teniendo en cuenta su hardware y software, desarrollando los manejadores ("drivers") para su conexión, gestión y comunicación.

4. Definir una arquitectura guiada por un modelo semántico: Se sugiere que definir un modelo semántico para la WoT, de tal forma que se convierta en una fuente de información importante para la búsqueda, descubrimiento y la interacción entre los objetos. A continuación se hacen unas sugerencias de lo que se debe tener en cuenta para la definición del modelo:

- **Principios de diseño**: El modelo debe regirse por principios, se sugieren:

  - **Principio de acoplamiento débil**: definir funciones que sean autocontenidas y que no dependan de los resultados de otras.

  - **Principio de parametrización**: Las funciones deben poder ser reconfiguradas y los servicios con entradas bien definidas.

  - **Principio de reutilización**: una vez que se han creado nuevos servicios se debe permitir su uso por parte de cualquier usuario, sin tener que volver a crear los servicios desde cero.

  - **Principio de contexto** – consciencia dinámica: La anotación semántica de los dispositivos debe estar relacionada al contexto actual y debe definirse o reutilizarse en el momento que sea requerido por el usuario.

- **Definición de Conceptos**: El modelo debe definir conceptos y tecnologías clave a usar y como lo usa, así se sugiere definir: Objetos Inteligentes, Recursos y Entidades de Interés, Espacios Inteligentes, Servicios Dinámicos y Usuarios entre otros que dependerán del modelo específico.

- **Funcionalidades:** El modelo debe implementar varias funcionalidades, de las cuales dependiendo de los requisitos se decide implementar unas o todas. Se presenta una lista de las principales funcionalidades que podría definir:

  - **Conectividad**: los mecanismos que permiten abstraer y agregar un nuevo dispositivo al sistema, para gestionar sus funcionalidades mediante una especificación uniforme y notificación a los demás dispositivos. Para dispositivos con bajas capacidades se sugiere utilizar un Gateway semántico [23, 46, 69, 76].
  - **Descubrimiento**: Funcionalidad que permite encontrar dispositivos y sus servicios de acuerdo a los requerimientos de usuarios.
  - **Gestión de Eventos**: Esta funcionalidad permite definir condiciones a los flujos de los sensores para activar disparadores que se conocen como eventos y generar una interacción con los usuarios o con otros objetos.
  - **Búsqueda por Contexto**: Permite a los usuarios realizar consultas para obtener dispositivos que cumplan con los requerimientos en un campo de conocimiento en el que posiblemente puedan utilizarse sensores. Esta consulta normalmente esta soportada en el uso de ontologías de dominio específico y relacionado a los sensores de la IoT.
  - **Búsqueda por Capacidades**: Son búsqueda de sensores que cumplen requisitos de acuerdo a sus especificaciones técnicas y de servicio. Aquí la ontología SSN es importante para esta funcionalidad.
  - **Gestión del Contexto**: Son los mecanismos que se utilizarán para detectar, almacenar y comunicar el contexto, ya sea primario o secundario, entre los usuarios y los mismos sensores.
  - **Mecanismos de comunicación**: especifica los protocolos, módulos y especificaciones que se utilizarán para comunicar los dispositivos entre sí y con los usuarios.
  - **Consulta en lenguaje natural**: La posibilidad de que los usuarios realicen sus consultas o interactúen con los dispositivos en su propio lenguaje hace que la solución sea más aceptada y escalable. Se deben utilizar técnicas de ontologías de lenguaje natural y de recuperación de la información.
  - **Fusión y presentación de datos**: esta funcionalidad define mecanismos para poder combinar datos de diferentes sensores y presentar su información de manera formal y entendible a los usuarios.
  - **Actuación y control**: son mecanismos que permiten ejecutar acciones sobre el entorno a través de los actuadores de los dispositivos, ya sea por parte de otros dispositivos o por parte de los usuarios.
  - **Autenticación y autorización**: son los mecanismos que permiten confirmar la identidad de los propietarios de los dispositivos y de los permisos de consulta o actuación que tienen los usuarios o de otros dispositivos.

- **Gestión de Servicios**: se define la forma en que se presentarán los servicios a los usuarios y los otros objetos. Debe implementar mecanismos de tolerancia a fallos, recambio de objetos y manteamiento de la sesión para operaciones largas.
- Interfaces de usuario: Define las características de las interfaces a utilizar para la interacción y acceso a la información por parte de los usuarios, estas pueden ser a través de Mashups, HTML5 o técnicas de realidad aumentada, entre otras.

5. Base de conocimiento: Para la construcción de la base de conocimiento es necesario decidir la estructura de almacenamiento y gestión del conocimiento a utilizar. Entre las estrategias se puede definir por utilizar:

- **Ontologías**: aquí hay que preguntarse ¿cuáles ontologías construir?, entre ellas: descripción de sensores, contexto, modelo de usuario, estimación y traducción. Adicionalmente establecer la reutilización de las estandarizadas existentes con la SSN.
- **Técnicas semánticas**: Se pueden utilizar las representaciones RDF con LOD para crear una WoT enlazada y semántica.
- **Combinación de ellas**: Se puede decidir por combinar ambas estrategias haciendo más relevante la solución.

6. Modelo semántico de Interacción de objetos: Se debe definir un modelo enfocado en la interacción semántica entre objetos y con los usuarios. Este modelo debe estar en coordinación con la arquitectura y modelo planteado en las etapas anteriores. Principalmente se debe tener en cuenta lo siguiente: ¿qué arquitectura adoptar?, ¿cómo comunicar los objetos?, ¿qué activa las interacciones?, ¿qué estrategia de composición de servicios de interacción utilizar?, ¿cuál es el modelo matemático adecuado para representar las interacciones?, ¿cómo definir una aproximación a un comportamiento inteligente? y finalmente ¿qué mecanismos de comportamiento social se utilizará?

7. Plataformas y Aplicaciones: Define en qué y cómo exponer los servicios del modelo implementado, se debe pensar en una plataforma Web que exponga inicialmente servicios de información semántica de los objetos y en ella se debe construir las funcionalidades que permitan registrar objetos y las personas o aplicaciones que podrán consumir dichos servicios. Se deben desarrollar las interfaces necesarias para consumir los servicios.

8. Pruebas y Validaciones: Con respecto a este ítem la mayoría de pruebas se realizan teniendo en cuenta tiempos de respuesta y eficiencia en el uso de recursos. Sin embargo es necesario empezar a validar la satisfacción de los usuarios y otras medidas de interacción que deben ser creadas para tal fin.

ANEXO 5. IMPLEMENTACIÓN DEL BUSCADOR Y EL ÍNDICE SEMÁNTICO

Este capítulo realiza la implementación de un índice semántico sobre la WoT, para el caso del dominio de contaminación ambiental. Se realiza una instancia del método definido en el capítulo anterior, mostrando en cada fase y actividad las decisiones tomadas para la construcción de dicho índice. Finalmente, se crea el prototipo de buscador semántico para WoT que utiliza el índice.

5.1 Fase I – Actividad 1 – Tarea 1: Visión del Sistema

5.1.1 Introducción

El presente documento tiene como finalidad mostrar la visión del índice semántico creado y un análisis de las funcionalidades implementadas en el buscador semántico. Los productos principales fueron dos: El índice semántico que accede al servidor de Objetos Xively y una aplicación Web que permite encontrar objetos dada una consulta en el dominio de contaminación medioambiental en ciudades específicas de Colombia.

5.1.2 Definición del problema

Para el caso particular se desea crear un buscador semántico sobre el servidor IoT llamado Xively, que fue seleccionado de acuerdo a unas características particulares. Este servidor tiene conectados millones de objetos con sus correspondientes datos y metadatos. La idea principal es que cualquier persona que utiliza el buscador pueda encontrar fácilmente objetos que le permitan reutilizar los datos que pertenecen al dominio particular. El dominio seleccionado como el caso de estudio ha sido la información y contaminación medioambiental. La razón de la selección de dicho dominio es que es uno de los campos en los que más sensores se han desplegado en la IoT y de los cuales sería más fácil obtener mayor cantidad de datos. Adicionalmente es muy poco lo que se ha encontrado de aplicaciones de la WoT en el campo de la contaminación medioambiental.

Existen buscadores tradicionales como google, bing, altavista y en el mismo servidor IoT, en los cuales se pueden realizar búsquedas sobre los eventos medioambientales. Sin embargo, es necesario hacer navegación por varios minutos para poder elegir las fuentes de datos y en algunos casos se trata de páginas web que proveen la información y no de los sensores existentes y públicos de la WoT. Para el caso colombiano la entidad oficial que está encargada de implementar el sistema de gestión ambiental es el IDEAM (http://www.ideam.gov.co/jsp/index.jsf), esta entidad ha implementado en todo el país sistemas hardware con sensores para medir variables climatológicas y ambientales, además de construir sistemas de información geográfica y bases de datos para recoger las mediciones. Lamentablemente, los usuarios que deseen acceder a dicha información lo deben hacer a través de las aplicaciones Web de la

entidad y los datos para usuarios generales corresponden en su mayoría son históricos y no mediciones en tiempo real. Para éstos últimos es necesario firmar convenios o realizar pagos por dicha información.

Con la llegada de los servidores de objetos IoT, los cuales permiten a cualquier usuario de la Web crear y conectar objetos y sensores a la Web, se ofrece la posibilidad de utilizar esta información como posibles fuentes gratuitas para realizar búsquedas de información relevantes a los usuarios de la misma Web.

La idea principal es mejorar la búsqueda de objetos de la IoT con información semántica de contexto de su: localización, variables que mide, si está activo o no a través de un modelo de conocimiento mediado con una ontología de aplicación de contaminación medioambiental y otras ontologías existentes que optimizan el proceso como la SSN-XG y objeto semántico.

Para el caso particular el usuario desea buscar información de los sensores relacionados a variables de contaminación medioambiental en una región particular de Colombia. La aplicación presenta un Ranking de los sensores que cumplan con los criterios de búsqueda y despliega su información para conocimiento del usuario.

En la Tabla 30, se muestra los cuatro aspectos principales de la visión del sistema.

El problema de:	¿Cómo realizar consultas a la WoT (Servidor IoT) con el fin de mejorar el descubrimiento de sensores relacionados a variables específicas de la contaminación medioambiental?
Afecta:	Usuarios de la Web que desean obtener información de los dispositivos de la WoT relacionados a la información de contaminación medioambiental en una región.
El impacto es:	La creación de Índice medioambiental y del servicio de búsqueda pueden ayudar a: • Mejorar el tiempo de Descubrimiento y Búsqueda de los objetos de la IoT que se pueden utilizar en la contaminación medioambiental. • Agregar rápidamente los servicios de los objetos a aplicaciones.
Una solución viable sería:	Tener un servicio de búsqueda Web de sensores de la WoT que provea información contextualizada a su localización y capacidades de medición de variables de interés en el campo de contaminación medioambiental.

Tabla 30. Visión de BSIOT

5.1.3 Ubicación de los Productos

El producto “Visión del Sistema” que contiene un resumen de los requisitos básicos establecidos para sistema. Se encuentra en la siguiente ruta del CD de entrega:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 1\Actividad 1\Tarea 1\ Visión del Sistema

5.2 Fase I – Actividad 1: Tarea 2: Especificación de Requisitos Buscador Semántico – BSIoT

5.2.1 Introducción

En este apartado se presentan los requerimientos funcionales y no funcionales del sistema, también se establecen el conjunto de productos que se obtendrán en las fases posteriores.

5.2.2 Requerimientos Funcionales

La lista de requerimientos funcionales es la siguiente:

R1. **Registro**: Esta es una funcionalidad que le permite al usuario darse de alta en el sistema con el fin de poder recaudar la información de sus calificaciones de relevancia.
R2. **Iniciar Sesión**: Es una funcionalidad que le permite al usuario identificarse en el buscador con el fin de activar las funcionalidades reservadas para los usuarios.
R3. **Realizar Búsqueda**: Esta funcionalidad permite a los usuarios digitar una consulta en lenguaje natural con el fin de obtener resultados acerca de los objetos que cumplen con la consulta realizada.

5.2.3 Interfaces del SistemaInterfaces de Usuario

La interfaz debe proveer una zona para digitar la consulta, una zona para ejecutar la consulta, otra zona en los que se encuentren unos menús textuales de registro, inicio de sesión y finalización de la sesión.

5.2.5 Interfaces a Sistemas Externos

5.2.5.1 Interfaces de Software

Las interfaces están desarrolladas como un repositorio de servicios Web que podrían ser semánticos. En la salida de los sensores que coinciden con el criterio de búsqueda se les coloca un enlace a las funcionalidades de los servicios web desarrollados para consumir sus datos. En este punto, sin embargo, es de esperar que no todos los sensores tengan los servicios desarrollados, ya que por ahora se enfoca a un trabajo manual y no automatizado, por salirse de los alcances del presente proyecto. Los servicios web del índice se expondrán mediante un enlace a una página de servicios web disponibles para su uso con otras aplicaciones.

5.2.5.2 Interfaces Hardware

Sin interfaces de hardware.

5.2.5.3 Interfaces de Comunicaciones

Las interfaces de comunicación son abstraídas por el servidor IoT.

5.2.6 Reglas de Negocio (Business Rules)

Cómo el índice que se va a trabajar es en el área de la información medioambiental y la contaminación, se espera que el índice permita alimentar un buscador que soluciones las siguientes reglas de negocio:

R1. Consultas de variables medioambientales: Esta es una funcionalidad que le permite al usuario solicitar información de las variables medioambientales tales como: magnetismo, humedad, radiación solar, presión atmosférica, calidad del aire y viento en un lugar específico**.**

R2. Consultas sobre contaminación medioambiental: Es una funcionalidad que le permite al usuario solicitar información de los agentes contaminantes que son medidos a través de los sensores en un lugar específico.

5.2.7 Documentación

Se documenta la instanciación del método y del código fuente desarrollado. Finalmente se debe crear un manual de usuario de la herramienta.

5.2.8 Ubicación de los Productos

El producto “Especificación de Requisitos v2” los requerimientos funcionales y no funcionales del sistema. Se encuentra en la siguiente ruta del CD de entrega:
C: \CD del Proyecto \IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 1\Actividad 1\Tarea 2

5.3 Fase I – Actividad 1: Tarea 3: Arquitectura del Sistema BSIoT

5.3.1 Introducción

El presente documento resume la arquitectura definida para crear el índice semántico en la IoT. Del estado del arte se han definido servicios específicos que se van a utilizar para la creación del índice. Por ello la arquitectura aquí presentada define los servicios a utilizar. Sin embargo, el servidor de objetos IoT no se había seleccionado en su momento, por lo tanto se asumieron algunas características que debería tener dicho servidor y que se tuvieron en cuenta al momento de hacer el estudio de las características del servidor.

5.3.2 Objetivos de la Arquitectura y Filosofía

Debido a que esta arquitectura se deriva de la propuesta realizada en el método para crear índices semánticos en la IoT, los referentes de su filosofía son los mismos expuestos en esta parte. Se adaptan al desarrollo específico los objetivos así:

- ***Eficacia en la extracción***: Consiste en obtener la mayor cantidad de semántica posible de los objetos como consecuencia de esto se mejora la extracción. Los datos se obtendrán de los metadatos provistos por el Servidor IoT seleccionado. El servidor debe proveer dos tipos de información a indexar:
    - ***Información Estructurada***: Esta información viene en formatos XML o JSON y campos de captura de datos a manera de metadatos, los campos se llenan en el momento en que cada usuario crea su sensor en el servidor IoT; la información de los metadatos es relacionada directamente a los conceptos en las ontologías, así se asegura una alineación correcta de dichos datos para poder ser incluidos al momento de ser indexados. Para que sean tenidos en cuenta, se realiza una anotación semántica de dichos conceptos en el archivo después de eliminar las palabras vacías.
    - ***Información No estructurada***: Los formatos XML y JSON del servidor, proveen información no estructurada en palabras clave, descripciones y demás campos abiertos en los cuales el usuario incorpora información de acuerdo a sus percepciones. Son convertidos en una bolsa de palabras que se agregan a las anotaciones anteriores y posteriormente

es indexada semánticamente.

- ***Mecanismo eficiente de captura de datos***: El servicio del API del servidor IoT tiene ya una forma eficiente de captura y reutilización de datos, utilizan metadatos que están en XML u otro formato estándar para cada uno de los sensores que almacena. Estos datos son ofrecidos través del API que ofrece el servidor IoT para usuarios finales y aplicaciones.
- ***Correcto uso de estándares***: Se utilizará como principal ontología la SSN-XG para clasificar características de los sensores, con respecto a los conceptos medioambientales se ha alineado la ontología ENVO, la cual ha sido creadapor expertos internacionales. Por otro lado, el formato en los cuales se transmite la información es XML o JSON.
- ***Selección adecuada del Modelo Semántico***: Se refiere al correcto análisis de los datos obtenidos provenientes de los dispositivos, se estudia la representatividad (inferencia de conceptos) y adecuado almacenamiento en una estructura de información eficiente, se realiza recuperación y actualización con uso de (estructura de datos y algoritmos). Adicionalmente, se deb manejar un modelo adecuado para la información en tiempo real y la información que se consulta posteriormente. Este modelo se presenta en detalle en la solución propuesta más adelante.
- ***Exposición de servicios***: Se utilizarán Servicios web que expondrán los datos del índice semántico desarrollado.

5.3.3 Dependencias y Asunciones

La lista de todo lo que se asumió para la arquitectura:

- Se utilizará un servidor IoT que será seleccionado en tareas posteriores, del cual se implementará a partir de sus API, la funcionalidad necesaria para consultar y extraer los archivos de metadatos en JSON.
- Se utilizará la ontología SSN-XG, WordNet y una ontología de Objeto Semántico propuesta por la tesis de doctorado del Magister Miguel Ángel Niño, con el fin de almacenar el conocimiento necesario para indexar los objetos relacionados en la consulta, asi como la alineación a la ontología ENVO.
- Se utilizará la Herramienta CMAPS – COE que permite crear modelos conceptuales y después convertirlos en ontologías OWL, las cuales serán refinadas posteriormente con otra herramienta.
- Se utilizará Protege como herramienta de ontologías para alinear las ontologías seleccionadas.
- Se utilizará infraestructura de terceros para los procesos de indexación de la información de los objetos del servidor IoT. La infraestructura es: Lucene.NET, Json.NET, NGeo, RestSharp, StemmersNet, APIs de Xively y APIs de WordNet.

- Se realizará un enriquecimiento de los archivos de metadatos e información de los objetos provistos por el servidor IoT de acuerdo a las ontologías utilizadas y luego se indexará utilizando el cálculo de TF –IDF (frecuencia de términos – frecuencia inversa del documento), donde se mide la frecuencia de los conceptos candidatos en los documentos que fueron recuperados, estos resultados son posteriormente guardados en una matriz.
- Para el cálculo de la similitud en el índice se hará la asignación de pesos a los conceptos de la ontología, teniendo en cuenta la jerarquía de la misma. Los pesos fueron asignados consecutivamente dando más peso a los hijos que los padres esperando tener mejores resultados en las búsquedas. La estrategía es mediante una expansión de consulta de usuario.

5.3.4 Vistas Arquitectónicas

En este punto se presenta la vista lógica del sistema, como una personalización de la arquitectura general propuesta en el método definido y una vista de casos de uso generales, los cuales definen los requisitos específicos de la aplicación de búsqueda semántica a desarrollar.

5.3.4.1 Vista Lógica de la Arquitectura

Esta vista describe la estructura y el comportamiento de las partes componentes del sistema. Se reutiliza la arquitectura propuesta en el método y se personaliza a las decisiones tomadas. En la Figura 15, se puede apreciar en azul las capas que se han personalizado con respecto de la arquitectura general propuesta en el método.

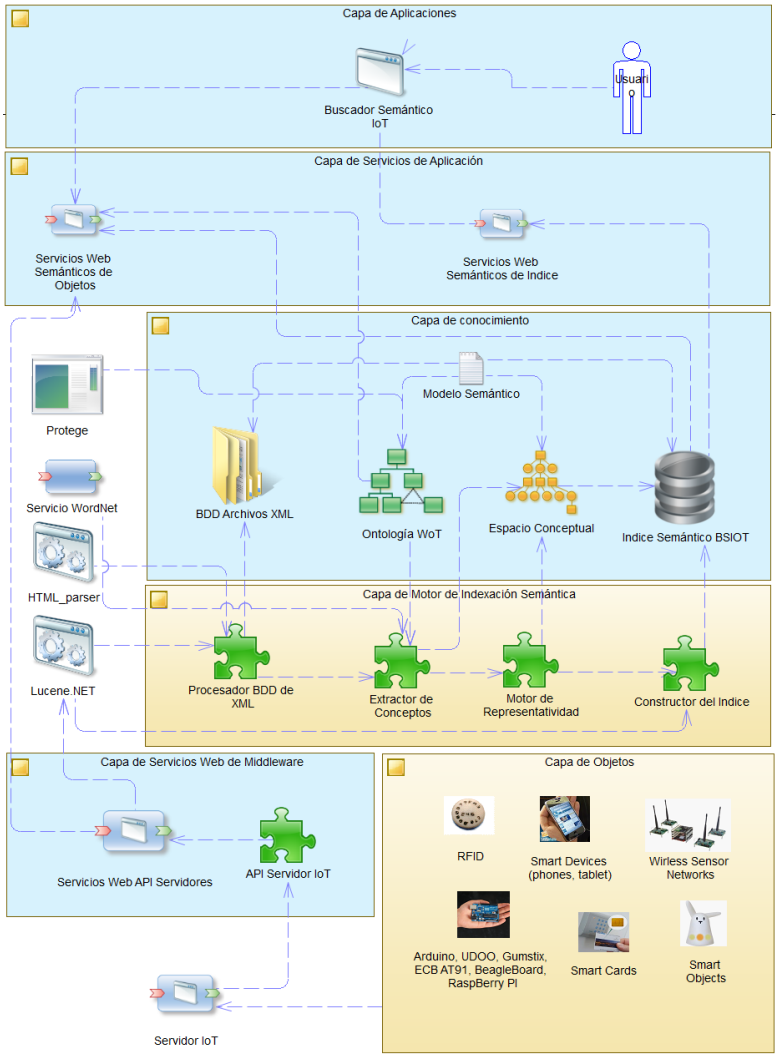


Figura 15. Arquitectura Lógica del Buscador Semántico IoT

Las herramientas software y servicios que son ajenos al sistema se han colocado por fuera de las capas. Se pueden apreciar las herramientas específicas y en que etapas de proceso son utilizadas en incluidas en el sistema.

Siguiendo con el análisis de la figura en la capa de servicios está el servidor IoT, el cuál será seleccionado en el estudio de evaluación de servidores. Este servidor debe ofrecer un API de desarrollo que permite acceder a todos los objetos de la IoT que tenga conectados a su base de datos centralizada. Dado que el servidor IoT no permite la interacción hacia los objetos el modelo está realizado sólo para obtener datos de los mismos.

La capa de indexación es igual a la propuesta en el modelo general, sin embargo cuando se realice la implementación de cada una de ellas se podrá ver las decisiones específicas de indexación semántica que se reúnen en el documento

del modelo semántico, el cual afecta cada uno de los productos de la indexación que se encuentran en la capa de conocimiento.

Capa de conocimiento, los productos más importantes son el índice semántico y la personalización de la ontología, ésta se debe desarrollar con el caso de estudio particular escogido que fue la contaminación medioambiental y la alineación a las estandarizadas existentes.

Finalmente, la capa de servicios, exponen servicios web para ser utilizados en la capa de aplicaciones. Estos servicios pueden acceder directamente al índice, a la ontología y al servicio creado sobre el API del servidor de IoT seleccionado, con el fin de obtener la información necesaria de primera mano.

5.3.4.2 Vista de Caso de Uso General

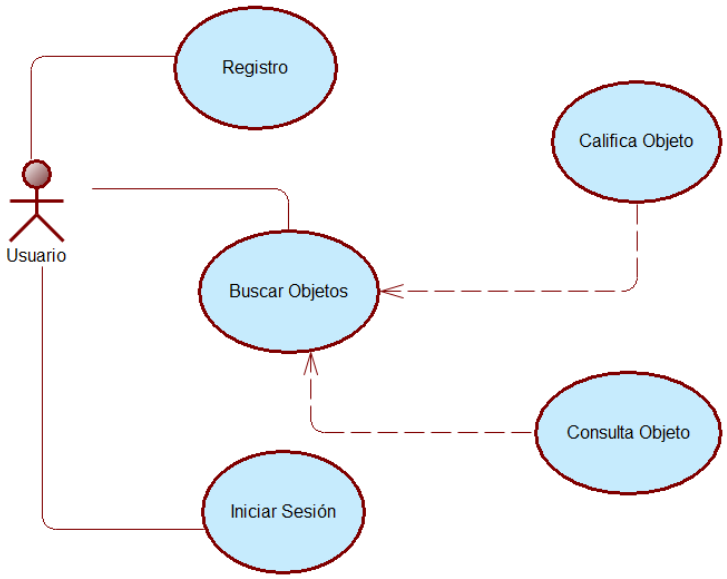


Figura 16. Modelo General de Casos de Uso

En la Figura 16 se pueden ver los casos de uso generales del sistema. El usuario puede realizar en general tres acciones cuando ve la página web por primera vez: Registro, Buscar Objetos o Iniciar Sesión. La versión final incluyo un segundo actor (Administrador) y opciones de gestión del índice y la ontología.

Si no se ha registrado puede hacerlo, sin embargo no es necesario que se registre para poder realizar las búsquedas, por defecto el usuario es iniciado como anónimo y se le permite también calificar y consultar objetos.

Se inicia sesión cuando se necesite identificar al usuario ya sea para que califique los objetos recuperados y poderlos asociar a un usuario particular y también ofrecer en el futuro mejores funcionalidades a los usuarios registrados.

5.3.5 Ubicación de los Productos

El producto “Visión del Sistema” contiene la arquitectura definida para crear el índice semántico. Se encuentra en la siguiente ruta del CD de entrega:
C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 1\Actividad 1\Tarea 3.

5.4 Fase I – Actividad 1: Tarea 4: Modelo Semántico BSIoT

5.4.1 4.4.1 Introducción

El presente documento presenta el modelo desarrollado para la construcción del índice semántico en la IoT. Este modelo incorpora los cuatro modelos que requiere la arquitectura general propuesta en la metodología de desarrollo. Primero se realiza una respuesta general a las preguntas guías para el desarrollo del modelo y posteriormente se profundizan con la presentación de los submodelos desarrollados para el caso de estudio particular.

5.4.2 Preguntas de Referencia del Modelo

Debido a que este modelo se deriva de la propuesta metodológica de desarrollo de índices semánticos para la IoT, en esta primera parte se introduce a como se responden las preguntas del modelo semántico para el caso particular de desarrollo:

1. **¿Cuáles son los usuarios del índice semántico?:** Los usuarios del índice son personas de la Web que estén interesadas en buscar objetos de la WoT que midan condiciones ambientales con el fin de reutilizar su información.
2. **¿Cuáles son los tipos de consultas que va resolver el índice semántico?:** Ya que el dominio del índice semántico se relaciona con el medioambiente se definen primero el rango de mediciones medioambientales cómo lo siguiente: temperatura, niveles de ruido, humedad, calidad del aire (CO2), presión atmosférica, calidad del agua, radiactividad, movimiento. Las consultas que puede resolver son:
    a. ¿Qué dispositivos cercanos a <<lugar>> pueden entregarme <<información medioambiental>>?
    b. ¿Qué <<información medioambiental>> puedo obtener de <<lugar>>?
    c. ¿Cuál es el histórico de la <<información medioambiental>> en el periodo de tiempo <<periodo>>?

d. ¿Qué <<variable medioambiental>> se puede medir en <<lugar>>?
e. ¿Cuál es la tendencia de la <<variable medioambiental>> en <<lugar>>?
f. ¿Cuáles <<variable medioambiental>> presentan <<contaminación ambiental>> en <<lugar>>?

1 Los tipos de consultas a), b), d) y e) se pueden obtener directamente de la información almacenada en el índice, a su vez el índice se crea en una primera etapa con consultas directas a al servidor IoT a través del servicio web de API. Las preguntas c) y f) se deben consultar en tiempo real al servidor IoT por datos y luego calcular las mediciones solicitadas a través del modelo de servicios que se desarrolle.

3. **¿Cómo se van a manejar las consultas que hacen los usuarios para que se pueda entregar resultados relevantes?:** Las consultas de los usuarios se realizará un preprocesamiento similar a la de los documentos, con el fin de expandir su consulta y alinearla con los conceptos de la ontología medioambiental seleccionada y adaptada a las necesidades del proyecto.
4. **¿Qué cantidad de información se va a indexar y cual se va a consultar directamente al sensor?:** Se va a realizar consultas sobre el servidor IoT acerca de todos los conceptos relacionados con la información medioambiental, que se encuentran en la ontología que ha sido adaptada para la creación del índice. Por cada sensor que coincida con la búsqueda se recupera la información en formato XML de sus datos y metadatos. Cuando se necesite información de los datos actuales o datos históricos se realizará una consulta directa al servidor IoT, recuperándolos y presentándolos o reprocesándolos en caso de que sea necesario.
5. **¿Cómo se va a manejar la concurrencia de consultas a un mismo sensor?:** La concurrencia se manejará en dos niveles, el primero es que el API del servidor IoT maneja la concurrencia a través del servidor Web en el cual se alojen los servicios. El acceso a los datos del índice se van a realizar en una base de datos SQL Server, con el fin de manejar la concurrencia con el mismo motor. Dado que los dispositivos no reciben instrucciones para actuadores, ya que el sistema es de sólo consulta, por ello no existe problemas de instrucciones conflictivas al mismo tiempo.
6. **¿Cómo se va a modelar el contexto que necesariamente está asociado a los sensores?:** El contexto que se va a manejar inicialmente se cierra a los datos primarios (Perera, Zaslavsky et al. 2013) de localización, identidad, tiempo y actividad de los sensores, los cuales en el servidor IoT deben están asociados a una ubicación geográfica y unos metadatos opcionales que coloca el creador del objeto. Si el tiempo lo permite se puede explorar datos secundarios en las mismas categorías. Los datos secundarios corresponden a un preprocesamiento de los datos de acuerdo a necesidades específicas. Sin embargo el principal dato secundario es el cálculo de cercanía de sensores respecto de un lugar particular.
7. **¿Cómo se va a manejar la frecuencia de actualización del índice?:** Con respecto a la frecuencia se pretende hacer dos etapas. La primera antes de

liberar el buscador se iniciará el proceso de poblar el índice con los sensores actuales en el servidor IoT seleccionado. Posteriormente se libera el Buscador y este trabajara con los sensores que fueron indexados en esa primera etapa. Si es necesario hacer una actualización se procederá de la misma manera pero de forma manual y programada.

8. **¿Cuál es la estructura de datos más adecuada para almacenar el índice**?: Se decidió adaptar una ontología de dominio específico en información medioambiental (ENVO[26]) a una ontología de contaminación ambiental desarrollada para este proyecto. Adicionalmente estas ontologías serán alineadas a su vez con la Ontología "Objeto Semántico" aportada por Niño, con el fin de tener una ontología que permita incluir el conocimiento necesario a los sensores para que se puedan responder las preguntas realizadas en el buscador.
9. **¿Cómo se va a compartir el índice para su correcto aprovechamiento?**: El índice creará unos servicios Web semánticos que se publicarán en un servidor Web público. Los resultados de las búsquedas sobre este servicio permitirán exponer información de búsqueda en el dominio medioambiental y enlaces a servicios web de los objetos específicos para que sean aprovechados por aplicaciones de terceros.

5.4.3 Modelos del Índice Semántico a Construir

En los siguientes apartados se determinan los modelos encargados de establecer las estructuras de conocimiento a utilizar, establecer el dominio conceptual de objetos, estructura de datos y algoritmos para la recuperación y actualización de los conceptos.

5.4.3.1 Modelo de Conceptos

Este modelo presenta las decisiones sobre cómo manejar la información del contexto de los dispositivos, es decir la información que se interrelaciona a los datos generados por el sensor o dispositivos y se presentan decisiones de cómo se va a manejar la información semántica de los objetos. En el **¡Error! No se encuentra el origen de la referencia.**-**¡Error! No se encuentra el origen de la referencia.** se presenta el caso de estudio el cual es la "Información Medio Ambiental y la Contaminación Producida por el hombre".

- **Alineación con Ontologías de Dominio Existentes:** El modelo conceptual de información medioambiental y contaminación del hombre despues de definir las variables, conceptos y contextos se genera un mapa conceptual el cual se

[26] Environment Ontology (http://environmentontology.org/home), consultado, 28/06/2013.

convertirá en una ontología OWL. Esta ontología debe ser adaptada y formalizada para que pueda llevada a la producción. La construcción de la ontología también contemplo la utilización de otra ontología realizada por expertos la cual abarcaba conceptos de tipificación ambiental de área o lugar se realizó para estar en la línea de estandarización y aumentar la posibilidad de que la ontología desarrollada en el proyecto sea aceptada por la comunidad científica. La alineación de la ontología desarrollada de contaminación medio ambiental con la ontología estándar ENVO.

Alineación con Ontologías de Dominio Existentes

El mapa conceptual de la Figura 17, se convertirá en ontología en formato OWL. La cual debe ser revisada y adaptaba para que quede formalizada de forma correcta. Sin embargo, se encontró una ontología en el área medioambiental desarrollada por expertos, la cual se puede aprovechar para obtener más conocimiento con respecto de la tipificación ambiental del área o lugar que se esté midiendo, adicionalmente aumenta la posibilidad de aceptación y uso por parte de la comunidad interesada. La alineación está dada de la siguiente manera (Kavouras and Kokla 2000; Kavouras 2005):

1. El concepto ***Bioma*** corresponde con el concepto (***Biome***) de ENVO. Así se conecta el ***biotipo*** a ***biome*** creando una clase equivalente a "***bioma***" del modelo conceptual. El Biome es un concepto raíz de la ontología ENVO y este desarrolla todos los conceptos relacionados por expertos al medioambiente, caracterizando de mejor forma el biotipo.
2. El concepto ***Ambiente*** corresponde con el concepto (***Habitat***) de ENVO. Así se conecta el Medio Ambiente a hábitat de ENVO. Este concepto desarrolla todos los tipos de hábitat disponibles en la ontología y clasificados de forma experta.
3. Los niveles de ***materiales del ambiente*** (***environmental material***) y las ***características ambientales*** (***environmental feature***) están asociadas a los biomas, a través de ellos se puede encontrar más información de los biotipos relacionados a determinados Biomas. Sin embargo para esta versión estos conceptos no se alinean ya que interesa más el trabajo en contaminación que en la parte caracterización medioambiental.
4. Finalmente, la condición ambiental (**environmental condition**) permite clasificar los biomas en categorías mayores, los cuales permiten entender mejor el tipo de hábitat. También se pueden llegar a estos a través del Biome.

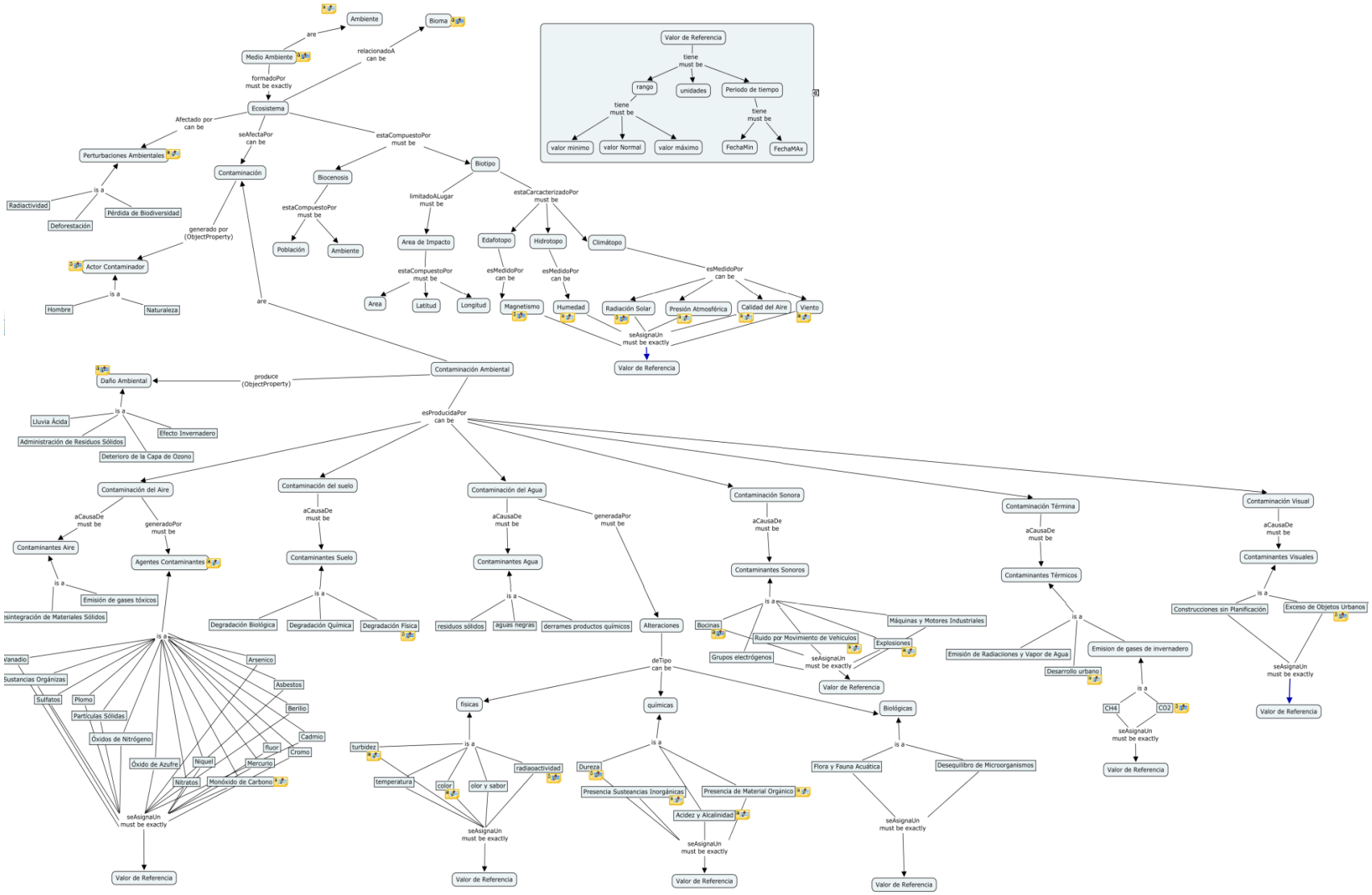


Figura 17. Mapa Mental del Modelo Medio Ambiental del proyecto

El elemento interesante el cual aporta la ontología desarrollada es la modelización correspondiente a la contaminación del medio ambiente y los posibles sensores que se pueden relacionar en cada elemento contaminante. Adicionalmente, se relacionan al Biotipo sensores que pueden caracterizar el mismo y se pueden definir medidas normalizadas, que posteriormente se pueden comparar con las medidas reales. Para este caso se utilizará la tabla de la **Global 200,** la cual es la lista de las ecorregiones globales o biorregiones identificadas como prioritarias para la conservación por el World Wide Fund for Nature (WWF), específicamente los datos relacionados con Colombia.

Cuando se relacionen los sensores a cada uno de los elementos conceptuales medioambientales, los sensores se representarán a través de la ontología de objeto semántico, desarrollada en el proyecto de doctorado de (Niño-Zambrano 2013) al cual aporta también este proyecto. Esta ontología permitirá definir los aspectos relevantes de los dispositivos aplicables al caso de estudio particular como: propiedades, elementos de contexto (conocimiento) y servicios a ofrecer (métodos). En la Figura 18, siguiente se presenta el objeto semántico atemperado a este trabajo:

Este modelo ya está en su segunda versión y ha sido alineado con las ontologías de observación de sensores SSN-XG (Compton, Barnaghi et al. 2012). Además del aprovechamiento de propuestas ontológicas como: clasificación de los objetos propuesto por (Mathew, Atif et al. 2011), manejo de contexto social de (Biamino 2012), modelo de servicios de (Kostelník, Sarnovský et al. 2011) y de representación de conocimiento IoT propuesto por (Wei, De et al. 2012). Esta alineación no se presenta en este documento por políticas de propiedad intelectual.

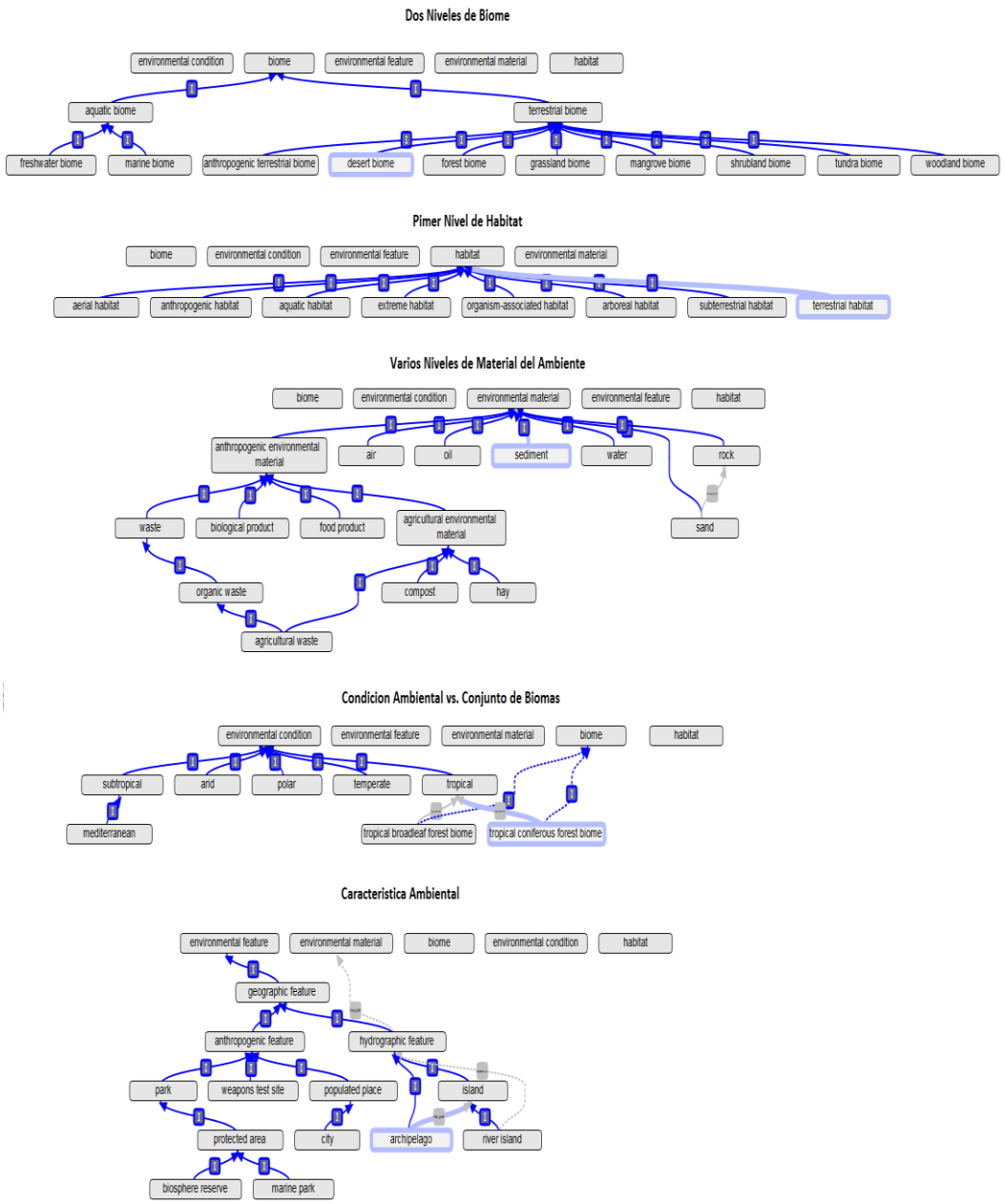


Figura 18. Vista Rápida de Alineación a la Ontología ENVO

En la Figura 19 se muestra el modelo conceptual de la ontología de objeto semántico.

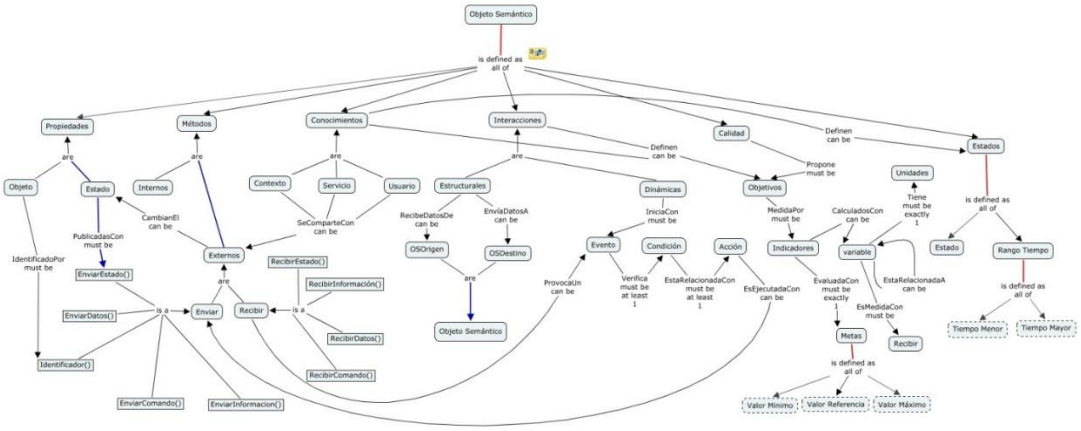


Figura 19. Modelo Conceptual del Objeto Semántico Tomado Tesis doctoral Miguel Angel Niño Zambrano

De la Ontología SSN-XG se utiliza el módulo de medición y el módulo de Observación. El primero permite describir un sensor y el segundo establecer que está midiendo. El módulo de capacidades se deja para una versión posterior del índice. Así tenemos:

Concepto de SSN-XG	Conceptos e Instancias Alineadas
ssn:SensingDevice	Se utiliza la ontología de "**Automatic Weather Station - AWS**" para crear subclase de conceptos relacionados con los tipos de sensores de las estaciones climáticas. Estos conceptos se relacionan a los tipos de sensores definidos en la ontología de contaminación medioambiental.
ssn:FeatureofInterest y ssn:Property	Se utiliza el aporte del "Climate and Forecast Metadata Conventions", que ofrece una ontología con "cf-features" que aporta una colección de nombres estándar para variables climáticas e incluida en "AWS". A través de la relación **ssn:hasProperty** se enlaza a **ssn:Property**, ésta última crea instancias de "cf:property. Las propiedades son relacionadas a las variables de contaminación definidas en la ontología de contaminación medioambiental del proyecto.
ssn:Platform y Area de Impacto	Para relacionar los lugares geográficos, que para este caso de estudio corresponde a las ciudades de Colombia, se utilizará la información existente en el servicio **GEONAMES** (http://www.geonames.org), la cual a partir del nombre de la cuidad se obtiene las coordenadas geoespaciales de su localización en el planeta. Estas instancias se crean bajo el concepto **Area de Impacto**, el cual a su vez relaciona una plataforma con **ssn:Platform** y este a su vez a un **ssn:SensingDevice**. La plataforma que se crea es virtual, la cual agruparía los sensores distribuidos y cercanos a dicha ciudad. Para el caso particular se colocaran las instancias de dos ciudades: "Popayán" y "Bucaramanga".

Tabla 31. Alineación de Ontologías SSN-XG y AWS

En la Tabla 31, se presenta adicionalmente el uso de otra ontología denominada Automatic Weather Station – AWS. La cual permite utilizar nombre estándar para los tipos de sensores y variables medioambientales, asegurando en mejor medida la inclusión de conocimiento experto a la ontología y de estándares que pueden ser utilizados por varios propietarios de dispositivos sensores.

De la clase SesnsingDevice se relaciona al Biotipo para medir cada una de las siguientes características con sus respectivos sensores a trvés de estrategia de clases equivalentes:

1) Humidity = Humedad
2) Atmosferic Pressure = Presión Atmosférica
3) Radiation = Radiación Solar
4) Temperature = Temperatura
5) Wind = Viento
6) Precipitation = Precipitación

Del proyecto de Sensores Virtuales se adicionan los siguientes conceptos:

1) Monóxido de Carbono = CO. Se adiciona etiqueta.
2) Partículas Suspendidas Totales = PST. Se crea el individuo de la clase "Agentes contaminantes" y se agrega una etiqueta con el acrónimo.
3) Material Partículado Menor a 10 micrómetros = MP10
4) Óxido de Azufre = SO2. Se crea una etiqueta con el acrónimo SO2.
5) Óxidos de Nitrógeno = NO2. Se crea una etiqueta con el acrónimo NO2.
6) Ozono = Ozone = O3. Aunque el ozono no es un agente contaminante, se ha adicionado a esta clase con el fin de tomar sus mediciones, como lo establece la norma colombiana de Sistemas de Calidad del Aire.

Adicionalmente, se utiliza el servicio de GEONAMES con la finalidad de caracterizar los lugares (ciudades) e identificarlos geoespacialmente, así también se asegura una correcta referenciación de dichos sitios lo más cercanos a la realidad espacial.

- **Ontología de Mediciones en Contaminación Medioambiental:** Finalmente y después de las alineaciones se presenta la Ontología que se utilizará para anotar los documentos de los dispositivos o sensores que se encuentre en el servidor IoT para su posterior indexación.

  Esta ontología será usada en esta primera versión con propósitos de anotar semánticamente los documentos analizados del servidor IoT, para que posteriormente se indexen para su consulta. En el momento de la consulta también será utilizada para presentar ayuda de autocompletado al usuario y así guiar mejor el proceso de búsqueda.

  La ontología no será usada para instanciar sensores y razonar sobre la misma las consultas. Esta opción se deja para la segunda versión del proyecto o trabajo futuro.

5.4.4 Modelo de Consultas Personalizadas

Se establece la información del contexto(Zhang, Cheng et al. 2012) teniendo en cuenta que los usuarios generalmente están interesados inicialmente en las entidades (cosas, lugares, personas) y posteriormente por las cualidades o estados de esas cosas (vacío, libre, sentado, caminando, etc.) y no en sensores individuales y sus datos de salida.

A continuación se presenta el modelo para establecer las consultas que el buscador responde en función del caso de estudio de contaminación medioambiental:

Modelo de Consultas Personalizadas

Establecen la información del contexto(Zhang, Cheng et al. 2012) teniendo en cuenta que los usuarios generalmente están interesados inicialmente en las entidades (cosas, lugares, personas) y posteriormente por las cualidades o estados de esas cosas (vacío, libre, sentado, caminando, etc.) y no en sensores individuales y sus datos de salida. El presente proyecto comparte la misma idea y por ello las preguntas van más enfocadas a entidades medioambientales (lugar) y sus estados. En las preguntas de referencia se había definido cuatro preguntas iniciales (a-e):

a. ¿Qué dispositivos *cercanos* a <<lugar>> pueden entregarme <<información medioambiental>>?
b. ¿Qué dispositivos se encuentran cerca de Popayán que entreguen información medioambiental?
    - ¿Qué dispositivos miden información medioambiental aquí?
c. ¿Qué <<información medioambiental>> puedo *obtener* de <<lugar>>?
    - ¿Qué información medioambiental hay en Colombia?
    - ¿Qué información medioambiental hay en latitud (°, ', ") y longitud (°, ', ")?
d. ¿Cuál es el *histórico* de la <<información medioambiental>> en un <<lugar>> en el <<periodo de tiempo>>?
    - ¿Cuál es el histórico de la información medioambiental en Popayán entre el 01/01/2010 y el 31/12/2010?
e. ¿Qué <<variable medioambiental>> hay en <<lugar>> y en que [<<periodo de tiempo>>]?
    - ¿Qué temperatura hay en Madrid?
    - ¿Qué humedad hay en New York entre el 30/03/2013 hasta hoy?

Aquí vamos a detallar en que consiste cada ítem encerrado entre paréntesis angulares:

- <<lugar>>: Corresponde en el sentido más simple a un punto geográfico con latitud y longitud, el cual está caracterizado en el área de impacto en la ontología. Sin embargo, como los biotipos están relacionados al biome en la ontología ENVO, podemos a partir de allí extender el concepto a todos las tipificaciones desarrolladas pudiendo caracterizar más explícitamente el lugar y ampliar el rango de preguntas en este sentido. Adicionalmente, la información de metadatos de los objetos de la WoT contiene su posición en latitud y longitud, lo cual se convierte en una ventaja para interrelacionar sensores con biotipos, ecosistemas y hábitat. Los lugares geográficos adicionalmente se relacionan al servicio de Google Maps y Geonames con el fin de establecer a partir de los nombres de ciudades o lugares los datos geo-posicionales.
- <<información medioambiental>>: Esta información se refiere a cualquier concepto debajo de este nodo de la ontología. Así podemos dar una información clasificada y bien definida de lo que caracteriza cierta ubicación geográfica y relacionarla con las mediciones actuales de los sensores encontrados en dicha ubicación. Si el concepto se encuentra en niveles superiores, el reporte puede incluir varios niveles de medición de sensores y sus posibles interacciones, dependiendo de los modelos de servicios desarrollados.
- <<periodo de tiempo>>: Compuesta de dos variables, una de fecha inicio y otra de fecha fin, con la condición que la fecha fin debe ser mayor que la fecha inicio. En los casos en que no se provee el rango se asume el periodo de tiempo puntual del momento en el que se hizo la consulta.
- <<variable medioambiental>>: Corresponde a las variables relacionadas al biotipo y en las cuales, dependiendo de la ubicación geográfica, podemos relacionar ciertos sensores existentes. Estas variables deben tener previamente un valor de referencia para poder realizar las comparaciones correspondientes con los datos medidos. El valor de referencia puede ser un rango. Por ejemplo: la temperatura en verano en España normalmente oscila entre 30 y 40 grados, lo que esté por debajo o por encima en ese periodo no es normal.
- <<agentes contaminantes>>: Corresponde a todas las variables medidas de elementos que contaminan el medio ambiente y de la cual se encontraron sensores relacionados. Cuando su medición está por fuera de los rangos normales se considera como agente que está incidiendo en el medio ambiente.
- <<daños ambientales>>: Corresponde a las principales preocupaciones de daños al medioambiente generados por el hombre y que se podría acercar a esta conclusión tomando como base las mediciones de los agentes contaminantes del lugar particular. Esto es un proceso complejo y aquí sólo pretendemos presentar una forma útil en que podríamos encontrar este tipo de respuestas, pero se espera que los modelos de servicios sean desarrollados por expertos del tema.

Los verbos (resaltados en cursiva) utilizados en los tipos de preguntas: cercanía, históricos, tendencia, diferente, causando y generar, se definirán en el modelo de servicios de este mismo documento, ya que corresponden a operaciones que se deben desarrollar en el dominio de la aplicación.

Con el fin de que las entradas sean más precisas se implementara un sistema de autocompletado en línea, en el cual cuando el usuario este digitando una consulta se buscaran palabras clave en la misma ontología y se le guiara en la forma correcta de hacer la estructura de la pregunta con el fin de obtener mejores resultados.

5.4.5 Modelo de Servicios

Es un modelo que establece las interfaces a desarrollar con respecto a los servicios web de los objetos y los servicios Web del índice mismo.Los servicios básicos del índice son:

1. ***Buscar (consulta, idioma)***: Esta función recibe un conjunto de caracteres a los cuales les hace un proceso de ampliación de consulta y posteriormente busca en el índice las coincidencias con respecto a toda la información que ha sido encontrada en el repositorio de información en la etapa anterior de indexación. Retorna un conjunto de resultados ordenados por relevancia. También recibe el idioma de manera opcional por el cual realizará la búsqueda y los resultados. Por defecto es español. Retorna un *DataSet* con un conjunto restringido fe información de los sensores, el cual es traído directamente de la información almacenada en el índice y por esto es más rápida.
2. ***BuscarFeeds (consulta, idioma)***: Es igual a la anterior, pero se diferencia en que retorna un listado de objetos denominados FeedXively, los cuales reúnen la información de los json anotados semánticamente que previamente fueron almacenados en la base de datos documental en el proceso de indexación. No trae *Datapoints*, o información medida por los sensores sólo los metadatos. Es más lento que su búsqueda predecesora ya que ha lectura de archivos local.
3. ***BuscarFeedsDataPoints (consulta, fechaInicio, fechaFin idioma)***: Esta función realiza lo mismo que las predecesoras, pero se diferencia en que trae toda la información del dispositivo incluido los *Datapoints* o mediciones directamente desde el servidor Xively. Es muy costosa en términos de tiempo.
4. ***RetornarConceptos (temino, idioma)***: Esta función recibe un término y lo compara en la ontología con el fin de establecer si es un concepto. Si es concepto retorna los conceptos más relacionados al mismo (relaciones directas a otros conceptos), desde el punto de vista del dominio de la ontología cargada. Adicionalmente, puede especificar el idioma el cual

puede ser: español, inglés o dejar vacío para traer datos en ambos idiomas.

5. ***RetormarMapaLugar (latitud, longitud, consulta, idiona, radio)***: Esta función recibe un lugar en términos de latitud y longitud, una consulta al índice. Devuelve la información relacionada a los sensores que tienen influencia en dicho lugar (determinado por el radio) y sus relaciones a conceptos medioambientales. El idioma y el radio son opcionales. Por defecto son español y 100 km. Esto lo realiza aplicando la fórmula de *Haversine* para calcular las distancias desde el punto provisto y los datos de latitud y longitud que poseen los sensores. Los sensores que no poseen estos datos son eliminados de los resultados finales.
6. ***RetormarMapaLugarDataPoints (latitud, longitud, consulta, idiona, radio)***: Realiza la misma acción que la función predecesora, sin embargo utiliza para la búsqueda la función *BuscarFeedsDataPoints*, permitiendo traer las mediciones de los sensores, esto la hace más costosa en términos de tiempo de respuesta.
7. ***RetornarMapaLugarListaSensores (latitud, longitud, ListaSensores, idiona, radio)***: Esta función retorna loa misma información que las anteriores, pero a diferencia de ellas recibe como parámetro un listado de objetos o sensores FeedXively en vez de la consulta, los cuales serán comparados con el lugar de búsqueda para retornar los que se encuentran ubicados en dicho lugar.
8. ***ExpandirConsultaConceptosOntologia (Consulta, idiona)***: Esta función recibe una consulta y la expande con conceptos de la ontología. Realiza un modelo de expansión teniendo en cuenta una similitud de cercanía entre conceptos. Añade conceptos hijos y relacionados a los conceptos identificados para hacer la consulta más precisa al domino de la ontología. Los términos no encontrados los compara con Wordnet para traer sinónimos, finalmente los no encontrados se añaden al final de la consulta.

Los servicios de Geolocalización se desarrollaron basados en el API de Geonames y la información que poseen en sus bases de datos, los servicios desarrollados en el índice son:

1. ***ObtenerLocalizacion(lugar)***: Permite Buscar un lugar geográfico por su nombre en el servicio Web de GeoNames. Retorna una lista objetos que corresponden con el lugar buscado y la información GeonameNode relacionada.
2. ***ObtenerCiudadesConsulta(Consulta)***: Esta función busca en la consulta, cuales conceptos corresponden a un lugar geográfico. Esta función se debe manejar con cuidado ya que términos comunes como aire, que también es del dominio en este caso, es un lugar geográfico con ese nombre.
3. ***ObtenerLugardeCoordenadas(latitud, longitud)***: Este método obtiene el Lugar más cercano a las coordenadas enviadas.
4. ***ObtenerListaLugarCoordenadas(geonodesSerializados)***: Esta función realiza el mismo proceso de ubicar un lugar específico, sin embargo lo hace a cada GeonameNode que ha sido provisto a través de una lista.
5. ***ObtenerListaLugarCoordenadasJerarquica(geonodesSerializados)***:

Obtiene las coordenadas de una lista de GeonameNode. Adicionalmente obtiene la jerarquía de lugares de cada nodo, con el fin de establecer con mayor precisión el lugar geográfico encontrado.

6. ***ObtenerListaLugarCoordenadas_AM1(geonodesSerializados)***: Obtiene las coordenadas de una lista de GeonameNode. Adicionalmente Obtiene de la jerarquía de lugares de cada nodo la primera división política que corresponde a dicho lugar. Esta función en otras palabras permite ubicar una región o (biotipo) para el caso de estudio.
7. ***ObtenerLugardeCoordenadasJerarquia(latitud, longitud)***: Obtiene la jerarquía de lugares de un lugar.
8. ***GeoNames_Hierarchy(GeonameId):*** Obtiene la jerarquía dado un GeonameNode a través de su identificador.

Los servicios web de los sensores son:

1. ***RetornarMetadatosSensor (IdSensor)***: Este servicio permite retornar en formato de objetos FeedXively la información relacionada al sensor particular. El enriquecido con la ontología a través de metadatos y se extraen de la base de datos documental almacenada localmente.
2. ***RetornarDatosSensor(IdSensor, [Fecha Inicio], [Fecha Fin])***: Esta función permite obtener información de los datos que mide el sensor entre el rango de fechas definido. En caso de no existir datos retorna vacio.
3. ***RetornarJsonSensor(idSensor)***: Esta función permite obtener información de metadatos en formato JSON directamente del servidor Xively.
4. ***RetornarDatapointsFeed(string IdSensor, string DatastreamId, string fechaInicio, string fechaFin)***: Esta función retorna los *datapoints* de un canal específico del dispositivo o sensor. Los datos son traídos directamente desde el servidor Xively con respecto al rango de fechas provisto.

Servicios del Caso se estudió de Información medioambiental:

1. ***OntenerBiotiposConsulta (ListaSensores)***: Después de realizar una consulta al índice semántico, se puede obtener los biotipos relacionados a dichos resultados. Para esto utiliza los datos de geo posicionamiento de los sensores y la obtención de la división política de primer orden que se encuentra en los servicios de Geonames.
2. ***ObtenerInfoBiotipoMedioAmbiente (luga, radio):*** Corresponde a un cuadro elaborado de conceptos medio ambientales (Edatopo, Hidrotopo, Climatopo) y las mediciones que se pueden obtener de los sensores del área de impacto (Biotipo) y la información de las variables definidas en la ontología. El área de influencia por ahora se maneja como el área circular alrededor del lugar y definida en el concepto de área de impacto del biotipo que se identifique.
3. ***ObtenerInfoBiotipoContaminacion (luga, radio):*** Igual a la anterior función, sin embargo centra su atención a los conceptos relacionados con

la contaminación ambiental.

4. ***HistóricosSensor (IdSensor, periodo de tiempo):*** A partir del periodo de tiempo retorna la información de los datos medidos y almacenados en el servidor IoT. Utiliza la función RetornarDatos de los servicios de sensores, pero debe implementar un mecanismo adecuado de presentación de la información y relación con la información medioambiental con el fin de que sea fácilmente entendible para el usuario.

5.4.6 Ubicación de los Productos

El producto “Modelo Semantico del Buscador BSIOT v8” contiene el modelo a seguir para la construcción del índice semántico. Se encuentra en la siguiente ruta del CD de entrega:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 1\Actividad 1\Tarea 4

5.5 4.5 Fase I – Actividad 2: Tarea 1: Evaluar el middleware / servidor IoT a utilizar (AEMS)

5.5.1 Introducción

El presente documento presenta el análisis realizado para la selección del Middleware de datos a utilizar para obtener la información de los sensores que permitirán satisfacer las necesidades de información de las consultas hechas por los usuarios.

5.5.2 Marco de Referencia de Servidores IoT

En la elaboración del método para crear índices semánticos en la IoT se hizo un estudio de los principales middleware existentes y de los cuales se podían utilizar para obtener información de los sensores que actualmente están conectados a la WoT (Tabla 32).

En la Tabla 32 presenta las características evaluadas y sus convenciones definidas con el fin de poder resumir adecuadamente la tabla comparativa. Las principales clasificaciones que interesaban al proyecto eran los tipos de metadatos que ofrecían de los sensores, formatos y la posibilidad de que se pudiera aceder a ellos de manera gratuita y en lo posible si restricciones, ya que la idea es que la herramienta y el índice se pensaron también crear de uso público.

Abreviatura	Definición de las abreviaturas	Abreviatura	Definición de las abreviaturas
Tipo de Arquitectura Implementada		*Tipos de Formatos de Estructura de Datos Soportados*	
R	REST (Reprsentational State Transfer)	J	JSON (JavaScript Object Notation)
Sk	Socket	X	XML (eXtensible Markup Language)
So	SOAP (Simple Object Access Protocol)	C	CVS (Comma Separated Values)
M	MQTT (Message Queuing Telemetry Transport)	Xd	XDR (eXternal Data Representation)
Tipos de Metadatos soportados		*Disponibilidad de la funcionalidad*	
G	OGC (Open Geospatial Consortium)	√	Quiere decir que si lo soporta
Em	EML (Environmental Markup Language)	-	Quiere decir que no se encontro información
Tg	Tags (Etiquetas definidas por los usuarios)	*Tipos de Usuarios*	
Xp	XMPP (Extensible Messaging and Presence Protocol)	T	Todo Tipo de Usuarios
Co	COAP (Constrained Application Protocol)	Ep	Empresriales
Lp	6LoWPAN (IPv6 over Low power Wireless Personal Area Networks)		
Tipos de Redes		*Tipos de Costos*	
I	Internet	O	Open: Abierta sin costo
M	Móviles	P	Paga: Se debe pagar por sus servicios
St	Satélite	O/P	Servicios Open y otros sonPagos
Formato de sindicación			
At	ATOM (Atom Publishing Protocol)		
Rs	RSS (Really Simple Syndication)		

Tabla 32. Convenciones de Servidores Mediadores de la IoT

El resultado del estudio comparativo se puede ver en la **¡Error! No se encuentra el origen de la referencia.**. Inicialmente se clasificaron las características en Soporte y desarrollo de aplicaciones, cobertura y servicios que ofrece. Lo anterior permitió seleccionar el servidor Xively (antes se llamaba Cosm y PachuBe) como el más adecuado para el estudio particular por su API de desarrollo abierto y de fácil utilización tipo REST, la gran cantidad de objetos conectados, los diferentes sormatos en los que ofrece su funcionalidad y la posibilidad de crear aplicaciones sin costos.

CARACTERISTICAS ANALIZADAS																				
PLATAFORMAS	**Soporte Desarrollo Aplicaciones**					**Cobertura**			**Servicios**											
Nº	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	Uso de Estándares de metadatos	Uso de formatos de Estructura de Datos	Arquitecturas que Soporta IoT	API de desarrollo para diferentes Lenguajes	Ciclo de vida y Despliegue de Productos	Redes que da soporte	Cantidad de Objetos Conectados*	Cantidad de Empresas Conectadas	Gestión de Metadatos	Gestión Datos en Tiempo Real	Directorio de Objetos - Búsqueda	Servicios de Negocio	Servicios de Personalización y Aplicaciones	Seguridad, Privacidad y Autorización	Usuarios en los que se focaliza	Costos	Escalabilidad	Extensa Documentación	Formato de sindicación	Trigger
AMEE	G,Tg	J	R	√	√	I	342	0,26	√	√			√	√	Ep					
Arkessa				√		I,M,St			√	√	√	√	√	√	T	O/P				
Axeda			R,So	√	√	I,M,St	1.2	150	√	√	√	√	√	√	T	P				
Bugswarm			R	√		I		300				√	√		T					
Carriots cloud		J,X	R	√		I				√				√	T,Ep	O/P		√		√
Eart Cam						I,M						√		√	E	P				
Evrythng		J	R	√		I				√		√		√	T,Ep	O		√		
GroveStreams		J	R	√		I,M				√			√	√	T,Ep	O/P		√	Rs	
Sensinode	Co,Lo				√	I,M,St				√		√	√	√	Ep	P				
Nimbits	Xp	J,X	R	√		I				√	√				T	O	√	√		√
Exosite		J		√	√	I,M,St				√	√	√	√	√	T,Ep	O/P	√	√		

CARACTERISTICAS ANALIZADAS																				
PLATAFORMAS	**Soporte Desarrollo Aplicaciones**					**Cobertura**			**Servicios**											
Nº	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
	Uso de Estándares de metadatos	Uso de formatos de Estructura de Datos	Arquitecturas que Soporta IoT	API de desarrollo para diferentes Lenguajes	Ciclo de vida y Despliegue de Productos	Redes que da soporte	Cantidad de Objetos Conectados*	Cantidad de Empresas Conectadas	Gestión de Metadatos	Gestión Datos en Tiempo Real	Directorio de Objetos - Búsqueda	Servicios de Negocio	Servicios de Personalización y Aplicaciones	Seguridad, Privacidad y Autorización	Usuarios en los que se focaliza	Costos	Escalabilidad	Extensa Documentación	Formato de sindicación	Trigger
Open.Sen.se		J	R	√		I			√	√					T	O		√		
Paraimpu		J	R			I									T	O				
NewAer					√	I,M,St							√		Ep	P				
SensorCloud		C,Xd	R	√		I,M,St			√	√		√	√	√	Ep	P		√		
SensorPedia	G	J	R	√		I,M,St			√	√	√		√		Ep	O	√	√	At	
Sensormap	Tg			√		I,M,St			√	√	√		√	√	T		√	√		
SenseWeb	Tg			√		I,M,St				√	√		√	√	T		√	√		
SensorBase	Em			√		I,M,St				√	√		√	√	T		√	√	Rs	
ThingSpeak		J		√		I	0.01		√	√	√		√	√	T,Ep	O/P		√	Rs	
Thingworx				√	√	I,M,St				√	√	√	√	√	Ep	P	√			
Xively Cloud Services* **	G,Tg	J,X,C	R,Sk,M	√	√	I	250	55	√	√	√	√	√	√	T	O	√	√	Rs	√
Yaler		J		√		I				√				√		O			Rs	

* Millones de dispositivos.

Tabla 33. Comparación de Características de Servidores Mediadores de la IoT

A parte de la clasificación de las características realizadas en la Tabla 33 para soporte al desarrollo, cobertura y servicios, adicionalmente, se agruparon los diferentes portales web de objetos de acuerdo a sus características interrelacionadas, con el fin de poder tipificar los portales relacionados a sus funcionalidades. En la Tabla 34 se presenta esta clasificación:

Nº de característica	Tipo de Servidor
2, 9, 17	**Orientado a Gestión de Datos**: Apoya el fácil acceso a las observaciones de sensor, el reusó de los datos y el peso ligero de los archivos.
1, 2, 4, 5, 9, 10, 11, 15, 16, 18	**Orientados al Desarrollo**: Apoyan el suministro de datos por medio de mecanismos sencillos, desarrollo rápido de aplicaciones para el usuario y bajo costo de implementaciones en las plataformas abiertas.
1, 3, 11, 14, 19, 20	**Orientados a Compartir Datos**: Apoya las mejores prácticas y las soluciones comprobadas provenientes de la web 2.0 para vinculación y control del contenido.
6, 7, 8, 12, 13	**Orientados a Usuarios**: Apoyo al desarrollo automático de productos que beneficie el reusó y aplicación de los datos.

Tabla 34. Tipos de Servidores Mediadores de la IoT

Aunque los servidores o portales de objetos pueden cumplir varias funcionalidades, es posible establecer cómo algunos privilegian un tipo de servicio

particular. Este estudio permitió definir parte los indicadores por los cuales se seleccionó el servidor IoT para el caso de estudio.

5.5.3 Ubicación de los Productos

El producto “Evaluación de Middleware y API de Desarrollo” contiene un análisis de los portales del IoT presentando un proceso de evaluación para seleccionar el más adecuado a las especificaciones buscadas. Se encuentra en la siguiente ruta del CD de entrega:

C: \CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 1\Actividad 2\Tarea 1

5.6 Fase I – Actividad 2: Tarea 2: Alineación de Conceptos del Servidor IoT y Fuentes Externas

En este apartado se analizan los formatos provistos por el servidor de objetos Xively que fue seleccionado en la tarea anterior, del cual se obtiene conceptos que serán equiparados en la ontología desarrollada. El portal Xively tiene un modelo de datos encargado de describir el dominio de conocimiento con el que representara la información. Este formato se procede a actualizar la ontología Objeto Semántico ya sea encontrando las similitudes con los conceptos existentes o ampliando los nuevos conceptos.

5.6.1 Ubicación de los Productos

El producto “Identificación de Categorías de Conceptos en los Formatos Estructurados”. Se encuentra en la siguiente ruta del CD de entrega:

C: \CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 1\Actividad 2\Tarea 2

5.7 Fase II – Actividad 1: Tarea 1: Seleccionar una metodología de desarrollo adecuada

5.7.1 Introducción

El presente documento establece la metodología que se sigue para la implementación de la ontología para el buscador de contaminación medioambiental. Aunque en la fase del modelo semántico ya se han definido las ontologías conceptualmente, en esta fase se procede con la construcción – reutilización de las mismas.

5.7.2 Implicaciones y Relaciones con el Modelo Semántico Propuesto

Con el fin de establecer el proceso de desarrollo de las ontologías, en la **¡Error! No se encuentra el origen de la referencia.** se presenta el ciclo de construcción basado en la fase de desarrollo de la metodología METHONTOLOGY (Corcho, Fernández-López et al. 2005). Adicionalmente se presentan los componentes principales de las ontologías a construir y su posterior utilización en una creación de servicios web.

El método inicia con una especificación. Para esto, además de definir los objetivos de la ontología se toma como principal insumo el estado del arte desarrollado para tal fin, reutilizando las taxonomías presentadas como la de (Mathew, Atif et al. 2011) y los estándares ontológicos existentes como SSN-XG (Compton, Barnaghi et al. 2012), las cuales serán aportadas a la fase de conceptualización con el fin de identificar su reutilización. Estos primeros pasos fueron realizados por los expertos en conocimiento en la definición del modelo semántico para el buscador propuesto.

Para la conceptualización se realizarán las tareas y formatos tabulares propuestas por METHONTOLOGY. En la formalización se utilizará una herramienta como CMAPS para representar semi-formalmente la especificación de las ontologías. En la construcción se trabaja con la herramienta Protégé (Noy, Sintek et al. 2001) y en formato OWL-DL se almacenaran las ontologías con el fin de su utilización y evaluación.

Finalmente, se plantea una etapa de construcción de servicios, en la cual se expondrán los servicios de la ontología a través de servicios web, para ser consumidos por aplicaciones y usuarios de Internet. Los servicios fundamentan su fuente de información en un índice semántico construido a partir de la interacción con los middleware y las ontologías construidas. Así mismo, los middleware, el mismo índice y las ontologías de aplicación soportan los servicios, como fuentes directas para hacer más ágil el funcionamiento de los mismos.

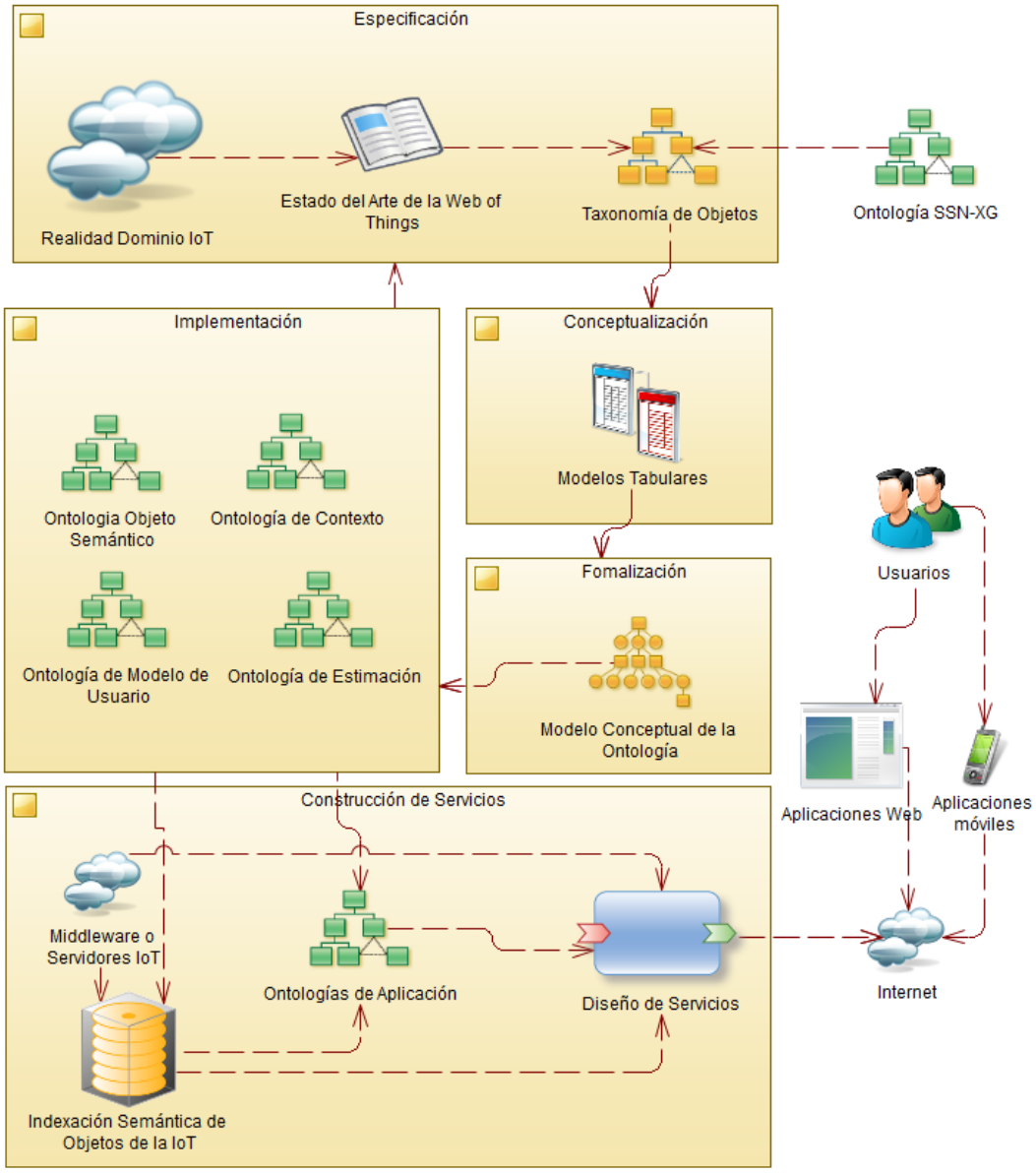


Figura 20. Diagrama Metodológico y de Componentes de la Ontología de Interacción Semántica WoT

Las ontologías de aplicación aparecen a partir de la personalización de las ontologías desarrolladas para la interacción semántica, estas ontologías reutilizan la información de las ontologías de dominio y definen tareas específicas con los elementos reales de un caso de estudio particular.

5.7.3 Principios de diseño de la ontología

Es importante definir unos principios de diseño de la ontología con el fin de mantener una coherencia con los objetivos de la misma. Ya que la ontología va de la mano con el conocimiento de la IoT y la WoT, es necesario tener presente que se necesitan estructuraras que faciliten su aprovechamiento, equilibrando

adecuadamente el impacto que produzcan estas infraestructuras con los requerimientos propios de los objetos de la IoT. Así, el trabajo de (Wei, De et al. 2012) en el cual proponen una ontología para esta área, definen unos principios de diseño, los cuales compartimos, adecuamos y ampliamos de la siguiente manera:

- ***Ligera***: un modelo ontológico ligero que equilibre adecuadamente la complejidad en la expresividad y la inferencia, tiene más posibilidades de ser utilizado y adaptado.
- ***Completitud***: la posibilidad de reutilizar los modelos anteriores y hacerlos en lo posible completos con respecto del dominio de la IoT garantizarán también su importancia.
- ***Compatibilidad***: la ontología debe ser coherente con las ontologías existentes bien diseñadas, así se pueden alinear con las existentes fácilmente.
- ***Modularidad***: la ontología diseñada se desarrolla con un enfoque altamente modular para facilitar su evolución, la extensión y la integración con ontologías externas.
- ***Abstracción***: La ontología debe permitir diferentes niveles de abstracción con el fin de ofrecer vistas adecuadas del conocimiento de los objetos de la IoT.
- ***Adaptable***: La ontología debe proveer un mecanismo de personalización, de tal forma que permita a sus usuarios adecuarla, ampliarla y reutilizarla como mejer les parezca.

El enfoque para la construcción de la ontología que permita interacción semántica debe seleccionar los elementos importantes de los estándares actuales para la descripción y manejo de los sensores como las especificaciones XML provista por la Open Geospatial Consortium (OGC) (Botts, Percivall et al. 2008) y las ontologías de sensores como SSN-XG (Compton, Barnaghi et al. 2012). Además del aprovechamiento de propuestas ontológicas como: clasificación de los objetos propuesto por (Mathew, Atif et al. 2011), manejo de contexto social de (Biamino 2012), modelo de servicios de (Kostelník, Sarnovský et al. 2011) y de representación de conocimiento IoT propuesto por (Wei, De et al. 2012).

El reto de alinear estas ontologías y seleccionar los elementos que más enriquecen el modelo aporta a la completitud, pero puede ir en detrimento de una propuesta ligera, por tal motivo se utilizará el principio de abstracción y modularidad con el fin de hacer más fácil de implementar y describir la ontología resultante.

5.7.4 Metodología de Diseño de Ontologías

Para el diseño de las ontologías se ha decidido realizar una adaptación de la metodología METHONTOLOGY (Corcho, Fernández-López et al. 2005), la cual originalmente propone las siguientes fases:

1. **Especificación**: ¿Cuál es su Nombre?, ¿Por qué se construye?, ¿Cuál es su

uso?, ¿Quiénes son sus usuarios?, ¿Cuál es el tipo de ontología?
2. **Conceptualización**: Convertir la percepción en representaciones semiformales y tabulares, del cual se puede obtener el modelo conceptual de la ontología.
3. **Formalización**: Transforma el modelo conceptual en uno formal o semicomputacional.
4. **Implementación**: Construye el modelo formal de ontologías en formatos estándar.
5. **Mantenimiento**: Actualización y corrección de la Ontología.

Se ha realizado una versión resumida y adecuada de dicha metodología con el fin de permitir construir las ontologías en versiones rápidas y evolutivas. Así se soporta la metodología en herramientas que permiten conceptualizar y formalizar automáticamente las mismas. Los pasos adecuados son:

1. **Abstracción Formal:** Se corresponde a especificación, conceptualización y formalización
    a. Se responden las preguntas de la especificación de la ontología.
    b. Con el fin de tener una idea clara del conocimiento a modelar se puede hacer una descripción textual o gráfica en compañía de los expertos o de una fuente de conocimiento válida. Se diligencian la plantilla tabular de glosario de términos propuesta en METONTOLOGY.
    c. Se utiliza la herramienta CmapTools COE[27] , las cuales permiten general el modelo conceptual gráficamente y siguiendo la especificación de patrones de diseño de COE con el fin de facilitar el proceso de implementación. Esta especificación evita tener que definir todas las plantillas de METHONTOLOGY.
2. **Implementación**:
    a. Exportar la Ontología en formato OWL a través de la utilidad de exportación de la herramienta CmapTools.
    b. Abrir la ontología en la herramienta Protégé [28], con la cual se revisa su correspondencia con la conceptualización y en caso de necesitar ajustes adicionales se realiza en esta etapa.
    c. Verificar el funcionamiento de la ontología teniendo en cuenta las preguntas que se espera que conteste, en lo posible utilizando el razonador integrado en Protégé.

---

[27] http://www.ihmc.us/groups/coe/, consultado 05/07/2013. Es un conjunto de herramientas software que permiten construir, compartir y visualizar ontologías basados en CmapTools. Estas herramientas son desarrolladas por Florida Institute for Human & Machine Cognition – IHMC y bajo el proyecto Concept-map Ontology Environment - COE

[28] http://protege.stanford.edu/, consultado 05/07/2013. Es un editor de ontologías de código abierto y gratuito que permite crear y editar ontologías en diversos formatos estándar como RDF(S), OWL y XML entre otros. Es modular y permite incorporar herramientas de visualización y razonamiento.

Teniendo en cuenta que el proyecto debe reutilizar las ontologías existentes en el tema de la Web de las cosas se decidió crear y adaptar ontologías de nivel superior, que posteriormente se reutilicen en ontologías de dominio específico y de tarea. Así será posible desarrollar ontologías de aplicación para el desarrollo de servicios en la WoT Figura 21.

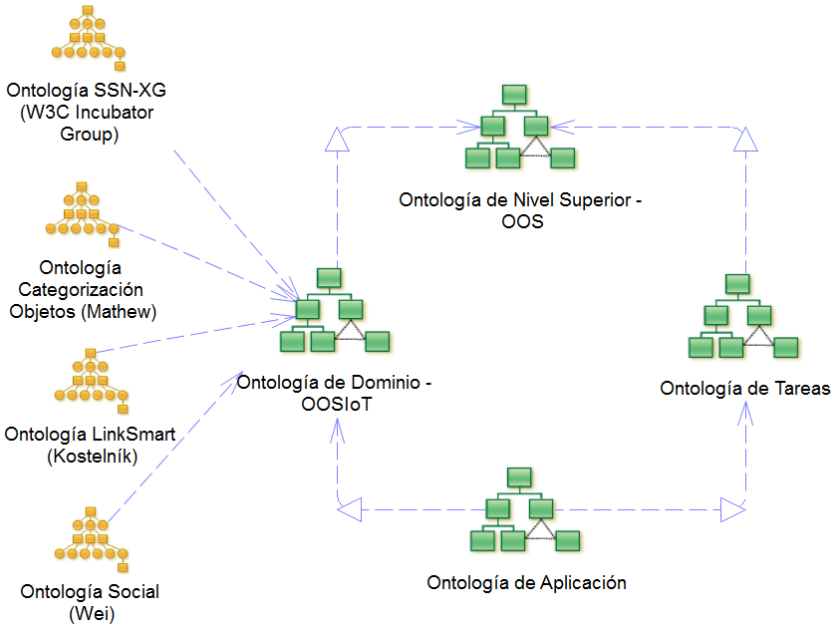


Figura 21. Modelo de Dependencia de Ontologías

Para este trabajo se propone una ontología de nivel superior que modela cualquier objeto denominada ontología objeto semántico – OOS. La cual define los elementos primordiales para que exista un objeto capaz de interactuar en cualquier contexto. Posteriormente, se alinearán o adaptarán las ontologías de la IoT relacionadas al presente trabajo (Kostelník, Sarnovský et al. 2011; Mathew, Atif et al. 2011; Biamino 2012; Compton, Barnaghi et al. 2012; Wei, De et al. 2012), creando una ontología de dominio llamada ontología de objeto semántico IoT – OOSIoT.
En el caso de estudio se crea una ontología de aplicación que adapta la OOSIoT a los requerimientos específicos y de tareas del dominio particular. Cada caso de estudio debe crear su propia ontología de aplicación para que pueda recopilar el conocimiento necesario para crear servicios adecuados.

En fase de reutilización de ontologías se aplicará el método propuesto por (Kavouras and Kokla 2000; Kavouras 2005) en el cual presentan la factorización semántica y los látices de conceptos como elementos para integrar ontologías y tres procesos de integración de las mismas.

5.7.5 Ubicación de los Productos

El producto “Modelo Semántico del Buscador BSIOT v8” metodología que se sigue para la implementación de la ontología para el buscador de contaminación medioambiental. Se encuentra en la siguiente ruta del CD de entrega:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 2\Actividad 1\Tarea 1

5.8 Fase II – Actividad 1: Tarea 2: Crear la estructura de conocimiento

El esta fase se llevó a cabo el plan de desarrollo de la estructura del conocimiento utilizando la herramienta PROTEGE, se construyó una ontología de Contaminación Ambiental alineando el modelo semántico con ontologías estandarizadas existentes como la (ENVO, Objeto Semántico, SSW-XG, AWS y CF-Features).

5.8.1 Ontología de Contaminación Ambiental

El producto de esta tarea es un documento OWL, primero se creó un modelo conceptual que fue realizado con la herramienta CMAP y luego fue adaptado con la herramienta PROTEGE.

5.8.2 Ubicación de los Productos

El producto “ontologiaambiental.owl” que contiene la ontología de Contaminación Ambiental. Se encuentra en la siguiente ruta del CD de entrega:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 2\Actividad 1\Tarea 2

5.9 Fase II – Actividad 2: Tarea 1: Crear los servicios del API de la Fuente de datos

En este apartado se desarrollaron los servicios web encargados de utilizar el portal de la IoT como fuente de datos, el portal seleccionado fue Xively, que sirve como repositorio de información de dispositivos que en tiempo real están subiendo información, la interfaz que dispone Xively, compartiendo servicios web de transferencia de estado representacional (Representational State Transfer – REST) los cuales puede ser invocados utilizando el protocolo HTTP por medio de línea de comandos. Los servicios REST para ser invocados a través de un servicio web REST cliente requiere conocer los métodos que implementaron y ya que no poseen metadatos WSDL no se puede crear un cliente REST que consuma el

servicio, pero REST ha sido pensado para poder ser consumido también por peticiones utilizando el protocolo HTTP los cuales fueron encapsuladas en métodos por el buscador semántico que implementa el índice y así lograr la comunicación con las interfaz REST de Xively. En el **¡Error! No se encuentra el origen de la referencia.**-**¡Error! No se encuentra el origen de la referencia.** se presentan la definición de las funciones encargadas de cumplir las dos fases funcionales Primera: Obtención de los documentos del portal web IoT Xively y crear una base documental. Segundo: Indexar los conceptos de dicha base documental.

5.9.1 Ubicación de los Productos

El producto “clases Xively.PNG” y “138mplementación servicios Xively.doc” que contiene que contiene el modelo de clase desarrollas para xively y la declaración de los métodos. Se encuentra en la siguiente ruta del CD de entrega:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 2\Actividad 2\Tarea 1

5.10 Fase II – Actividad 2: Tarea 2: Realimentación de la funcionalidad

En los servicios desarrollados y por motivos de alcance del proyecto se desarrollaron pruebas internas de funcionamiento, las cuales permitieron realimentar el método y los programas, por ello en la documentación entregada se encuentran las diferentes versiones por cada realimentación hecha.

5.11 Fase III – Actividad 1: Tarea 1: Extraer Información de Objetos

La implementación del índice semántico se realizarón dos maneras de indexación, una de ellas es crear extrayendo los documentos JSON directamente desde Xively lo que aumenta en gran medida el tiempo de creación del índice, el otro método es generar el índice a partir de la base de datos documental que previamente fue extarida desde el servicor Xively en una indexación previa, el tiempo de creación disminuye drásticamente.

5.11.1 Ubicación de los Productos

El producto “clases Xively.PNG” y “implementación servicios Xively.doc” que contiene que contiene el modelo de clase desarrollas para Xively y la declaración de los métodos. Se encuentra en la siguiente ruta del CD de entrega:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 2\Actividad 2\Tarea 1

5.12 Fase III – Actividad 1: Tarea 2, 3 y 4: Análisis Léxico, Eliminación de Palabras vacías y Aplicación de Lematización

Se implementó un módulo, el cual usando los servicios web del API Xively, hace un escaneo sistemático de toda la base de datos de Xively, extrayendo la información de cada dispositivo que coincide con los conceptos almacenados en la ontología desarrollada. Con esto se crea la base de datos documental - BBD, la cual es un directorio que está compuesto por documentos en formato JSON con el nombre del FEED que es el identificador único que tiene Xively a cada documento. En el proceso de extracciópn se van anotando los JSON con respecto a los conceptos encontrados en su contenido.

Las Tarea 2, Tarea 3 y Tarea 4 corresponde directamente a la segunda fase de funcionalidad que es procesamiento de los documentos que fueron recopilados de Xively, es decir se procesan los documentos para crear el índice semántico.

Una vez ya se tienen los documentos en la base documental se procede a hacer el análisis léxico. El formato en el que se obtuvo los documentos es con JSON, como este ya es un formato estructurado entonces se va extrayendo la información estructurada y se analiza. Se toma el documento asociado a un FEED y se extrae la información del documento analizando los documentos. Los procesos principales son realizados a tráves de la herramienta LUCENE, la cual al momento de realizar la indexación del documento particular realiza el análisis léxico, la eliminación de palabras vacías y aplicación de Lematización, dependiendo del analizador relacionado (español, inglés).

Una vez recopilados los documentos y analizados el documento ahora si se pueden indexar. Este apartado requiere proveer una clase de realice la funcionalidad en español la cual se creó directamente de cero. Este paso constituye un aporte adicional en la aplicación ya que se implementó no solamente el analizador en inglés existente en Lucene, sino también el analizador en español, con el objetivo de adecuar la herramienta a lenguajes natural castellano para ampliar el espectro de usuarios que puedan utilizar la misma.

5.12.1 Ubicación de los Productos

La base de datos documental se encuentra en la siguiente ruta del CD de entrega:
C:\CD
del
Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\BuscadorSemantico\WSSemanticSearch\App_Data\Json_Data.

Adicionalmente, se creo un documento de nombre: “Preprocesamiento de la Base de datos Documental.doc” en el cual se puede precisar al método de extarcción y anotación desarrollado:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 3\Actividad 1

Se pueden consultar también las líneas de código encargadas de procesar los documentos se encuentra con el nombre de “Colocar el codigo que extrae los documentos.txt” que contiene el contenido del archibo de nombre “WSSemanticSearch.asmx.cs” de Visual Studio ubicado en:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 3\Actividad 1\Tarea 1

5.13 Fase III – Actividad 2 – Tareas 1, 2 y 3 – Selección de Términos, Obtención de los Conceptos, Desambiguar Conceptos

5.13.1 Introducción

En esta actividad se hace una selección de términos candidatos, es decir, los términos relevantes en los documentos y que son candidatos para extraer sus conceptos. Estos conceptos aparecerán a partir de un proceso de consulta en las estructuras de información diseñadas para almacenar conocimiento en la primera fase, como los tesauros y las ontologías u otras aproximaciones de la teoría cognitiva

5.13.2 Extracción de Conceptos

5.13.2.1 Selección de Términos

Dependiendo de los objetivos del índice se deben realizar algunas funcionalidades, en este apartado se hace algo diferente normalmente la indexación se realiza tomando los documentos anotarlo con los términos, sino que toman los documentos y se anotan con los conceptos relacionados a la ontología de contaminación ambiental y por lo tanto fue recuperado desde el servidor. El resultado es una lista de conceptos relacionados al JSON o al sensor recopilado.
Por una decisión de diseño no se aplicaron Frases Nominales pero si se anota el concepto con frases nominales, es decir no se tomó cada concepto del documento y se busca en la ontología sino lo que se realizo fue tomar el concepto de la ontología y se estableció que documento lo contiene y por último se anota.

5.13.2.2 Obtención de los Conceptos

Con la lista de conceptos relacionados a cada documento JSON se procede a realizar indexación con la librería de LUCENE, esta herramienta utiliza el modelo de Espacio Vectorial. En el directorio de productos se muestra el método que implementa la indexación de los documentos. El producto final es una lista de conceptos que fueron anotados en el JSON en la etapa anterior.

5.13.2.3 Desambiguar Conceptos

De acuerdo al método de anotación elegido no hay necesidad de desambiguar los conceptos ya que la lista de conceptos anotados son tomados directamente de la ontología en un dominio particular y específico y por otro lado la librería de Lucene se encarga de hacer un proceso similar al momento de indexar el documento con los términos de los campos abiertos.

5.14 Fase III – Actividad 3 – Tareas 1, 2 y 3 – Análisis de Coocurrecia y Análisis de Representatividad

5.14.1 Análisis de coocurrencia

Este proceso se decidió utilizar la similitud semántica que ofrece el modelo vectorial, con el fin de reutilizar la librería Lucene, la cual ya internamente desarrolla dicho modelo con el método ***MoreLikeThis***. Sin embargo, aquí se puede utilizar cualquier otro modelo o algoritmo personalizado y en este caso los desarrolladores debería utilizar otra librería o crear sus propios algoritmos.

5.14.1.1 Análisis de Representatividad

El análisis de representatividad se realizó con el mismo método de modelo vectorial, Sin embargo, el hecho de anotar los documentos con listas de términos, hace que este modelo pondere más alto los documentos que más conceptos tengan relacionados en la ontología. Los resultados son muy precisos para búsquedas que se relacionan directamente con el dominio particular, pero sacrifica las búsquedas que no encuentran coincidencias con los conceptos de la ontología, por esta razón a veces los lugares geográficos salen varias posiciones después en la salida del buscador.

1. Codigo Fuente del Análisis de Representatividad

```
       //Realiza la búsqueda de un texto "original" contra los documentos indexados utilizando TFIDF

4              public  static  List<Document>  moreLikeThisAnalyzer(String  original,  ISet<string>  stopWords,
       Lucene.Net.Analysis.Analyzer analyzer)
5              {
6                  Trace.WriteLine("Realizando la Búsqueda");
7                  List<Document> DocumenResult = new List<Document>();
8
9                  IndexReader indexReader = IndexReader.Open(_directory, true);
```

```
10                IndexSearcher indexSearcher = new IndexSearcher(indexReader);
11
12                MoreLikeThis mlt = new MoreLikeThis(indexReader);
13
14                mlt.SetFieldNames(DEFAULT_FIELD_NAMES);
15                mlt.MinDocFreq = DEFALT_MIN_DOC_FREQ;
16                mlt.MinTermFreq = DEFAULT_MIN_TERM_FREQ;
17                mlt.MaxQueryTerms = MAX_QUERY_TERMS;
18                mlt.MinWordLen = DEFAULT_MIN_WORD_LENGTH;
19                mlt.Analyzer = analyzer;
20                mlt.SetStopWords(stopWords);
21
22                Query query = mlt.Like(new System.IO.StringReader(original));
23
24                int topCount = DEFAULT_DOCUMENT_TO_SEARCH;
25
26                TopScoreDocCollector collector = TopScoreDocCollector.Create(topCount, true);
27                indexSearcher.Search(query, collector);
28                ScoreDoc[] hits = collector.TopDocs().ScoreDocs;
29                var result = new List<string>();
30                //Hits hits = indexSearcher.Search(query);
31
32                int len = hits.Length;
33
34                Trace.WriteLine("Entering");
35                Trace.WriteLine("------------------------------------------");
36                Trace.WriteLine("original :" + original);
37                Trace.WriteLine("query: " + query);
38                Trace.WriteLine("found: " + len + " documents");
39                for (int i = 0; i < Math.Min(25, len); i++)
40                {
41                    int d = hits[i].Doc;
42                    Trace.WriteLine("score   : " + hits[i].Score);
43                    Trace.WriteLine("name    : " + d.ToString());
44                    //Colocar los datos en el arreglo de resultados
45                    Document doc = indexSearcher.Doc(hits[i].Doc);
46                    DocumenResult.Add(doc);
47                }
48                Trace.WriteLine("------------------------------------------");
49                Trace.WriteLine("Exiting");
50                return DocumenResult;
51            }
52
```

Ubicación de los Productos

Esta es la ubicación donde se encuentra el producto correspondiente a toda la actividad 2 en el CD de entrega con el nombre de “Extracción de conceptos.doc”:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\Modelo Semántico Indice IoT\Fase 3\Actividad 2

5.15 Fase III – Actividad 4: Tarea 1: Implementación del Índice Semántico

Ya que la creación del índice se realizó con la librería de LUCENE, estos módulos se encuentran en la implementación interna de la misma.

Ubicación de los Productos

LUCENE utiliza arboles invertidos y los archivos del índice se pueden encontrar en la ruta del CD:

C:\CD del Proyecto\IndiceSemanticoIoT\BuscadorSemanticoIoT\BuscadorSemantico\WSSemanticSearch\SemanticIndex

5.16 Fase III – Actividad 4: Tarea 2: Implementación de Servicios Web Semántico de Objetos

Para el caso particular se creó un conjunto de servicios Web que exponen toda la funcionalidad del índice semántico y teniendo en cuenta el dominio particular y el diseño realizado del modelo semántico de la fase 1 del desarrollo del índice.

2. Codigo Fuente de Implementación de Servicios Web Semántico de Objetos

```
4       ///<summary>
5           ///Esta función recibe un conjunto de caracteres a los cuales les hace un proceso de ampliación de consulta y
6           ///posteriormente busca en el índice las coincidencias con respecto a toda la información que ha sido encontrada
7           ///en el repositorio de información.
8           ///</summary>
9           ///<returns>
10          ///DataSet con una lista de Sesnsores que cumplen con la consulta realizada. Los datos no tienen información de
        datastreams
11          ///porque la información es recuperada directamente del índice
12          /// </returns>
13          /// <param name="consulta">Texto en lenguaje natural con la necesidad de información del usuario</param>
14          /// <param name="idioma">Idioma preferido de consulta seleccionado por el usuario</param>
15          public DataSet Buscar(string consulta, string idioma)
16          {
17              return srvIndex.Buscar(consulta, idioma);
18          }
19
20          ///<summary>
21          ///Esta función recibe un conjunto de caracteres a los cuales les hace un proceso de ampliación de consulta y
22          ///posteriormente busca en el índice las coincidencias con respecto a toda la información que ha sido encontrada
23          ///en el repositorio de información.
24          ///</summary>
25          ///<returns>
26          ///Liata de FeedXively o lista de Sesnsores que cumplen con la consulta realizada. Adiferecncia de Buscar
27          ///este procedimiento retorna todos los metadatos de Feed incluyendo dtastreams.
28          /// </returns>
29          /// <param name="consulta">Texto en lenguaje natural con la necesidad de información del usuario</param>
30          /// <param name="idioma">Idioma preferido de consulta seleccionado por el usuario</param>
31          public List<FeedXively> BuscarFeeds(string consulta, string idioma)
32          {
33              FeedXively[] feedstmp = srvIndex.BuscarFeeds(consulta, idioma);
34              //List<FeedXively> feedtmplist = new List<FeedXively>();
35              //feedtmplist.AddRange(feedstmp);
36              return feedstmp.Cast<FeedXively>().ToList();
37              //return feedtmplist;
38          }
39
40          ///<summary>
41          ///Esta función recibe la consulta y la expande con respecto a la ontologia cargada en el servicio. Para ello
42          ///Identifica de los términos de la copnsulta los conceptos y extrae sus conceptos hijos o relacionados de primer nivel.
43          ///Los conceptos que no encuentra, hace una expanción de sinonimos con wordnet y finalmente añade los términos no
44          ///identificados al final de la consulta. Con los conceptos identificados realiza un procesos de similitud semantica
45          ///para establecer su importancia en la consulta.
46          ///</summary>
47          ///<returns>
48          ///Cadena con la consulta expandida en un formato más accesible a los buscadores.
49          /// </returns>
50          /// <param name="consulta">Corresponde a la consulta original del usuario</param>
51          /// <param name="idioma">Idioma de preferencia del usuario. Por defecto busca Español</param>
52          public string ExpandirConsulta(string consulta, string idioma)
53          {
54              return srvIndex.ExpandirConsultaConceptosOntologia(consulta, idioma);
55          }
56
57          ///<summary>
58          ///Esta función recibe un Biotipo(lugar) en términos de latitud y longitud, una consulta (buscar)  y devuelve la
        información
59          ///relacionada a los sensores que tienen influencia en dicho lugar (determinado por el radio) y sus relaciones a
60          ///conceptos medioambientales. El idioma y el radio son opcionales. Por defecto busca en: Español y 100 km.
61          ///</summary>
```

```
62          ///<returns>
63          ///DataSet con una lista de Sensores que estan en el área de onfluencia circular del biotipo
64          /// </returns>
65          /// <param name="lat">Latitud del lugar en el cual se buscarna los sensores</param>
66          /// <param name="lng">Longitud del lugar en el cual se buscarna los sensores</param>
67          /// <param name="dtsResultados">Dataset con los sensores que se encontraron en la consulta del usuario</param>
68          /// <param name="idioma">Idioma de preferencia del usuario. Por defecto busca Español</param>
69          /// <param name="radio">Área circular al rededor del punto geografico enviado en kilometros en los cuales se buscaran
        los sensores</param>
70          public List<FeedXively> RetornarMapaLugar(string lat, string lng, string consulta, string idioma, string radio)
71          {
72              List<FeedXively> feedList = new List<FeedXively>();
73
74              feedList.AddRange(srvIndex.RetornarMapaLugar(Convert.ToDouble(lat, culture),
75                                                  Convert.ToDouble(lng, culture),
76                                                  consulta, idioma, Convert.ToDouble(radio, culture)));
77              return feedList;
78          }
79
80          ///<summary>
81          ///Esta función recibe un Biotipo(lugar) en términos de latitud y longitud, una consulta (buscar)   y devuelve la
        información
82          ///relacionada a los sensores que tienen influencia en dicho lugar (determinado por el radio) y sus relaciones a
83          ///conceptos medioambientales. El idioma y el radio son opcionales. Por defecto busca en: Español y 100 km.
84          ///La variante con respecto al anteriopr es que busca en Xively los datapoints
85          ///</summary>
86          ///<returns>
87          ///DataSet con una lista de Sensores que estan en el área de onfluencia circular del biotipo
88          /// </returns>
89          /// <param name="lat">Latitud del lugar en el cual se buscarna los sensores</param>
90          /// <param name="lng">Longitud del lugar en el cual se buscarna los sensores</param>
91          /// <param name="consulta">Consulta del usuario</param>
92          /// <param name="fechaFin">Fecha inicial del rango de datapoint a recuperar</param>
93          /// <param name="fechaInicio">FEcha Final del rango de datapoint a recuperar</param>
94          /// <param name="idioma">Idioma de preferencia del usuario. Por defecto busca Español</param>
95          /// <param name="radio">Área circular al rededor del punto geografico enviado en kilometros en los cuales se buscaran
        los sensores</param>
96          public  List<FeedXively>  RetornarMapaLugarDatapoints(string  lat,  string  lng,  string  consulta,  DateTime  fechaInicio,
        DateTime fechaFin, string idioma, string radio)
97          {
98              List<FeedXively> feedList = new List<FeedXively>();
99
100             srvIndex.Timeout = -1;
101             feedList.AddRange(srvIndex.RetornarMapaLugarDatapoints(Convert.ToDouble(lat, culture),
102                                                                    Convert.ToDouble(lng, culture),
103                                                                    consulta,
104                                                                    fechaInicio,
105                                                                    fechaFin,
106                                                                    idioma,
107                                                                    Convert.ToDouble(radio, culture)));
108             return feedList;
109         }
110
111         ///<summary>
112         ///Esta función recibe un Biotipo(lugar) en términos de latitud y longitud, una lista de sensores   y devuelve la
        información
113         ///relacionada a los sensores que tienen influencia en dicho lugar (determinado por el radio) y sus relaciones a
114         ///conceptos medioambientales. El idioma y el radio son opcionales. Por defecto busca en: Español y 100 km.
115         ///</summary>
116         ///<returns>
117         ///DataSet con una lista de Sensores que estan en el área de onfluencia circular del biotipo
118         /// </returns>
119         /// <param name="lat">Latitud del lugar en el cual se buscarna los sensores</param>
120         /// <param name="lng">Longitud del lugar en el cual se buscarna los sensores</param>
121         /// <param name="dtsResultados">Dataset con los sensores que se encontraron en la consulta del usuario</param>
122         /// <param name="idioma">Idioma de preferencia del usuario. Por defecto busca Español</param>
123         /// <param name="radio">Área circular al rededor del punto geografico enviado en kilometros en los cuales se buscaran
        los sensores</param>
124         public List<FeedXively> RetornarMapaLugarListaSensores(string lat, string lng, List<FeedXively> feedList, string idioma,
        string radio)
125         {
126             FeedXively[] feedsarr = feedList.ToArray();
127             feedsarr = srvIndex.RetornarMapaLugarListaSensores(Convert.ToDouble(lat, culture),
128                                                                Convert.ToDouble(lng, culture),
129                                                                feedsarr, idioma,
130                                                                Convert.ToDouble(radio, culture));
131             return feedsarr.Cast<FeedXively>().ToList();
132         }
133
134         ///<summary>
135         ///Esta función busca los Biotipos realcionados a una lista de sensores encontrados en una busqueda realizada al indice.
136         ///</summary>
137         ///<returns>
138         ///DataSet con una lista de Sesnsores que cumplen con la consulta realizada
139         /// </returns>
140         /// <param name="ListaSensoresEncontrados">Lista de sensores encontrados despues de la consulta de usuario</param>
141         public List<GeonameNode> ObtenerBiotiposConsulta(List<GeonameNode> ListaSensoresEncontrados)
142         {
143             //Serializar la lista de GeonameNodes
144             string xmlListGeonodes = ListaSensoresEncontrados.SerializarToXml();
145
146             //Obtenemos la División Administrativa de Primer orden para la lsita de sensores
```

```
147             //que tienene localización
148             srvIndex.Timeout = -1;
149             xmlListGeonodes = srvIndex.ObtenerListaLugarCoordenadas_AM1(xmlListGeonodes);
150 
151             //Desserializamos los objetos retornados
152             List<GeonameNode> gnCercanos = xmlListGeonodes.DeserializarTo<List<GeonameNode>>();
153 
154             return gnCercanos;
155         }
156 
157         ///<summary>
158         ///Esta función recibe un término y lo compara en la ontología con el fin de establecer si es un concepto.
159         ///Si es concepto retorna los conceptos más relacionados al mismo (relaciones directas a otros conceptos),
160         ///desde el punto de vista del dominio de la ontología cargada.
161         ///</summary>
162         ///<returns>
163         ///Lista de conceptos relacionados semánticamente al concepto buscado
164         /// </returns>
165         /// <param name="concepto">Conncepto que se esta buscando en la ontología</param>
166         /// <param name="idioma">Idioma de preferencia del usuario.</param>
167         public List<string> RetornarConceptos(string concepto, string idioma)
168         {
169             string res = srvIndex.RetornarConceptos(concepto, idioma);
170             return res.DeserializarTo<List<string>>();
171         }
172 
173         ///<summary>
174         ///Esta función retorna un FeedXively con lod datapoints correspondientes entre un rago de tiempo
175         ///</summary>
176         ///<returns>
177         ///Feed completo
178         /// </returns>
179         /// <param name="concepto">Conncepto que se esta buscando en la ontología</param>
180         /// <param name="idioma">Idioma de preferencia del usuario.</param>
181         public FeedXively HistoricosSensor(string feedID, DateTime fechaInicio, DateTime fexhaFin)
182         {
183             FeedXively fx = new FeedXively();
184 
185             fx = srvIndex.RetornarDatosSensor(feedID, fechaInicio, fexhaFin);
186 
187             return fx;
188         }
189 
190         //Aqui vienen los demás servicios del negocio
191         //5.     TendenciaVariableAmbiental (variable medioambiental): La tendencia debe implementar un sistema de análisis correlacional básico con el fin de proveer información de tendencias de variables. Se utilizará lo más básico pero aquí se pueden colocar análisis más elaborados que se dejan para futuros estudios.
192         //6.     VariablesDiferentesBiotipo (lugar): Este método debe comparar los valores de referencia almacenados en el Biotipo de ciertos lugares y posteriormente con las mediciones de los sensores establecer si se encuentran en los rangos establecidos o se han salido de sus valores esperados. Para esto hay que buscar la información de referencia y poblar la ontología con la misma en un lugar seleccionados para demostrar sus posibilidades de uso.
193         //7.     CausandoDañoAmbiental (lugar): Este método devuelve la información de los posibles agentes contaminantes que están afectando cierto lugar. Para esto debe tomar los sensores de dicho lugar y posteriormente utilizar la función de VariablesDiferentesBiotipo, las variable identificadas se relacionan a sus agentes contaminantes y se retorna una respuesta en una estructura de conceptos provista por la ontología en la cual se identifican dichos contaminantes.
194         //8.     DañosAmbientales (lugar, AgentesContaminantes): A partir de un conjunto de agentes contaminantes y su magnitud se desarrolla un modelo matemático sencillo para relacionarlo a un daño medioambiental. La respuesta son los posibles daños causados el ecosistema por estos agentes.
195 
```

3. Código Fuente de Implementación de Servicios de Índice

```
1.          #region "Servicios Básicos del Indice semántico"
2.  
3.          //Realiza la Búsqueda de la consulta dada, con la ontología y las opciones de configuración
4.          //realizadas previamente y almacenada en el Web.config
5.          [WebMethod]
6.          public DataSet Buscar(string stringBuscar, string idioma = "Español")
7.          {
8.              if (!string.IsNullOrEmpty(stringBuscar))
9.                  return SearchSemanticIndex(stringBuscar, idioma);
10.             else
11.                 return null;
12.         }
13.
14.         //Igual que la amterior pero no expande consulta y retorna una lista de FeedXively Serializados
15.         [WebMethod]
16.         public List<FeedXively> BuscarFeeds(string consulta, string idioma = "Español")
17.         {
18.             if (!string.IsNullOrEmpty(consulta))
19.             {
20.                 //Crea el Objeto que guarda todo el conocimiento del negocio
```

```
21.                 SemanticIndexManager semanticIndexManager = new SemanticIndexManager();
22.
23.                 //Crea la lista de documentos que serán el resultado de la búsqueda
24.                 List<UrlDocument> allReponse = new List<UrlDocument>();
25.
26.                 //Se busca en español o el ingles. Si es nulo busca por ambos.
27.                 if (idioma == "Español")
28.                 {
29.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true);
30.                 }
31.                 else if (idioma == "Ingles")
32.                 {
33.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, false);
34.                 }
35.                 else
36.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true); //Por defecto
    busca español
37.
38.                 //Armar la estructura de datos a retornar como resltado
39.                 List<FeedXively> SensoresResult = new List<FeedXively>();
40.
41.                 //Traemos todos los metadatos del sensor que estan almacenados en la BDD local
42.                 foreach (UrlDocument urldoc in allReponse)
43.                 {
44.                     FeedXively xtemp = semanticIndexManager.ObtenerJSONFeedBDD(urldoc.Id);
45.                     SensoresResult.Add(xtemp);
46.                 }
47.
48.                 return SensoresResult;
49.             }
50.             else
51.                 return null;
52.         }
53.
54.         //Igual que la anterior pero retorna una lista de FeedXively Serializados
55.         //Además de los datapoints correspondientes consultados en Xively.
56.         [WebMethod]
57.         public List<FeedXively> BuscarFeedsDataPoints(string consulta, DateTime fechaInicio, DateTime
    fechaFin, string idioma = "Español")
58.         {
59.             if (!string.IsNullOrEmpty(consulta))
60.             {
61.                 //Crea el Objeto que guarda todo el conocimiento del negocio
62.                 SemanticIndexManager semanticIndexManager = new SemanticIndexManager();
63.
64.                 //Crea la lista de documentos que serán el resultado de la búsqueda
65.                 List<UrlDocument> allReponse = new List<UrlDocument>();
66.
67.                 //Se busca en español o el ingles. Si es nulo busca por ambos.
68.                 if (idioma == "Español")
69.                 {
70.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true);
71.                 }
72.                 else if (idioma == "Ingles")
73.                 {
74.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, false);
75.                 }
76.                 else
77.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true); //Por defecto
    busca español
78.
79.                 //Armar la estructura de datos a retornar como resltado
80.                 List<FeedXively> SensoresResult = new List<FeedXively>();
81.
82.                 //Traemos todos los metadatos del sensor qdesde le servidor Xively, junto con sus mediciones
83.                 foreach (UrlDocument urldoc in allReponse)
84.                 {
85.                     FeedXively xtemp = RetornarDatosSensor(urldoc.Id, fechaInicio, fechaFin);
86.                     SensoresResult.Add(xtemp);
87.                 }
88.
89.                 return SensoresResult;
90.             }
91.             else
92.                 return null;
93.         }
94.
95.         //Retorna un listado de conceptos directamente relacionados al concepto. Estos de obtienen
96.         //de la ontología almacenada. Descarta términos no encontrados y no busca en wordnet.
```

```
97.         //Si el concepto no esta en la ontología retorna una cadena vacia.
98.         [WebMethod]
99.         public string RetornarConceptos(string Concepto, string idioma = "Español")
100.        {
101.            return BuscarConceptosOntologia(Concepto, idioma);
102.        }
103.
104.        //Retorna un listado de sensores y su localización con respecto a la localización dada y una Consulta
105.        [WebMethod]
106.        public List<FeedXively> RetornarMapaLugar(double Latitud, double Longitud, string Consulta, string
    idioma = "Español", double radio = 100)
107.        {
108.            string expansion = ObtenerConfig("utilizarExpansion");
109.
110.            //Crea el Objeto que guarda todo el conocimiento del negocio
111.            semanticIndexManager = new SemanticIndexManager();
112.
113.            //Crea una copia de la lista para eliminar los sensores que no se les puede calcular distancia
114.            List<FeedXively> FeedsConPosicion = new List<FeedXively>();
115.
116.            //Lista de los resultados de la busqueda
117.            List<FeedXively> dtsResult = new List<FeedXively>();
118.
119.            //Buscar la consulta en el índice
120.            if (!string.IsNullOrEmpty(Consulta))
121.                dtsResult = BuscarFeeds(Consulta, idioma);
122.            else
123.                return null;
124.
125.            //Si encontro sensores entonces validar los que entran en el radio de acción
126.            if(dtsResult != null)
127.            {
128.                List<string> clavesBorrar = new List<string>();
129.
130.                foreach (FeedXively dtr in dtsResult)
131.                {
132.                    if (dtr.feed.location != null)
133.                    {
134.                        //La funcion establece que el sensor este en el rango de N kms del lugar
    especificado. Si esta devuelve la distancia, sino devuelve cero
135.                        double distancia = semanticIndexManager.VerificarSensorenLugar(Latitud,
136.                                                    Longitud, Convert.ToDouble(dtr.feed.location.lat,
    culture),
137.                                                    Convert.ToDouble(dtr.feed.location.lon, culture),
    radio);
138.                        if (distancia > 0)
139.                        {
140.                            //ListaResultados.Add(indice, distancia);
141.                            GeonameNode nodolugardistancia = new GeonameNode();
142.                            nodolugardistancia.Latitud = Latitud.ToString();
143.                            nodolugardistancia.Longitud = Longitud.ToString();
144.                            lugar ld = new lugar();
145.                            ld.nodoGeoname = nodolugardistancia;
146.                            ld.distancia = distancia;
147.                            //Creamos el espacio en la variable de lista de lugares relacionada al sensor
148.                            dtr.lugares = new List<lugar>();
149.                            dtr.lugares.Add(ld);
150.                            FeedsConPosicion.Add(dtr);
151.                        }
152.                    }
153.                }
154.            }
155.
156.            //Finalmente retorna la lista con los sensores correspondientes
157.            return FeedsConPosicion;
158.        }
159.
160.        //Retorna un listado de sensores y su localización con respecto a la localización dada y una Consulta
161.        //Adicional retorna los datapoints de la fechas dadas
162.        [WebMethod]
163.        public List<FeedXively> RetornarMapaLugarDatapoints(double Latitud, double Longitud, string Consulta,
    DateTime fechaInicio, DateTime fechaFin, string idioma = "Español", double radio = 100)
164.        {
165.            string expansion = ObtenerConfig("utilizarExpansion");
166.
167.            //Crea el Objeto que guarda todo el conocimiento del negocio
168.            semanticIndexManager = new SemanticIndexManager();
```

```
169.
170.             //Crea una copia de la lista para eliminar los sensores que no se les puede calcular distancia
171.             List<FeedXively> FeedsConPosicion = new List<FeedXively>();
172.
173.             //Lista de los resultados de la busqueda
174.             List<FeedXively> dtsResult = new List<FeedXively>();
175.
176.             //Buscar la consulta en el índice
177.             if (!string.IsNullOrEmpty(Consulta))
178.                 dtsResult = BuscarFeedsDataPoints(Consulta,fechaInicio, fechaFin, idioma);
179.             else
180.                 return null;
181.
182.             //Si encontro sensores entonces validar los que entran en el radio de acción
183.             if (dtsResult != null)
184.             {
185.                 List<string> clavesBorrar = new List<string>();
186.
187.                 foreach (FeedXively dtr in dtsResult)
188.                 {
189.                     if (dtr.feed.location != null && dtr.feed.location.lat != null && dtr.feed.location.lon
    != null)
190.                     {
191.                         //La funcion establece que el sensor este en el rango de N kms del lugar
    especificado. Si esta devuelve la distancia, sino devuelve cero
192.                         double distancia = semanticIndexManager.VerificarSensorenLugar(Latitud,
193.                                                     Longitud, Convert.ToDouble(dtr.feed.location.lat,
    culture),
194.                                                     Convert.ToDouble(dtr.feed.location.lon, culture),
    radio);
195.                         if (distancia > 0)
196.                         {
197.                             //ListaResultados.Add(indice, distancia);
198.                             GeonameNode nodolugardistancia = new GeonameNode();
199.                             nodolugardistancia.Latitud = Latitud.ToString();
200.                             nodolugardistancia.Longitud = Longitud.ToString();
201.                             lugar ld = new lugar();
202.                             ld.nodoGeoname = nodolugardistancia;
203.                             ld.distancia = distancia;
204.                             //Creamos el espacio en la variable de lista de lugares relacionada al sensor
205.                             dtr.lugares = new List<lugar>();
206.                             dtr.lugares.Add(ld);
207.                             FeedsConPosicion.Add(dtr);
208.                         }
209.                     }
210.                 }
211.             }
212.
213.             //Finalmente retorna la lista con los sensores correspondientes
214.             return FeedsConPosicion;
215.         }
216.
217.         //Retorna un listado de Sensores y su localización con respecto a una localización dada y un conjunto
    se sensores
218.         [WebMethod]
219.         public List<FeedXively> RetornarMapaLugarListaSensores(double Latitud, double Longitud,
    List<FeedXively> dtsResult, string idioma = "Español", double radio = 100)
220.         {
221.             //Crea el Objeto que guarda todo el conocimiento del negocio
222.             semanticIndexManager = new SemanticIndexManager();
223.
224.             //Crea una copia de la lista para eliminar los sensores que no se les puede calcular distancia
225.             List<FeedXively> FeedsConPosicion = new List<FeedXively>();
226.
227.             //Si encontro sensores entonces validar los que entran en el radio de acción
228.             if (dtsResult != null)
229.             {
230.                 List<string> clavesBorrar = new List<string>();
231.
232.                 foreach (FeedXively dtr in dtsResult)
233.                 {
234.                     if (dtr.feed.location != null)
235.                     {
236.                         //La funcion establece que el sensor este en el rango de N kms del lugar
    especificado. Si esta devuelve la distancia, sino devuelve cero
237.                         double distancia = semanticIndexManager.VerificarSensorenLugar(Latitud,
238.                                                     Longitud, Convert.ToDouble(dtr.feed.location.lat,
    culture),
239.                                                     Convert.ToDouble(dtr.feed.location.lon, culture),
```

```
    radio);
240.                         if (distancia > 0)
241.                         {
242.                             //ListaResultados.Add(indice, distancia);
243.                             GeonameNode nodolugardistancia = new GeonameNode();
244.                             nodolugardistancia.Latitud = Latitud.ToString();
245.                             nodolugardistancia.Longitud = Longitud.ToString();
246.                             lugar ld = new lugar();
247.                             ld.nodoGeoname = nodolugardistancia;
248.                             ld.distancia = distancia;
249.                             //Creamos el espacio en la variable de lista de lugares relacionada al sensor
250.                             dtr.lugares = new List<lugar>();
251.                             dtr.lugares.Add(ld);
252.                             FeedsConPosicion.Add(dtr);
253.                         }
254.
255.                     }
256.                 }
257.             }
258.
259.             //Finalmente retorna la lista con los sensores correspondientes
260.             return FeedsConPosicion;
261.         }
262.
263.         //Realiza un proceso de expanción de la consulta del usuario utilizando la ontología para hacerla más
    exacta
264.         //Para esto primero Retorna conceptos, posteriormente busca en wordnet conceptos no encontrados y
    finalmente
265.         //enlaza los términos no identificados al final de la consulta por considerarse importantes para el
    usuario.
266.         [WebMethod]
267.         public string ExpandirConsultaConceptosOntologia(string Consulta, string idioma)
268.         {
269.             return ExpandirConsulta(Consulta, idioma);
270.         }
271.         #endregion
272.
273.         #region "Servicios de Configuración del Indice Semántico"
274.
275.         //Método Adicional: Permite recuperar los valores de las claves almacenadas en el Web.Config
276.         //con el fin de establecer la configuración actual del índice
277.         [WebMethod]
278.         public string ObtenerConfiguracion(string key)
279.         {
280.             if (!string.IsNullOrEmpty(key))
281.                 return ObtenerConfig(key);
282.             else
283.                 return "La clave no puede ser vacia o nula";
284.         }
285.
286.         //Método Adicional: Permite asignar un valor a una clave específica en el Web.config con el fin
287.         //de establecer una nueva configuración de indexción
288.         [WebMethod]
289.         public string AsignarConfiguracion(string key, string valor)
290.         {
291.             if (!string.IsNullOrEmpty(key) && !string.IsNullOrEmpty(key))
292.                 return AsignarConfig(key, valor);
293.             else
294.                 return "La clave o el valor no pueden ser nulos o vacios";
295.         }
296.
297.         //Método Adicional: Una vez se han cambiado las configuraciones o se desea inicia una nueva ontología
    de dominio
298.         //Es necesario crear el índice semántico. Este método crea el índice teniendo en cuenta si lo hace
    desde el servidor
299.         //de objetos Xively o desde los archivos locales almacenados en una indexación previa.
300.         [WebMethod]
301.         public string CrearIndiceSemantico()
302.         {
303.             //Crea el Objeto que guarda todo el conocimiento del negocio
304.             semanticIndexManager = new SemanticIndexManager();
305.             if (ObtenerConfig("BDDfuente") == "Xively")
306.             {
307.                 semanticIndexManager.CrearIndiceSemantico(true);
308.                 return "El índice se ha creado correctamente desde el servidor Xively.";
309.             }
310.             else
```

```
311.             {
312.                 semanticIndexManager.CrearIndiceSemantico(false);
313.                 return "El índice se ha creado correctamente desde la Base de Datos Local de Archivos JSON.";
314.             }
315.         }
316.
317.         //Método Adicional: Permite cambiar y cargar los conceptos a la BD del servicio de una nueva
    ontología.
318.         //Por defecto crea nuevamente el índice desde el servidor de objetos Xively.
319.         [WebMethod]
320.         public string CargarOntologiaeIndexar(string nuevaontologia)
321.         {
322.             string NombreArchivoNuevaOntologia = Path.GetFileName(nuevaontologia);
323.             string camino = SemanticIndexManager.RutaOntologiaSinArchivo;
324.             string conceptoscargados = string.Empty;
325.
326.             //Verificar ue el archivo de la ontologia existe
327.             if (File.Exists(nuevaontologia))
328.             {
329.                 //Copia la ontologia al directorio de ontologías
330.                 File.Copy(nuevaontologia, camino + NombreArchivoNuevaOntologia, true);
331.                 //Se debe cargar los conceptos de la ontologia para el proceso de expansión de consulta
332.                 conceptoscargados = CargarConceptosOntologia(nuevaontologia);
333.
334.                 //Si el proceso fue Exitoso se cambia la nueva ruta de la Ontología
335.                 AsignarConfig("FileOntology", NombreArchivoNuevaOntologia);
336.
337.                 //Recupera el objeto de logica de negoicio de la aplicación para que el cambio de ontologia
    tenga efecto
338.                 semanticIndexManager = new SemanticIndexManager();
339.
340.                 //Finalmente Crea el índice semántico Nuevamente desde Xively
341.                 semanticIndexManager.CrearIndiceSemantico(true);
342.
343.                 //Retorna los conceptos cargados
344.                 return conceptoscargados;
345.             }
346.             return "ERROR - La ruta del archivo de la ontología no es correcta o el archivo no existe";
347.         }
348.
349.         //Método Adicional: Permite cargar un feed de Xively espécifico a la BDD
350.         //Posteriormente lo indexa en el indice semántico
351.         [WebMethod]
352.         public string CargarFeedXivelyBDD(string feedId)
353.         {
354.             try
355.             {
356.                 //Recupera el objeto de logica de negoicio de la aplicación para que el cambio de ontologia
    tenga efecto
357.                 semanticIndexManager = new SemanticIndexManager();
358.                 semanticIndexManager.CargarFeedXivelyBDD(feedId);
359.                 return "El índice se ha Actualizado correctamente desde el servidor Xively.";
360.             }
361.             catch(Exception ex)
362.             {
363.                 return "Ocurrio un error para cargar e indexar el Feed: " + ex.Message;
364.             }
365.         }
366.
367.         #endregion
368.
369.         #region "Servicios de Geolocalización de Sensores del Indice Semántico"
370.
371.         //Método Adicional: Permite Buscar un lugar geográfico por su nobre en el servicio Web de GeoNames
372.         [WebMethod]
373.         public List<GeonameNode> ObtenerLocalizacion(string lugar)
374.         {
375.             //Crea el Objeto que guarda todo el conocimiento del negocio
376.             semanticIndexManager = new SemanticIndexManager();
377.
378.             return semanticIndexManager.ObtenerLocalizaciones(lugar);
379.         }
380.
381.         //Método Adicional: Permite determinar y retornar los conceptos que son lugares geográficos y con sus
    posiciones
382.         //geolocalizadasa a trvés del servicio de Geonames.
383.         [WebMethod]
384.         public List<GeonameNode> ObtenerCiudadesConsulta(string Consulta)
385.         {
```

```
386.             //Crea el Objeto que guarda todo el conocimiento del negocio
387.             semanticIndexManager = new SemanticIndexManager();
388.
389.             return semanticIndexManager.ObtenerCiudadesConsulta(Consulta);
390.         }
391.
392.         //Obtiene un solo lugar dadas sus coordenadas
393.         [WebMethod]
394.         public GeonameNode ObtenerLugardeCoordenadas(string latitud, string longitud)
395.         {
396.             //Crea el Objeto que guarda todo el conocimiento del negocio
397.             semanticIndexManager = new SemanticIndexManager();
398.
399.             return semanticIndexManager.ObtenerLugardeCoordenadas(latitud, longitud);
400.         }
401.
402.         //Obtiene Los nombres de los lugares de una lista de GeonameNodes que recibe
403.         [WebMethod]
404.         public List<GeonameNode> ObtenerListaLugarCoordenadas(string geonodesSerializados)
405.         {
406.             //Crea el Objeto que guarda todo el conocimiento del negocio
407.             semanticIndexManager = new SemanticIndexManager();
408.
409.             //Desserializamos el objeto enviado
410.             List<GeonameNode> geonodes = geonodesSerializados.DeserializarTo<List<GeonameNode>>();
411.
412.             //Objeto copia para las operaciones
413.             List<GeonameNode> geonodestmp = new List<GeonameNode>();
414.
415.             foreach (GeonameNode geotmp in geonodes)
416.             {
417.                 geonodestmp.Add(semanticIndexManager.ObtenerLugardeCoordenadas(geotmp.Latitud,
    geotmp.Longitud));
418.             }
419.
420.             return geonodestmp;
421.         }
422.
423.         //Obtiene las coordenadas de una lista. Adicionalmente Obtiene la jerarquia de lugares de cada nodo
424.         [WebMethod]
425.         public List<GeonameNode> ObtenerListaLugarCoordenadasJerarquica(string geonodesSerializados)
426.         {
427.             //Crea el Objeto que guarda todo el conocimiento del negocio
428.             semanticIndexManager = new SemanticIndexManager();
429.
430.             //Desserializamos el objeto enviado
431.             List<GeonameNode> geonodes = geonodesSerializados.DeserializarTo<List<GeonameNode>>();
432.
433.             //Objeto copia para las operaciones
434.             List<GeonameNode> geonodestmp = new List<GeonameNode>();
435.
436.             foreach (GeonameNode geotmp in geonodes)
437.             {
438.                 List<GeonameNode> geonodestmpj = new List<GeonameNode>();
439.                 geonodestmpj = ObtenerLugardeCoordenadasJerarquia(geotmp.Latitud, geotmp.Longitud);
440.                 //Añadimos el último nodo que contiene el lugar buscado
441.                 if(geonodestmpj != null)
442.                     geonodestmp.Add(geonodestmpj[geonodestmpj.Count-1]);
443.             }
444.
445.             return geonodestmp;
446.         }
447.
448.         //Obtiene las coordenadas de una lista. Adicionalmente Obtiene de la jerarquia de lugares de cada
    nodo
449.         //La primera división política que corresponde a dicho lugar
450.         [WebMethod]
451.         public string ObtenerListaLugarCoordenadas_AM1(string geonodesSerializados)
452.         {
453.             //Crea el Objeto que guarda todo el conocimiento del negocio
454.             semanticIndexManager = new SemanticIndexManager();
455.
456.             //Desserializamos el objeto enviado
457.             List<GeonameNode> geonodes = geonodesSerializados.DeserializarTo<List<GeonameNode>>();
458.
459.             //Objeto copia para las operaciones
460.             List<GeonameNode> geonodestmp = new List<GeonameNode>();
```

```
461.
462.             foreach (GeonameNode geotmp in geonodes)
463.             {
464.                 List<GeonameNode> geonodestmpj = new List<GeonameNode>();
465.                 geonodestmpj = semanticIndexManager.ObtenerLugardeCoordenadasJerarquia(geotmp.Latitud,
    geotmp.Longitud);
466.
467.                 if (geonodestmpj != null)
468.                     foreach (GeonameNode getmpj in geonodestmpj)
469.                     {
470.                         if (getmpj.fcode == "ADM1")
471.                             geonodestmp.Add(getmpj);
472.                     }
473.             }
474.             //Serializar la lista de salida
475.             return geonodestmp.SerializarToXml();
476.         }
477.
478.         //Obtiene la jerarquía de lugares de un punto en el mapa
479.         [WebMethod]
480.         public List<GeonameNode> ObtenerLugardeCoordenadasJerarquia(string latitud, string longitud)
481.         {
482.             //Crea el Objeto que guarda todo el conocimiento del negocio
483.             semanticIndexManager = new SemanticIndexManager();
484.
485.             return semanticIndexManager.ObtenerLugardeCoordenadasJerarquia(latitud, longitud);
486.         }
487.
488.         //Obtiene la jerarquía dado un geonameId
489.         [WebMethod]
490.         public List<GeonameNode> GeoNames_Hierarchy(int GeonameId)
491.         {
492.             //Crea el Objeto que guarda todo el conocimiento del negocio
493.             semanticIndexManager = new SemanticIndexManager();
494.
495.             return semanticIndexManager.GeoNames_Hierarchy(GeonameId);
496.         }
497.
498.         #endregion
499.
500.         #region "Métodos de los Sensores"
501.
502.         //Este método devuelve en formato FeedXively los Metadatos del Sensor
503.         [WebMethod]
504.         public FeedXively RetornarMetadatosSensor(string IdSensor)
505.         {
506.             //Crea el Objeto que guarda todo el conocimiento del negocio
507.             semanticIndexManager = new SemanticIndexManager();
508.
509.             //Obtener el objeto de la BDD
510.             FeedXively xtemp = semanticIndexManager.ObtenerJSONFeedBDD(IdSensor);
511.
512.             return xtemp;
513.         }
514.
515.         //Este método devuelve en formato json los Metadatos del Sensor desde Xively
516.         [WebMethod]
517.         public string  RetornarJsonSensor(string IdSensor)
518.         {
519.             //Crea el Objeto que guarda todo el conocimiento del negocio
520.             semanticIndexManager = new SemanticIndexManager();
521.
522.             //Obtener el JSON de la BDD
523.             string xtemp = semanticIndexManager.ObtenerJsonFeed(ConfigurationManager.AppSettings["APIkey"],
    IdSensor);
524.
525.             //Retornar el json
526.             return xtemp;
527.         }
528.
529.         //Este método obtiene información de los datos que mide el sensor.
530.         //Se retorna todo el feed con los datapoints
531.         [WebMethod]
532.         public FeedXively RetornarDatosSensor(string feedID, DateTime fechaInicio, DateTime fechaFin)
533.         {
534.             //Crea el Objeto que guarda todo el conocimiento del negocio
535.             semanticIndexManager = new SemanticIndexManager();
536.
537.             //Obtener el JSON de la BDD
```

```
538.             return semanticIndexManager.RetornarDatosSensor(feedID, fechaInicio, fechaFin);
539.         }
540.
541.         //Este método retorna los datapoints de un datastream específico, de un feed específico
542.         [WebMethod]
543.         public string RetornarDatapointsFeed(string feedID, string DatastreamId, string fechaInicio, string
     fechaFin)
544.         {
545.             //Crea el Objeto que guarda todo el conocimiento del negocio
546.             semanticIndexManager = new SemanticIndexManager();
547.
548.             //Obtener el JSON de la BDD
549.             string datapoints = semanticIndexManager.RetornarDatapointsFeed(feedID, DatastreamId,
550.                                                                             Convert.ToDateTime(fechaInicio),
551.                                                                             Convert.ToDateTime(fechaFin));
552.             return datapoints;
553.         }
554.
555.         #endregion
556.
557.         #region "Métodos Internos del Servicio de Indice Semántico"
558.
559.         private DataSet SearchSemanticIndex(string Consulta, string idioma)
560.         {
561.             //string idioma = ObtenerConfig("idiomaBusqueda");
562.             string expansion = ObtenerConfig("utilizarExpansion");
563.
564.             //Crea el Objeto que guarda todo el conocimiento del negocio
565.             semanticIndexManager = new SemanticIndexManager();
566.
567.             //Crea la lista de documentos que serán el resultado de la búsqueda
568.             List<UrlDocument> allReponse = new List<UrlDocument>();
569.
570.             if (expansion == "Si")
571.             {
572.                 //Realiza la expansión de la consulta a través del servicio.
573.                 Consulta = ExpandirConsulta(Consulta, idioma);
574.             }
575.
576.             //Se busca en español o el ingles. Si es nulo busca por ambos.
577.             if (idioma == "Español")
578.             {
579.                 allReponse = semanticIndexManager.BuscarSemanticIndex(Consulta, true);
580.             }
581.             else if(idioma == "Ingles")
582.             {
583.                 allReponse = semanticIndexManager.BuscarSemanticIndex(Consulta, false);
584.             }
585.             else
586.                 allReponse = semanticIndexManager.BuscarSemanticIndex(Consulta, true); //Por defecto busca
     español
587.
588.             //para DATASET que se devolvera como resultado
589.             UrlDocument objUrld = new UrlDocument();
590.             DataSet nuevoDS = new DataSet();
591.             DataTable table1 = new DataTable("Objetos Encontrados");
592.
593.             table1.Columns.Add("Id");
594.             table1.Columns.Add("Title");
595.             table1.Columns.Add("Titulo_HTML");
596.             table1.Columns.Add("Url");
597.             table1.Columns.Add("URL_Sensor");
598.             table1.Columns.Add("Resumen");
599.             table1.Columns.Add("Tags");
600.             table1.Columns.Add("Localizacion");
601.             table1.Columns.Add("Dominio");
602.             table1.Columns.Add("Datastreams_feed");
603.             table1.Columns.Add("Website");
604.             table1.Columns.Add("Elevacion");
605.             table1.Columns.Add("Latitud");
606.             table1.Columns.Add("Longitud");
607.             //Campos para el PageRangkin
608.             table1.Columns.Add("Conceptos");   //Conceptos por los cuales fue encontrado en Xively
609.             table1.Columns.Add("Consulta");    //Consulta por la cual fue seleccionado el sensor
610.             table1.Columns.Add("Distancia");   //Distancia para efectos de un lugar específico
611.             table1.PrimaryKey = new DataColumn[] { table1.Columns["Id"] };
612.
```

```
613.             foreach (UrlDocument urldoc in allReponse)
614.             {
615.                 table1.Rows.Add(urldoc.Id, urldoc.Tittle, urldoc.TituloHTML(), urldoc.URL,
    urldoc.URLMostrar(), urldoc.Resume,
616.                                 urldoc.Tags, urldoc.Localizacion_name, urldoc.Domain,
    urldoc.Datastreams_feed, urldoc.Website,
617.                                 urldoc.Elevacion, urldoc.Latitud, urldoc.Longitud, urldoc.ConceptosLista(),
    Consulta);
618.             }
619.
620.             nuevoDS.Tables.Add(table1);
621.             return nuevoDS;
622.         }
623.
624.         private string ExpandirConsulta(string Consulta, string idioma)
625.         {
626.             //Realiza la expansión de la consulta a través del servicio.
627.             ExpancionConsulta ex = new ExpancionConsulta();
628.             return  ex.ExpandirConsulta(SemanticIndexManager.RutaOntologia, Consulta, idioma);
629.         }
630.
631.         private string BuscarConceptosOntologia(string Concepto, string idioma)
632.         {
633.             //Realiza la expansión de la consulta a través del servicio.
634.             ExpancionConsulta ex = new ExpancionConsulta();
635.             return ex.RetornarConceptosOntologia(SemanticIndexManager.RutaOntologia, Concepto, idioma);
636.         }
637.
638.         private string CargarConceptosOntologia(string nuevaontologia)
639.         {
640.             srvExpansionConsulta.ExpancionConsulta ex = new srvExpansionConsulta.ExpancionConsulta();
641.             string conceptos = ex.CargarConceptos(nuevaontologia);
642.             return conceptos;
643.         }
644.
645.         private string ObtenerConfig(string key)
646.             {
647.                 return ConfigurationManager.AppSettings[key];
648.             }
649.
650.         private string AsignarConfig(string key, string valor)
651.             {
652.                 try
653.                 {
654.                     // Get the application configuration file.
655.                     System.Configuration.Configuration config =
656.                       System.Web.Configuration.WebConfigurationManager.OpenWebConfiguration("~/");
657.
658.                     // Update the configuration file appSettings section.
659.                     config.AppSettings.Settings.Remove(key);
660.                     config.AppSettings.Settings.Add(key, valor);
661.
662.                     // Save the configuration file.
663.                     config.Save(System.Configuration.ConfigurationSaveMode.Modified);
664.
665.                     //Configuración guardada correctamente
666.                     return "ok";
667.                 }
668.                 catch (Exception ex)
669.                 {
670.                     return ex.Message;
671.                 }
672.             }
673.
674.         #endregion
```

5.17 Fase III – Actividad 4: Tarea 3: Implementación de Servicios de Índice

La implementación de los servicios del índice se encuentra a continuación

4. Código Fuente de Implementación de Servicios de Índice

```
675.         #region "Servicios Básicos del Indice semántico"
676.
677.         //Realiza la Búsqueda de la consulta dada, con la ontología y las opciones de configuración
678.         //realizadas previamente y almacenada en el Web.config
679.         [WebMethod]
680.         public DataSet Buscar(string stringBuscar, string idioma = "Español")
681.         {
682.             if (!string.IsNullOrEmpty(stringBuscar))
683.                 return SearchSemanticIndex(stringBuscar, idioma);
684.             else
685.                 return null;
686.         }
687.
688.         //Igual que la amterior pero no expande consulta y retorna una lista de FeedXively Serializados
689.         [WebMethod]
690.         public List<FeedXively> BuscarFeeds(string consulta, string idioma = "Español")
691.         {
692.             if (!string.IsNullOrEmpty(consulta))
693.             {
694.                 //Crea el Objeto que guarda todo el conocimiento del negocio
695.                 SemanticIndexManager semanticIndexManager = new SemanticIndexManager();
696.
697.                 //Crea la lista de documentos que serán el resultado de la búsqueda
698.                 List<UrlDocument> allReponse = new List<UrlDocument>();
699.
700.                 //Se busca en español o el ingles. Si es nulo busca por ambos.
701.                 if (idioma == "Español")
702.                 {
703.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true);
704.                 }
705.                 else if (idioma == "Ingles")
706.                 {
707.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, false);
708.                 }
709.                 else
710.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true); //Por defecto
    busca español
711.
712.                 //Armar la estructura de datos a retornar como resltado
713.                 List<FeedXively> SensoresResult = new List<FeedXively>();
714.
715.                 //Traemos todos los metadatos del sensor que estan almacenados en la BDD local
716.                 foreach (UrlDocument urldoc in allReponse)
717.                 {
718.                     FeedXively xtemp = semanticIndexManager.ObtenerJSONFeedBDD(urldoc.Id);
719.                     SensoresResult.Add(xtemp);
720.                 }
721.
722.                 return SensoresResult;
723.             }
724.             else
725.                 return null;
726.         }
727.
728.         //Igual que la anterior pero retorna una lista de FeedXively Serializados
729.         //Además de los datapoints correspondientes consultados en Xively.
730.         [WebMethod]
731.         public List<FeedXively> BuscarFeedsDataPoints(string consulta, DateTime fechaInicio, DateTime
    fechaFin, string idioma = "Español")
732.         {
733.             if (!string.IsNullOrEmpty(consulta))
734.             {
735.                 //Crea el Objeto que guarda todo el conocimiento del negocio
736.                 SemanticIndexManager semanticIndexManager = new SemanticIndexManager();
737.
738.                 //Crea la lista de documentos que serán el resultado de la búsqueda
739.                 List<UrlDocument> allReponse = new List<UrlDocument>();
740.
741.                 //Se busca en español o el ingles. Si es nulo busca por ambos.
742.                 if (idioma == "Español")
743.                 {
744.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true);
745.                 }
746.                 else if (idioma == "Ingles")
```

```
747.                 {
748.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, false);
749.                 }
750.                 else
751.                     allReponse = semanticIndexManager.BuscarSemanticIndex(consulta, true); //Por defecto
    busca español
752.
753.                 //Armar la estructura de datos a retornar como resltado
754.                 List<FeedXively> SensoresResult = new List<FeedXively>();
755.
756.                 //Traemos todos los metadatos del sensor qdesde le servidor Xively, junto con sus mediciones
757.                 foreach (UrlDocument urldoc in allReponse)
758.                 {
759.                     FeedXively xtemp = RetornarDatosSensor(urldoc.Id, fechaInicio, fechaFin);
760.                     SensoresResult.Add(xtemp);
761.                 }
762.
763.                 return SensoresResult;
764.             }
765.             else
766.                 return null;
767.         }
768.
769.         //Retorna un listado de conceptos directamente relacionados al concepto. Estos de obtienen
770.         //de la ontología almacenada. Descarta términos no encontrados y no busca en wordnet.
771.         //Si el concepto no esta en la ontología retorna una cadena vacia.
772.         [WebMethod]
773.         public string RetornarConceptos(string Concepto, string idioma = "Español")
774.         {
775.             return BuscarConceptosOntologia(Concepto, idioma);
776.         }
777.
778.         //Retorna un listado de sensores y su localización con respecto a la localización dada y una Consulta
779.         [WebMethod]
780.         public List<FeedXively> RetornarMapaLugar(double Latitud, double Longitud, string Consulta, string
    idioma = "Español", double radio = 100)
781.         {
782.             string expansion = ObtenerConfig("utilizarExpansion");
783.
784.             //Crea el Objeto que guarda todo el conocimiento del negocio
785.             semanticIndexManager = new SemanticIndexManager();
786.
787.             //Crea una copia de la lista para eliminar los sensores que no se les puede calcular distancia
788.             List<FeedXively> FeedsConPosicion = new List<FeedXively>();
789.
790.             //Lista de los resultados de la busqueda
791.             List<FeedXively> dtsResult = new List<FeedXively>();
792.
793.             //Buscar la consulta en el índice
794.             if (!string.IsNullOrEmpty(Consulta))
795.                 dtsResult = BuscarFeeds(Consulta, idioma);
796.             else
797.                 return null;
798.
799.             //Si encontro sensores entonces validar los que entran en el radio de acción
800.             if(dtsResult != null)
801.             {
802.                 List<string> clavesBorrar = new List<string>();
803.
804.                 foreach (FeedXively dtr in dtsResult)
805.                 {
806.                     if (dtr.feed.location != null)
807.                     {
808.                         //La funcion establece que el sensor este en el rango de N kms del lugar
    especificado. Si esta devuelve la distancia, sino devuelve cero
809.                         double distancia = semanticIndexManager.VerificarSensorenLugar(Latitud,
810.                                                 Longitud, Convert.ToDouble(dtr.feed.location.lat,
    culture),
811.                                                 Convert.ToDouble(dtr.feed.location.lon, culture),
    radio);
812.                         if (distancia > 0)
813.                         {
814.                             //ListaResultados.Add(indice, distancia);
815.                             GeonameNode nodolugardistancia = new GeonameNode();
816.                             nodolugardistancia.Latitud = Latitud.ToString();
817.                             nodolugardistancia.Longitud = Longitud.ToString();
818.                             lugar ld = new lugar();
819.                             ld.nodoGeoname = nodolugardistancia;
820.                             ld.distancia = distancia;
```

```
821.                            //Creamos el espacio en la variable de lista de lugares relacionada al sensor
822.                            dtr.lugares = new List<lugar>();
823.                            dtr.lugares.Add(ld);
824.                            FeedsConPosicion.Add(dtr);
825.                        }
826.                    }
827.                }
828.            }
829.
830.            //Finalmente retorna la lista con los sensores correspondientes
831.            return FeedsConPosicion;
832.        }
833.
834.        //Retorna un listado de sensores y su localización con respecto a la localización dada y una Consulta
835.        //Adicional retorna los datapoints de la fechas dadas
836.        [WebMethod]
837.        public List<FeedXively> RetornarMapaLugarDatapoints(double Latitud, double Longitud, string Consulta,
    DateTime fechaInicio, DateTime fechaFin, string idioma = "Español", double radio = 100)
838.        {
839.            string expansion = ObtenerConfig("utilizarExpansion");
840.
841.            //Crea el Objeto que guarda todo el conocimiento del negocio
842.            semanticIndexManager = new SemanticIndexManager();
843.
844.            //Crea una copia de la lista para eliminar los sensores que no se les puede calcular distancia
845.            List<FeedXively> FeedsConPosicion = new List<FeedXively>();
846.
847.            //Lista de los resultados de la busqueda
848.            List<FeedXively> dtsResult = new List<FeedXively>();
849.
850.            //Buscar la consulta en el índice
851.            if (!string.IsNullOrEmpty(Consulta))
852.                dtsResult = BuscarFeedsDataPoints(Consulta,fechaInicio, fechaFin, idioma);
853.            else
854.                return null;
855.
856.            //Si encontro sensores entonces validar los que entran en el radio de acción
857.            if (dtsResult != null)
858.            {
859.                List<string> clavesBorrar = new List<string>();
860.
861.                foreach (FeedXively dtr in dtsResult)
862.                {
863.                    if (dtr.feed.location != null && dtr.feed.location.lat != null && dtr.feed.location.lon
    != null)
864.                    {
865.                        //La funcion establece que el sensor este en el rango de N kms del lugar
    especificado. Si esta devuelve la distancia, sino devuelve cero
866.                        double distancia = semanticIndexManager.VerificarSensorenLugar(Latitud,
867.                                                    Longitud, Convert.ToDouble(dtr.feed.location.lat,
    culture),
868.                                                    Convert.ToDouble(dtr.feed.location.lon, culture),
    radio);
869.                        if (distancia > 0)
870.                        {
871.                            //ListaResultados.Add(indice, distancia);
872.                            GeonameNode nodolugardistancia = new GeonameNode();
873.                            nodolugardistancia.Latitud = Latitud.ToString();
874.                            nodolugardistancia.Longitud = Longitud.ToString();
875.                            lugar ld = new lugar();
876.                            ld.nodoGeoname = nodolugardistancia;
877.                            ld.distancia = distancia;
878.                            //Creamos el espacio en la variable de lista de lugares relacionada al sensor
879.                            dtr.lugares = new List<lugar>();
880.                            dtr.lugares.Add(ld);
881.                            FeedsConPosicion.Add(dtr);
882.                        }
883.                    }
884.                }
885.            }
886.
887.            //Finalmente retorna la lista con los sensores correspondientes
888.            return FeedsConPosicion;
889.        }
890.
891.        //Retorna un listado de Sensores y su localización con respecto a una localización dada y un conjunto
    se sensores
```

```
892.         [WebMethod]
893.         public List<FeedXively> RetornarMapaLugarListaSensores(double Latitud, double Longitud,
     List<FeedXively> dtsResult, string idioma = "Español", double radio = 100)
894.         {
895.             //Crea el Objeto que guarda todo el conocimiento del negocio
896.             semanticIndexManager = new SemanticIndexManager();
897.
898.             //Crea una copia de la lista para eliminar los sensores que no se les puede calcular distancia
899.             List<FeedXively> FeedsConPosicion = new List<FeedXively>();
900.
901.             //Si encontro sensores entonces validar los que entran en el radio de acción
902.             if (dtsResult != null)
903.             {
904.                 List<string> clavesBorrar = new List<string>();
905.
906.                 foreach (FeedXively dtr in dtsResult)
907.                 {
908.                     if (dtr.feed.location != null)
909.                     {
910.                         //La funcion establece que el sensor este en el rango de N kms del lugar
     especificado. Si esta devuelve la distancia, sino devuelve cero
911.                         double distancia = semanticIndexManager.VerificarSensorenLugar(Latitud,
912.                                                 Longitud, Convert.ToDouble(dtr.feed.location.lat,
     culture),
913.                                                 Convert.ToDouble(dtr.feed.location.lon, culture),
     radio);
914.                         if (distancia > 0)
915.                         {
916.                             //ListaResultados.Add(indice, distancia);
917.                             GeonameNode nodolugardistancia = new GeonameNode();
918.                             nodolugardistancia.Latitud = Latitud.ToString();
919.                             nodolugardistancia.Longitud = Longitud.ToString();
920.                             lugar ld = new lugar();
921.                             ld.nodoGeoname = nodolugardistancia;
922.                             ld.distancia = distancia;
923.                             //Creamos el espacio en la variable de lista de lugares relacionada al sensor
924.                             dtr.lugares = new List<lugar>();
925.                             dtr.lugares.Add(ld);
926.                             FeedsConPosicion.Add(dtr);
927.                         }
928.
929.                     }
930.                 }
931.             }
932.
933.             //Finalmente retorna la lista con los sensores correspondientes
934.             return FeedsConPosicion;
935.         }
936.
937.         //Realiza un proceso de expanción de la consulta del usuario utilizando la ontología para hacerla más
     exacta
938.         //Para esto primero Retorna conceptos, posteriormente busca en wordnet conceptos no encontrados y
     finalmente
939.         //enlaza los términos no identificados al final de la consulta por considerarse importantes para el
     usuario.
940.         [WebMethod]
941.         public string ExpandirConsultaConceptosOntologia(string Consulta, string idioma)
942.         {
943.             return ExpandirConsulta(Consulta, idioma);
944.         }
945.         #endregion
946.
947.         #region "Servicios de Configuración del Indice Semántico"
948.
949.         //Método Adicional: Permite recuperar los valores de las claves almacenadas en el Web.Config
950.         //con el fin de establecer la configuración actual del índice
951.         [WebMethod]
952.         public string ObtenerConfiguracion(string key)
953.         {
954.             if (!string.IsNullOrEmpty(key))
955.                 return ObtenerConfig(key);
956.             else
957.                 return "La clave no puede ser vacia o nula";
958.         }
959.
960.         //Método Adicional: Permite asignar un valor a una clave específica en el Web.config con el fin
961.         //de establecer una nueva configuración de indexción
962.         [WebMethod]
963.         public string AsignarConfiguracion(string key, string valor)
```

```
964.         {
965.             if (!string.IsNullOrEmpty(key) && !string.IsNullOrEmpty(key))
966.                 return AsignarConfig(key, valor);
967.             else
968.                 return "La clave o el valor no pueden ser nulos o vacios";
969.         }
970.
971.         //Método Adicional: Una vez se han cambiado las configuraciones o se desea inicia una nueva ontología
    de dominio
972.         //Es necesario crear el índice semántico. Este método crea el índice teniendo en cuenta si lo hace
    desde el servidor
973.         //de objetos Xively o desde los archivos locales almacenados en una indexación previa.
974.         [WebMethod]
975.         public string CrearIndiceSemantico()
976.         {
977.             //Crea el Objeto que guarda todo el conocimiento del negocio
978.             semanticIndexManager = new SemanticIndexManager();
979.             if (ObtenerConfig("BDDfuente") == "Xively")
980.             {
981.                 semanticIndexManager.CrearIndiceSemantico(true);
982.                 return "El índice se ha creado correctamente desde el servidor Xively.";
983.             }
984.             else
985.             {
986.                 semanticIndexManager.CrearIndiceSemantico(false);
987.                 return "El índice se ha creado correctamente desde la Base de Datos Local de Archivos JSON.";
988.             }
989.         }
990.
991.         //Método Adicional: Permite cambiar y cargar los conceptos a la BD del servicio de una nueva
    ontología.
992.         //Por defecto crea nuevamente el índice desde el servidor de objetos Xively.
993.         [WebMethod]
994.         public string CargarOntologiaeIndexar(string nuevaontologia)
995.         {
996.             string NombreArchivoNuevaOntologia = Path.GetFileName(nuevaontologia);
997.             string camino = SemanticIndexManager.RutaOntologiaSinArchivo;
998.             string conceptoscargados = string.Empty;
999.
1000.                 //Verificar ue el archivo de la ontologia existe
1001.                 if (File.Exists(nuevaontologia))
1002.                 {
1003.                     //Copia la ontologia al directorio de ontologías
1004.                     File.Copy(nuevaontologia, camino + NombreArchivoNuevaOntologia, true);
1005.                     //Se debe cargar los conceptos de la ontologia para el proceso de expansión de consulta
1006.                     conceptoscargados = CargarConceptosOntologia(nuevaontologia);
1007.
1008.                     //Si el proceso fue Exitoso se cambia la nueva ruta de la Ontología
1009.                     AsignarConfig("FileOntology", NombreArchivoNuevaOntologia);
1010.
1011.                     //Recupera el objeto de logica de negoicio de la aplicación para que el cambio de
    ontologia tenga efecto
1012.                     semanticIndexManager = new SemanticIndexManager();
1013.
1014.                     //Finalmente Crea el índice semántico Nuevamente desde Xively
1015.                     semanticIndexManager.CrearIndiceSemantico(true);
1016.
1017.                     //Retorna los conceptos cargados
1018.                     return conceptoscargados;
1019.                 }
1020.                 return "ERROR - La ruta del archivo de la ontología no es correcta o el archivo no existe";
1021.             }
1022.
1023.             //Método Adicional: Permite cargar un feed de Xively espécifico a la BDD
1024.             //Posteriormente lo indexa en el indice semántico
1025.             [WebMethod]
1026.             public string CargarFeedXivelyBDD(string feedId)
1027.             {
1028.                 try
1029.                 {
1030.                     //Recupera el objeto de logica de negoicio de la aplicación para que el cambio de
    ontologia tenga efecto
1031.                     semanticIndexManager = new SemanticIndexManager();
1032.                     semanticIndexManager.CargarFeedXivelyBDD(feedId);
1033.                     return "El índice se ha Actualizado correctamente desde el servidor Xively.";
1034.                 }
1035.                 catch(Exception ex)
```

```
1036.                 {
1037.                     return "Ocurrio un error para cargar e indexar el Feed: " + ex.Message;
1038.                 }
1039.             }
1040.
1041.             #endregion
1042.
1043.             #region "Servicios de Geolocalización de Sensores del Indice Semántico"
1044.
1045.             //Método Adicional: Permite Buscar un lugar geográfico por su nobre en el servicio Web de
     GeoNames
1046.             [WebMethod]
1047.             public List<GeonameNode> ObtenerLocalizacion(string lugar)
1048.             {
1049.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1050.                 semanticIndexManager = new SemanticIndexManager();
1051.
1052.                 return semanticIndexManager.ObtenerLocalizaciones(lugar);
1053.             }
1054.
1055.             //Método Adicional: Permite determinar y retornar los conceptos que son lugares geográficos y
     con sus posiciones
1056.             //geolocalizadasa a trvés del servicio de Geonames.
1057.             [WebMethod]
1058.             public List<GeonameNode> ObtenerCiudadesConsulta(string Consulta)
1059.             {
1060.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1061.                 semanticIndexManager = new SemanticIndexManager();
1062.
1063.                 return semanticIndexManager.ObtenerCiudadesConsulta(Consulta);
1064.             }
1065.
1066.             //Obtiene un solo lugar dadas sus coordenadas
1067.             [WebMethod]
1068.             public GeonameNode ObtenerLugardeCoordenadas(string latitud, string longitud)
1069.             {
1070.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1071.                 semanticIndexManager = new SemanticIndexManager();
1072.
1073.                 return semanticIndexManager.ObtenerLugardeCoordenadas(latitud, longitud);
1074.             }
1075.
1076.             //Obtiene Los nombres de los lugares de una lista de GeonameNodes que recibe
1077.             [WebMethod]
1078.             public List<GeonameNode> ObtenerListaLugarCoordenadas(string geonodesSerializados)
1079.             {
1080.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1081.                 semanticIndexManager = new SemanticIndexManager();
1082.
1083.                 //Desserializamos el objeto enviado
1084.                 List<GeonameNode> geonodes = geonodesSerializados.DeserializarTo<List<GeonameNode>>();
1085.
1086.                 //Objeto copia para las operaciones
1087.                 List<GeonameNode> geonodestmp = new List<GeonameNode>();
1088.
1089.                 foreach (GeonameNode geotmp in geonodes)
1090.                 {
1091.                     geonodestmp.Add(semanticIndexManager.ObtenerLugardeCoordenadas(geotmp.Latitud,
     geotmp.Longitud));
1092.                 }
1093.
1094.                 return geonodestmp;
1095.             }
1096.
1097.             //Obtiene las coordenadas de una lista. Adicionalmente Obtiene la jerarquia de lugares de cada
     nodo
1098.             [WebMethod]
1099.             public List<GeonameNode> ObtenerListaLugarCoordenadasJerarquica(string geonodesSerializados)
1100.             {
1101.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1102.                 semanticIndexManager = new SemanticIndexManager();
1103.
1104.                 //Desserializamos el objeto enviado
1105.                 List<GeonameNode> geonodes = geonodesSerializados.DeserializarTo<List<GeonameNode>>();
1106.
1107.                 //Objeto copia para las operaciones
1108.                 List<GeonameNode> geonodestmp = new List<GeonameNode>();
1109.
1110.                 foreach (GeonameNode geotmp in geonodes)
```

```
1111.                 {
1112.                     List<GeonameNode> geonodestmpj = new List<GeonameNode>();
1113.                     geonodestmpj = ObtenerLugardeCoordenadasJerarquia(geotmp.Latitud, geotmp.Longitud);
1114.                     //Añadimos el último nodo que contiene el lugar buscado
1115.                     if(geonodestmpj != null)
1116.                         geonodestmp.Add(geonodestmpj[geonodestmpj.Count-1]);
1117.                 }
1118.
1119.                 return geonodestmp;
1120.             }
1121.
1122.             //Obtiene las coordenadas de una lista. Adicionalmente Obtiene de la jerarquia de lugares de
     cada nodo
1123.             //La primera división política que corresponde a dicho lugar
1124.             [WebMethod]
1125.             public string ObtenerListaLugarCoordenadas_AM1(string geonodesSerializados)
1126.             {
1127.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1128.                 semanticIndexManager = new SemanticIndexManager();
1129.
1130.                 //Desserializamos el objeto enviado
1131.                 List<GeonameNode> geonodes = geonodesSerializados.DeserializarTo<List<GeonameNode>>();
1132.
1133.                 //Objeto copia para las operaciones
1134.                 List<GeonameNode> geonodestmp = new List<GeonameNode>();
1135.
1136.                 foreach (GeonameNode geotmp in geonodes)
1137.                 {
1138.                     List<GeonameNode> geonodestmpj = new List<GeonameNode>();
1139.                     geonodestmpj = semanticIndexManager.ObtenerLugardeCoordenadasJerarquia(geotmp.Latitud,
     geotmp.Longitud);
1140.
1141.                     if (geonodestmpj != null)
1142.                         foreach (GeonameNode getmpj in geonodestmpj)
1143.                         {
1144.                             if (getmpj.fcode == "ADM1")
1145.                                 geonodestmp.Add(getmpj);
1146.                         }
1147.                 }
1148.                 //Serializar la lista de salida
1149.                 return geonodestmp.SerializarToXml();
1150.             }
1151.
1152.             //Obtiene la jerarquía de lugares de un punto en el mapa
1153.             [WebMethod]
1154.             public List<GeonameNode> ObtenerLugardeCoordenadasJerarquia(string latitud, string longitud)
1155.             {
1156.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1157.                 semanticIndexManager = new SemanticIndexManager();
1158.
1159.                 return semanticIndexManager.ObtenerLugardeCoordenadasJerarquia(latitud, longitud);
1160.             }
1161.
1162.             //Obtiene la jerarquía dado un geonameId
1163.             [WebMethod]
1164.             public List<GeonameNode> GeoNames_Hierarchy(int GeonameId)
1165.             {
1166.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1167.                 semanticIndexManager = new SemanticIndexManager();
1168.
1169.                 return semanticIndexManager.GeoNames_Hierarchy(GeonameId);
1170.             }
1171.
1172.             #endregion
1173.
1174.             #region "Métodos de los Sensores"
1175.
1176.             //Este método devuelve en formato FeedXively los Metadatos del Sensor
1177.             [WebMethod]
1178.             public FeedXively RetornarMetadatosSensor(string IdSensor)
1179.             {
1180.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1181.                 semanticIndexManager = new SemanticIndexManager();
1182.
1183.                 //Obtener el objeto de la BDD
1184.                 FeedXively xtemp = semanticIndexManager.ObtenerJSONFeedBDD(IdSensor);
1185.
```

```
1186.                 return xtemp;
1187.             }
1188.
1189.             //Este método devuelve en formato json los Metadatos del Sensor desde Xively
1190.             [WebMethod]
1191.             public string  RetornarJsonSensor(string IdSensor)
1192.             {
1193.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1194.                 semanticIndexManager = new SemanticIndexManager();
1195.
1196.                 //Obtener el JSON de la BDD
1197.                 string xtemp =
     semanticIndexManager.ObtenerJsonFeed(ConfigurationManager.AppSettings["APIkey"], IdSensor);
1198.
1199.                 //Retornar el json
1200.                 return xtemp;
1201.             }
1202.
1203.             //Este método obtiene información de los datos que mide el sensor.
1204.             //Se retorna todo el feed con los datapoints
1205.             [WebMethod]
1206.             public FeedXively RetornarDatosSensor(string feedID, DateTime fechaInicio, DateTime fechaFin)
1207.             {
1208.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1209.                 semanticIndexManager = new SemanticIndexManager();
1210.
1211.                 //Obtener el JSON de la BDD
1212.                 return semanticIndexManager.RetornarDatosSensor(feedID, fechaInicio, fechaFin);
1213.             }
1214.
1215.             //Este método retorna los datapoints de un datastream específico, de un feed específico
1216.             [WebMethod]
1217.             public string RetornarDatapointsFeed(string feedID, string DatastreamId, string fechaInicio,
     string fechaFin)
1218.             {
1219.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1220.                 semanticIndexManager = new SemanticIndexManager();
1221.
1222.                 //Obtener el JSON de la BDD
1223.                 string datapoints = semanticIndexManager.RetornarDatapointsFeed(feedID, DatastreamId,
1224.
     Convert.ToDateTime(fechaInicio),
1225.
     Convert.ToDateTime(fechaFin));
1226.                 return datapoints;
1227.             }
1228.
1229.             #endregion
1230.
1231.             #region "Métodos Internos del Servicio de Indice Semántico"
1232.
1233.             private DataSet SearchSemanticIndex(string Consulta, string idioma)
1234.             {
1235.                 //string idioma = ObtenerConfig("idiomaBusqueda");
1236.                 string expansion = ObtenerConfig("utilizarExpansion");
1237.
1238.                 //Crea el Objeto que guarda todo el conocimiento del negocio
1239.                 semanticIndexManager = new SemanticIndexManager();
1240.
1241.                 //Crea la lista de documentos que serán el resultado de la búsqueda
1242.                 List<UrlDocument> allReponse = new List<UrlDocument>();
1243.
1244.                 if (expansion == "Si")
1245.                 {
1246.                     //Realiza la expansión de la consulta a través del servicio.
1247.                     Consulta = ExpandirConsulta(Consulta, idioma);
1248.                 }
1249.
1250.                 //Se busca en español o el ingles. Si es nulo busca por ambos.
1251.                 if (idioma == "Español")
1252.                 {
1253.                     allReponse = semanticIndexManager.BuscarSemanticIndex(Consulta, true);
1254.                 }
1255.                 else if(idioma == "Ingles")
1256.                 {
1257.                     allReponse = semanticIndexManager.BuscarSemanticIndex(Consulta, false);
1258.                 }
1259.                 else
1260.                     allReponse = semanticIndexManager.BuscarSemanticIndex(Consulta, true); //Por defecto
```

```
    busca español
1261.
1262.                //para DATASET que se devolvera como resultado
1263.                UrlDocument objUrld = new UrlDocument();
1264.                DataSet nuevoDS = new DataSet();
1265.                DataTable table1 = new DataTable("Objetos Encontrados");
1266.
1267.                table1.Columns.Add("Id");
1268.                table1.Columns.Add("Title");
1269.                table1.Columns.Add("Titulo_HTML");
1270.                table1.Columns.Add("Url");
1271.                table1.Columns.Add("URL_Sensor");
1272.                table1.Columns.Add("Resumen");
1273.                table1.Columns.Add("Tags");
1274.                table1.Columns.Add("Localizacion");
1275.                table1.Columns.Add("Dominio");
1276.                table1.Columns.Add("Datastreams_feed");
1277.                table1.Columns.Add("Website");
1278.                table1.Columns.Add("Elevacion");
1279.                table1.Columns.Add("Latitud");
1280.                table1.Columns.Add("Longitud");
1281.                //Campos para el PageRangkin
1282.                table1.Columns.Add("Conceptos");   //Conceptos por los cuales fue encontrado en Xively
1283.                table1.Columns.Add("Consulta");    //Consulta por la cual fue seleccionado el sensor
1284.                table1.Columns.Add("Distancia");   //Distancia para efectos de un lugar específico
1285.                table1.PrimaryKey = new DataColumn[] { table1.Columns["Id"] };
1286.
1287.                foreach (UrlDocument urldoc in allReponse)
1288.                {
1289.                    table1.Rows.Add(urldoc.Id, urldoc.Tittle, urldoc.TituloHTML(), urldoc.URL,
    urldoc.URLMostrar(), urldoc.Resume,
1290.                                    urldoc.Tags, urldoc.Localizacion_name, urldoc.Domain,
    urldoc.Datastreams_feed, urldoc.Website,
1291.                                    urldoc.Elevacion, urldoc.Latitud, urldoc.Longitud,
    urldoc.ConceptosLista(), Consulta);
1292.                }
1293.
1294.                nuevoDS.Tables.Add(table1);
1295.                return nuevoDS;
1296.            }
1297.
1298.            private string ExpandirConsulta(string Consulta, string idioma)
1299.            {
1300.                //Realiza la expansión de la consulta a través del servicio.
1301.                ExpancionConsulta ex = new ExpancionConsulta();
1302.                return  ex.ExpandirConsulta(SemanticIndexManager.RutaOntologia, Consulta, idioma);
1303.            }
1304.
1305.            private string BuscarConceptosOntologia(string Concepto, string idioma)
1306.            {
1307.                //Realiza la expansión de la consulta a través del servicio.
1308.                ExpancionConsulta ex = new ExpancionConsulta();
1309.                return ex.RetornarConceptosOntologia(SemanticIndexManager.RutaOntologia, Concepto, idioma);
1310.            }
1311.
1312.            private string CargarConceptosOntologia(string nuevaontologia)
1313.            {
1314.                srvExpansionConsulta.ExpancionConsulta ex = new srvExpansionConsulta.ExpancionConsulta();
1315.                string conceptos = ex.CargarConceptos(nuevaontologia);
1316.                return conceptos;
1317.            }
1318.
1319.            private string ObtenerConfig(string key)
1320.                {
1321.                    return ConfigurationManager.AppSettings[key];
1322.                }
1323.
1324.            private string AsignarConfig(string key, string valor)
1325.                {
1326.                    try
1327.                    {
1328.                        // Get the application configuration file.
1329.                        System.Configuration.Configuration config =
1330.                          System.Web.Configuration.WebConfigurationManager.OpenWebConfiguration("~/");
1331.
1332.                        // Update the configuration file appSettings section.
1333.                        config.AppSettings.Settings.Remove(key);
```

```
1334.                         config.AppSettings.Settings.Add(key, valor);
1335.
1336.                         // Save the configuration file.
1337.                         config.Save(System.Configuration.ConfigurationSaveMode.Modified);
1338.
1339.                         //Configuración guardada correctamente
1340.                         return "ok";
1341.                     }
1342.                     catch (Exception ex)
1343.                     {
1344.                         return ex.Message;
1345.                     }
1346.                 }
1347.
1348.             #endregion
```

5.18 Fase III – Actividad 4: Tarea 4: Evaluación del índice construido

La evaluación del índice construido se realizó utilizando las medidas actuales de recuperación de la información, ya que lo que se deseaba en el momento era establecer su capacidad para extraer información relevante del WoT. En el siguiente capítulo de la monografía se puede evidenciar este proceso.
Un elemento importante a mencionar es que cuando se hizo el sondeo de la información específica a recuperar en el servidor Xively, nos encontramos que no había suficientes sensores relacionados a dicho dominio sobre todo en Colombia, por tal razón se procedió a crear un conjunto de sensores virtuales en dos regiones de Colombia (Cauca y Santander), con el fin de poder tener datos más relacionados en nuestro entorno particular.

5.19 Fase IV – Actividad 1: Tarea 1, 2: Elaboración del Plan de Restitución y Ejecutar el despliegue

En esta última fase se realizo el despliegue en un Servidor Windows Server 2008, con base de datos SQL Server 2008 y Microsoft Framework 4, en un servidor web IIS 7.5 alojado en la siguiente dirección: http://semanticsearchiot.net/sswot/SearchWoT/.

ANEXO 6. IMPLEMENTACIÓN DE LA INTERFAZ MÓVIL EN MONITOREO DE LA CALIDAD DEL AIRE

6.1 REQUISITOS DE LA INTERFAZ MÓVIL DE ALERTAS MEDIOAMBIENTALES

Los requisitos que se discutieron y aprobaron en las sesiones, entre las partes fueron los siguientes:

- Mostrar Alerta: Informar al usuario sobre los niveles de contaminación medio ambiental según la normatividad vigente establecida por el ministerio del medio ambiente.
- Mostrar Geo localización de dispositivos: El usuario podrá desplegar un mapa delimitado en una de las instalaciones de la universidad del Cauca con todos los sensores desplegados y discriminados por colores. Cada color indicarán los rangos de contaminación.
- Mostrar Lista de dispositivos: El usuario podrá desplegar una lista con los sensores registrados en el área, discriminados por contaminación y ordenados por distancia según ubicación del usuario, además poder acceder a la información proporcionada por dichos sensores.
- Indexar dispositivo: Con el fin de proporcionarle al usuario una aplicación personalizable, se le ofrece la posibilidad de indexar en el servicio web semántico, su dispositivo creado en Xively, con el fin de que este pueda ser monitoreado por la aplicación.

Aprobaciones por el cliente

ID Hria	Enunciado de la Historia				Criterios de Aceptación			
	Rol	Característica / Funcionalidad	Razón / Resultado	(#) Esc.	Criterio de Aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado
1	Usuario	Informar al usuario sobre los niveles de contaminación medio ambiental según la normatividad vigente establecida por el ministerio del medio ambiente.	Prevenir posibles riesgos de salud para los usuarios.	1	Sin riesgo	El usuario está dentro del área geográfica donde la aplicación es efectiva.	Cuando la contaminación medio ambiental esta en los rangos más bajos.	El usuario puede disfrutar de sus actividades ininterrumpidamente sabiendo que su salud no corre peligro.
				2	Contaminación moderada	El usuario está dentro del área geográfica donde la aplicación es efectiva.	Cuando la contaminación medio ambiental esta en los rangos aceptables.	El usuario es informado que el área por donde transita tiene niveles de contaminación, que si bien, no son los más altos, pueden afectar su salud a largo plazo, y se le recomiendo pasar lo más rápido posible de esa zona.
				3	Alerta	El usuario está dentro del área geográfica donde la aplicación es efectiva.	Cuando la contaminación medio ambiental esta en los rangos más altos.	El usuario es informado que se encuentra en un área con alto nivel de contaminación y que su salud está en riesgo, debe salir de manera inmediata del área hasta que la aplicación le indique que está en una zona libre de riesgos medioambientales.

ID Hria	Enunciado de la Historia				Criterios de Aceptación			
	Rol	Característica / Funcionalidad	Razón / Resultado	(#) Esc.	Criterio de Aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado
2	Usuario	El usuario podrá desplegar un mapa delimitado en una de las instalaciones de la universidad del Cauca con todos los sensores desplegados y discriminados por colores. Cada color indicarán los rangos de contaminación.	El usuario podrá tener una visión amplia sobre el territorio a recorrer y elegir una ruta que favorezca a su salud.	1	Mapa desplegado	El usuario está dentro del área geográfica donde la aplicación es efectiva.	El usuario despliega el mapa de la aplicación.	El usuario tendrá pleno conocimiento de la situación medio ambiental del área por donde transita.
3	Usuario	El usuario podrá desplegar una lista con los sensores registrados en el área, discriminados por contaminación y ordenados por distancia según ubicación del usuario, además poder acceder a la información proporcionada por dichos sensores.	El usuario tendrá acceso al conocimiento relacionado con la contaminación medio ambiental de su entorno, de una manera ordenada y discriminada.	1	Lista desplegada	El usuario está dentro del área geográfica donde la aplicación es efectiva.	El usuario despliega la lista con los sensores de la aplicación.	El usuario tendrá pleno conocimiento de la situación medio ambiental discriminada por sensores.
				2	Lista desplegada y filtrada	El usuario está dentro del área geográfica donde la aplicación es efectiva.	El usuario despliega la lista filtrada con los sensores de la aplicación.	El usuario tendrá pleno conocimiento de la situación medio ambiental discriminada por sensores y filtrada por las necesidades del usuario.
4	Usuario	Con el fin de proporcionarle al usuario una aplicación personalizable, se le ofrece la posibilidad	El usuario podrá personalizar su aplicación con nuevos	1	Dispositivo Indexado.	El usuario cuenta con servicio a internet.		El usuario podrá añadir dispositivos al servicio web semántico para monitorearlos con la aplicación.

ID Hria	Enunciado de la Historia				Criterios de Aceptación			
	Rol	Característica / Funcionalidad	Razón / Resultado	(#) Esc.	Criterio de Aceptación (Título)	Contexto	Evento	Resultado / Comportamiento esperado
		de indexar en el servicio web semántico, su dispositivo creado en Xively, con el fin de que este pueda ser monitoreado por la aplicación.	dispositivos que él indexe en el servicio web semántico.					

Tabla 35. Aprobaciones por el cliente

6.2 DOCUMENTO DE CASOS E USO PARA LA APLICACIÓN MÓVIL DE ALERTAS MEDIOAMBIENTALES

6.2.1 Introducción

Crear una solución software siempre debe tener en cuenta los aspectos actuales que posea la organización donde se espera presente una solución; para este proyecto la Universidad del Cauca no es la excepción. De allí que se importante analizar las instalaciones de la Universidad del Cauca para encontrar puntos clave para el despliegue de dispositivos sensores para la captura de la calidad del aire, de allí que se presente el esquema actual de forma general y se haga aclaración del esquema de casos de uso necesario para dar soporte tanto a las arquitecturas que se plantearan para este proyecto como para la solución software que se llegue a implementar.

6.2.2 Propósito del Documento

El propósito de este documento es definir los casos de uso para una interfaz móvil de alertas medioambientales, presentando el modelo de casos de uso del estado actual y la propuesta con sus respectivos requerimientos. Es de recordar que la solución que se presentara al final del proyecto a nivel de software tangible será un prototipo de interfaces de usuario de los procesos monitoreo de la calidad del aire.

6.2.3 Objetivo general de los casos de uso

Se busca identificar el comportamiento deseado de la futura aplicación que se construya basado en el modelo de arquitectura de sistema planteado en el presente proyecto. Se describen el conjunto de secuencias identificadas y la interacción de los elementos tanto externos como los actores con el sistema. Asimismo, se identificarán los roles que tendrá cada actor del sistema; usualmente, un actor representa un rol en el sistema y puede ser una persona, un dispositivo hardware u otro sistema [1].

6.2.4 Enfoque

El enfoque del modelo de arquitectura que se va a proponer es el de una Interfaz Móvil de Alertas Medioambientales construido empleando un servicio Web Semántico.

6.2.5 Contexto

El contexto donde se espera implementar este proyecto es en las instalaciones universitarias de la Universidad del Cauca como son, la Facultad de ingeniería electrónica y telecomunicaciones, Facultad de ciencias contables, Centro deportivo universitario, Salones IPET, Facultad de física y química, con la finalidad de simular la monitorización del estado del aire en dichas locaciones. Se espera con este proyecto fortalecer y fomentar la Internet de las cosas.

6.3 Especificación Casos de Uso

6.3.1 Mostrar Alerta

RF- 1		
Versión	1.0 19/04/15	
Nombre	Mostrar Alerta	
Actores	Usuario	
Objetivos asociados		
Descripción	El dispositivo emitirá una alerta dependiendo del nivel de contaminación del aire.	
Precondición	La aplicación debe estar recibiendo información de un sensor.	
Secuencia	**Paso**	**Acción**
	1	El usuario abre la aplicación en el dispositivo móvil.
	2	La aplicación busca los sensores disponibles dentro del rango establecido
	3	La aplicación compara los datos recibidos por los sensores.
	4	La aplicación muestra tres tipos de mensajes (auditivo, visual, sensorial).
	5	El usuario accede a los detalles de la alerta desplegada por la aplicación.
Post-condición	Mensaje presentado al usuario	
Excepciones	**Paso**	**Acción**
	1	Si no hay conexión a internet la App muestra mensaje.

Tabla 36. Mostrar alerta

6.3.2 Mostrar Geo Localizacion de los Dispositivos

RF- 2		
Versión	1.0 19/04/15	
Nombre	Mostrar geo localización de los Dispositivos.	
Actores	Usuario, sistema	
Objetivos asociados		
Descripción	Monitorear sensores, ubicados en un punto geográfico dentro del radio de alcance definido.	
Precondición	El usuario debe tener conexión a internet.	
Secuencia	**Paso**	**Acción**
	1	El usuario abre la aplicación en el dispositivo móvil.

RF- 2		
	2	La aplicación busca los sensores disponibles dentro del rango establecido.
Post-condición	Se muestran la vista geográfica de los sensores.	
Excepciones	**Paso**	**Acción**
	1	Si no hay conexión a internet la App muestra mensaje.

Tabla 37. Mostrar Geolocalización de los dispositivos

6.3.3 Mostrar Lista de los Dispositivos.

RF- 3		
Versión	1.0 20/04/15	
Nombre	Mostrar Lista de los Dispositivos.	
Actores	Usuario, sistema	
Objetivos asociados		
Descripción	Mostrar una lista de los sensores ubicados en un punto geográfico dentro del radio de alcance definido.	
Precondición	El usuario debe tener conexión a internet.	
Secuencia	**Paso**	**Acción**
	1	El usuario abre la aplicación en el dispositivo móvil.
	2	La aplicación busca los sensores disponibles dentro del rango establecido.
	3	El usuario presiona el botón Menú
	4	El usuario Presiona el botón Listar Sensores
Post- condición	Se muestran los sensores.	
Excepciones	**Paso**	**Acción**
	1	Si no hay conexión a internet la App muestra mensaje.

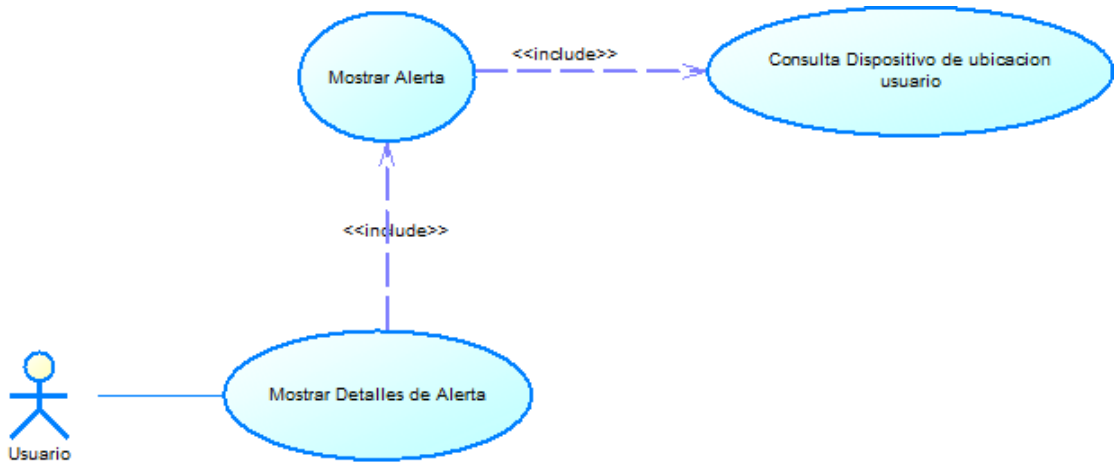
6.3.4 Indexar Dispositivos.

RF- 4		
Versión	1.0 20/04/15	
Nombre	Indexar Dispositivo.	
Actores	Usuario, sistema	
Objetivos asociados		
Descripción	Indexar un dispositivo, previamente creado en xively, al servidor web semántico	
Precondición	El usuario debe tener conexión a internet.	
Secuencia	**Paso**	**Acción**
	1	El usuario abre la aplicación en el dispositivo móvil.

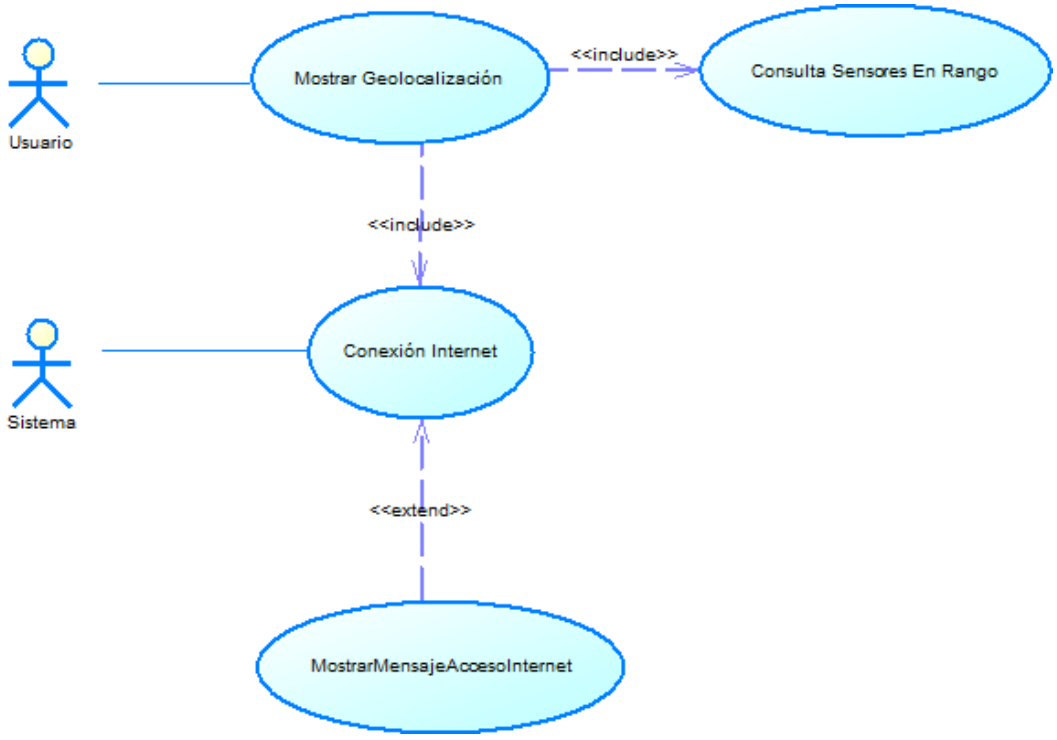
RF- 4		
	2	El usuario presiona el botón Indexar Dispositivo
	3	El usuario digita el Feed ID del Dispositivo
	4	El usuario espera por mensaje de confirmación.
Post- condición	Se muestran Mensaje de confirmmacion.	

6.4 Modelo de Casos de Uso

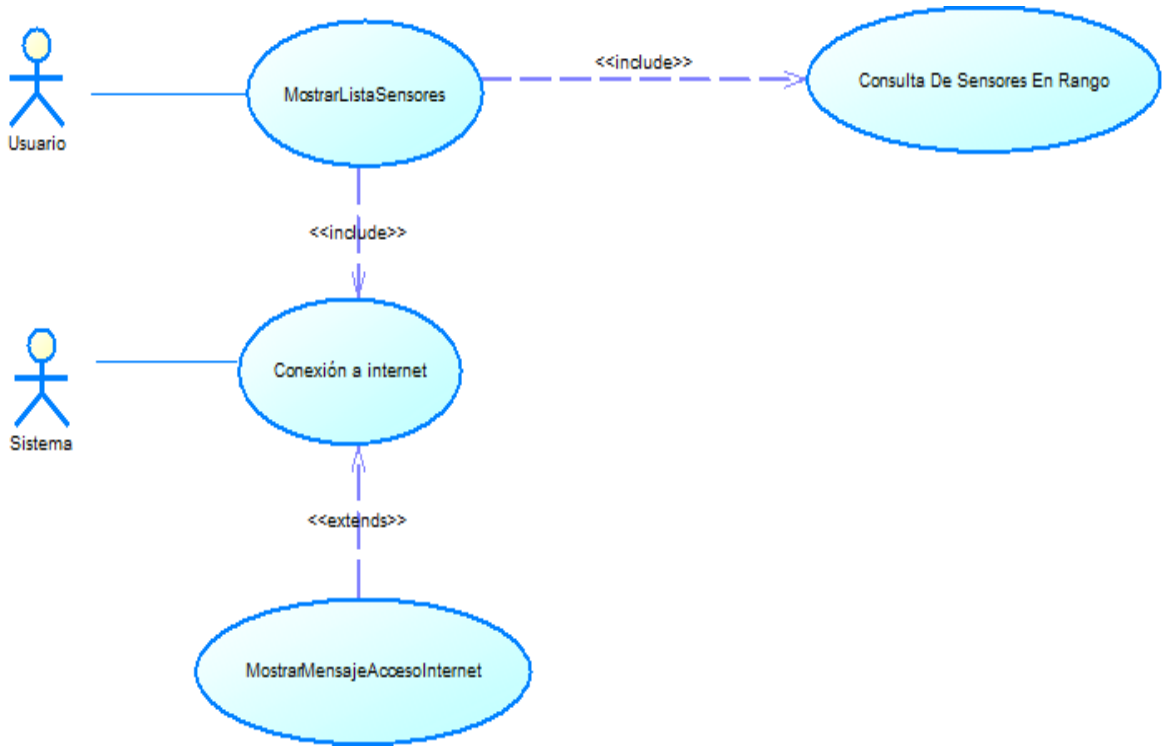
6.4.1 RF-1: Mostrar Alerta



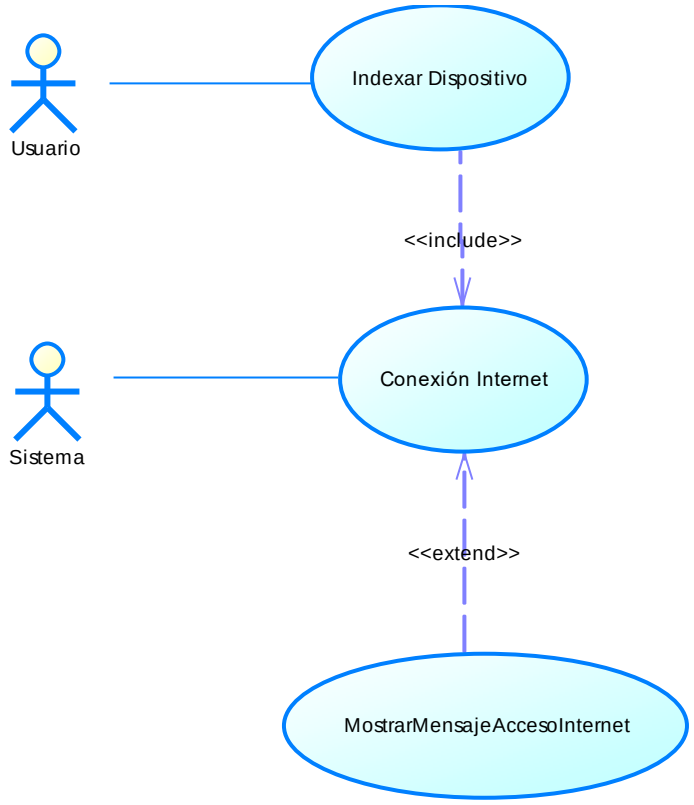
6.4.2 RF-2 Mostrar Geolocalización de los Dispositivos



6.4.3 RF-3 Mostrar Lista de los Dispositivos



6.4.4 RF-3 Mostrar Lista de los Dispositivos



6.5 MODELO DE DATOS DEL PROYECTO ENVIRALERT

EnvirAlert es una aplicación cuya fuente de datos quien provee los recursos para conocer la calidad del aire, esta soportado por el servicio web denominado WSSemanticSearch.

El servicio web nos provee 27 servicios que podemos usar y son estos:

- AsignarConfiguracion
- Buscar
- BuscarFeeds
- BuscarFeedsDataPoints
- CargarFeedXivelyBDD
- CargarOntologiaeIndexar
- CrearIndiceSemantico
- ExpandirConsultaConceptosOntologia
- GeoNames_Hierarchy
- ObtenerCiudadesConsulta
- ObtenerConfiguracion
- ObtenerListaLugarCoordenadas
- ObtenerListaLugarCoordenadasJerarquica
- ObtenerListaLugarCoordenadas_AM1

- ObtenerLocalizacion
- ObtenerLugardeCoordenadas
- ObtenerLugardeCoordenadasJerarquia
- RetornarConceptos
- RetornarDatapointsFeed
- RetornarDatosSensor
- RetornarJsonSensor
- RetornarMapaLugar
- RetornarMapaLugarDatapoints
- RetornarMapaLugarListaSensores
- RetornarMetadatosSensor
- SaveCalificacion
- SaveConsulta

Este objetivo de este servicio web es proveer información a las aplicaciones suscritas y que están enfocadas en la IoT (Internet de las cosas), en las cuales esas "cosas" siempre van a estar midiendo algún (os) atributo (s) del entorno en el que se encuentra, donde el objetivo del proyecto es conectar todo tipo de objeto y que nos brinde información en un tiempo t adecuado y esa información poder ser consumida desde cualquier aplicación o software.

EnvirAlert de los servicios que ofrece WSSemanticSearch, los que consume son:

- RetornarMapaLugarDatapoints
- RetornarDatosSensor

✓ RetornarMapaLugarDatapoints
Esta operación la usamos para consultar a partir de nuestra posición, recupera los FeedXively más cercanos en un rango en km establecidos; Con esta información podemos desplegar en el mapa los dispositivos que nos rodean en ese rango establecido, asi como listar estos dispositivos.
Los parámetros necesarios para esta operación son:

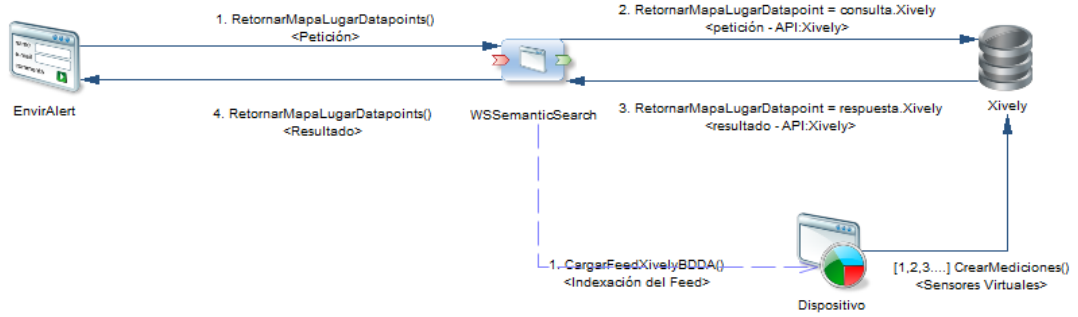
- Latitud
- Longitud
- Consulta
- Fecha final
- Fecha inicial
- Idioma
- Rango (km)

✓ RetornarDatosSensor
Esta operación la usamos para consultar la información de un dispositivo en especifico, mas precisamente nos enfocamos en el ultimo dato que arroja de ese dispositivo, que seria la información actual sobre la contaminación.
Los parámetros necesarios para esta operación son:

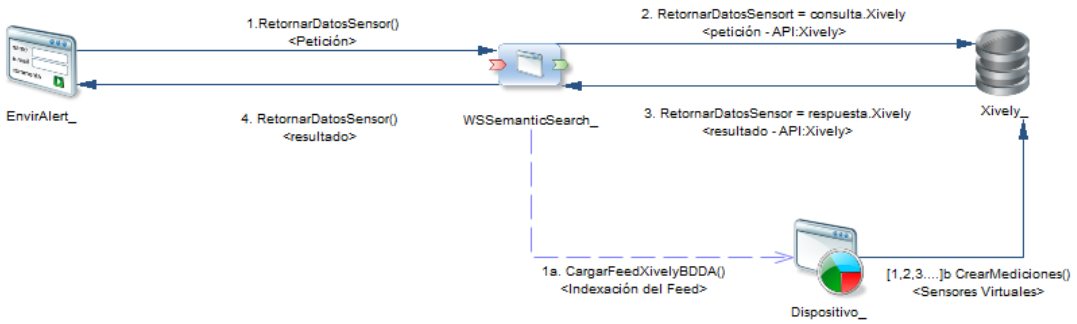
- feeId
- Fecha Inicial
- Fecha final

Cada operación que se realiza sobre WSSemanticSearch, este servicio realiza un proceso de busqueda semántica, y por medio del API de Xively hace que sea posible la obtención de datos que los dispositivos registran en esta.
Un pequeño esquema de como es la arquitectura de petición respuesta es este:

Operación : RetornarMapaLugarDatapoints



Operación: RetornarDatosSesor



Los dispositivos para efectos practicos, fue simulada la creación de datos como si se tratara de un dispositivo real, esto con una aplicación independiente que nos permite esto.

6.6 PLAN DE ITERACIONES

Historas de usuario con la iteración asignada

HISTORIA DE USUARIO	
NUMERO: 1	NOMBRE: Mostrar geolocalización de Dispositivos
USUARIO: Todos	PUNTOS ESTIMADOS:
PRIORIDAD EN NEGOCIO: Alto	PUNTOS ESTIMADOS:

RIESGO EN DESARROLLO: Medio	PUNTOS REALES:
DESCRIPCION: El usuario podrá mirar el mirar su ubicación y la de los dispositivos que se encuentran cercanos en el mapa, y discriminados por un color según sea el nivel de riesgo.	
OBSERVACIONES: Los colores que se representaran son cuatro, azul que pertenece a la posición del usuario, verde que indica que no hay riesgo de contaminación, naranja que pertenece a un nivel de contaminación medio, y rojo el cual significa que se encuentra en una zona de alto nivel de riesgo.	

HISTORIA DE USUARIO	
NUMERO: 2	NOMBRE: Mostrar lista de los Dispositivos
USUARIO: Todos	PUNTOS ESTIMADOS:
PRIORIDAD EN NEGOCIO: Alto	PUNTOS ESTIMADOS:
RIESGO EN DESARROLLO: Alto	PUNTOS REALES:
DESCRIPCION: Mostrar una lista de los sensores ubicados en un punto geográfico dentro del radio de alcance definido y mostrar en detalle un dispositivo seleccionado en tiempo real.	
OBSERVACIONES:	

HISTORIA DE USUARIO	
NUMERO: 3	NOMBRE: Mostrar Alertas
USUARIO: Todos	PUNTOS ESTIMADOS:
PRIORIDAD EN NEGOCIO: Alto	PUNTOS ESTIMADOS:
RIESGO EN DESARROLLO: Alto	PUNTOS REALES:
DESCRIPCION: El dispositivo emitirá una alerta dependiendo del nivel de contaminación del aire.	
OBSERVACIONES: Las alertas se muestran siempre y cuando este activadas.	

HISTORIA DE USUARIO	
NUMERO:4	NOMBRE: Buscar dispositivos
USUARIO: Todos	PUNTOS ESTIMADOS:
PRIORIDAD EN NEGOCIO: Medio	PUNTOS ESTIMADOS:
RIESGO EN DESARROLLO: Medio	PUNTOS REALES:
DESCRIPCION: El usuario podra buscar los dispositivos cercanos a el por el nombre de los dispositivos, o buscar por su FeedId, y poder mirar los niveles de contaminación que estan marcando los sensores en tiempo real.	
OBSERVACIONES:	

HISTORIA DE USUARIO	
NUMERO:5	NOMBRE: Indexar Dispositivo
USUARIO: Todos	PUNTOS ESTIMADOS:
PRIORIDAD EN NEGOCIO: Medio	PUNTOS ESTIMADOS:
RIESGO EN DESARROLLO: Alto	PUNTOS REALES:
DESCRIPCION: El usuario podra indexar en el servicio web WSSemanticSearch	

un nuevo dispositivo.
OBSERVACIONES: El nuevo dispositivo a indexar debe estar bien configurado con los parametros necesarios y completos. Este requerimiento es adicional por petición del cliente, adicional a los requerimientos iniciales.

HISTORIA DE USUARIO	
NUMERO:5	NOMBRE: Crear medición
USUARIO: Sistema	PUNTOS ESTIMADOS:
PRIORIDAD EN NEGOCIO: Alto	PUNTOS ESTIMADOS:
RIESGO EN DESARROLLO: Alto	PUNTOS REALES:
DESCRIPCION: El sistema generará datos de simulación para almacenarlos en Xively	
OBSERVACIONES: Es un subproyecto aparte de la aplicación EnvirAlert, que fue modificada y adaptada a nuestros criterios.	

Asignación de iteraciones por parejas de programadores, y fechas de asignación y resultados.

Historia de usuario	Tarea	Fecha Inicio	Fecha finalización	Asignado a:	Estado
ITERACION 1					
Crear medición	Desensamble de proyecto y reconstrucción	20/03/2015	23/03/2015	Yamid Noguera, Julio Luna	Completado
ITERACION 3					
Mostrar geolocalización de dispositivos	Diseño GUI	2/04/2015	5/04/2015	Andrés Gómez, Yamid Noguera	Completado
	Programar lógica.	6/05/2015	10/05/2015	Andrés Gómez, Luis Medina	Completado
Mostrar lista de dispositivos	Diseño GUI	6/04/2015	9/04/2015	Julio Luna, Yamid Noguera	Completado
	Programar lógica	11/05/2015	17/05/2015	Yamid Noguera, Julio Luna	Completado
Mostrar Alertas	Diseño GUI	6/04/2015	6/04/2015	Luis Médina, Julio Cesar	Completado
	Programar Lógica	11/05/2015	21/05/2015	Luis Médina, Julio Cesar	Completado
Buscar Dispositivos	Diseño GUI	19/05/2015	23/05/2015	Andrés Gómez, Luis Medina	Completado
	Programar Logica	27/05/2015	10/06/2015	Andrés Goméz, Yamid Noguera	Completado
Indexar Dispositivo	Diseño GUI	9/06/2015	15/06/2015	Andrés Gómez	Completado
	Programar Lógica	15/06/2015	17/06/2015	Andrés Gómez	Completado

ANEXO 7. IMPLEMENTACIÓN DEL ESCENARIO DE INTERACCIÓN SEMÁNTICA

En este capítulo se presenta la definición particular del escenario de interacción semántica de objetos inteligentes en la web de las cosas después de desarrollar un estado del arte de proyectos relacionados con la interacción semántica y los modelos de interacción existentes.

A partir de la arquitectura propuesta por Niño-Zambrano (Architecture for Semantic Interaction of Things in the Web of Things) se identificaron los elementos, las relaciones y procesos requeridos para la implementación del escenario, luego de la fase de capacitación se obtiene el diseño del escenario de interacción semántica en la web de las cosas. Para el desarrollo de las fases se hace uso de la metodología de desarrollo de software UP Ágil.

7.1 FASE DE INICIO

La implementación del escenario de interacción inicia con la manipulación de las herramientas hardware brindadas por la universidad, las placas Galileo Intel, una por cada servicio básico implementado; las placas permiten la instalación de la distribución de Linux POKY aprobada por Intel y trabajar con el lenguaje Python. Para ello es necesario utilizar una tarjeta de memoria SD externa.

Se realizó un estudio de ontologías (ref), del cual se eligió la ontología DogOnt REF, la cual permite asociar cada dispositivo a un contexto dentro del entorno de la domótica, además permite apoyar comportamientos inteligentes y el modelado de la casa, está diseñada para describirla en el mundo real con las capacidades del sistema domótico y apoyar la interoperación entre las soluciones actualmente disponibles y futuras. Describe los dispositivos en elementos controlables y arquitectónicos. Asocia para cada dispositivo un estado y una funcionalidad. Esta ontología es capaz de enfrentarse a consultas como:

- Donde se encuentra un dispositivo domótico.
- El conjunto de capacidades de un dispositivo.
- Los rasgos específicos de tecnología necesarios para la interconexión.
- Las posibles configuraciones que el dispositivo puede asumir.
- cómo se compone el ambiente del hogar;
- qué tipo de elementos arquitectónicos y muebles se colocan dentro de la casa.

Para el desarrollo de este proyecto se toma como el elemento arquitectónico a la sala de estar “living room”, se toma los servicios de regulación de temperatura y de luz. Fue preciso expandirla debido a que las funcionalidades de regulación de calidad de aire y humedad no se encontraban predefinidas.

Se utilizó Protegé como herramienta software para la gestión de las ontologías utilizadas.

Como producto de esta fase se tiene que se requiere un espacio físico donde se pueda realizar el despliegue del hardware para suplir los servicios inteligentes; en el caso particular del despliegue de este proyecto se toma como escenario de interacción una casa inteligente -Smart home, equipada con cinco servicios inteligentes, los cuales se despliegan en sólo un lugar arquitectónico denominado la sala de estar. En este espacio se debe realizar la implementación de los servicios inteligentes básicos: regulación de temperatura, luz, calidad de aire y humedad; así como se debe proveer de un mecanismo de control con el cual el usuario gestiona y crea nuevos servicios. Este

mecanismo es llamado Clipio, objeto inteligente asociado a un teléfono inteligente Nexus 4, cuyas características principales son: memoria RAM de 2 GB, posee GPS y comunicaciones inalámbricas mediante Wifi y NFC. A continuación se realiza una breve descripción de los servicios inteligentes.

- Regulador de temperatura. Este servicio le permite al usuario establecer la temperatura deseada en la cual se debe mantener en el espacio denominado sala de estar; es prestado por el objeto inteligente con el mismo nombre, el cual contiene dos actuadores, un conjunto de sensores del mismo tipo y un controlador. El actuador termostato está dedicado a subir la temperatura en la habitación, con fines prácticos este objeto será simulado por un led rojo. El actuador ventilador está dedicado a bajar la temperatura en la habitación. Un conjunto de sensores desplegados en diferentes puntos de la habitación dedicados a medir la temperatura del ambiente, estos sensores son de tipo analógico LM34.
- Regulador de luz. Este servicio es capaz de mantener un nivel de luz deseado en el espacio denominado sala de estar; es prestado por el objeto inteligente denominado de la misma forma, está compuesto por un actuador, un conjunto de sensores y un controlador. El actuador denominado bombillo, por fines prácticos será simulado mediante un led amarillo, el conjunto de sensores serán desplegados en diferentes puntos de la habitación, los cuales están dedicados a medir la intensidad de luz, estos sensores son de tipo fotoresistores analógicos LDR07
- Regulador de humedad en una matera: Permite controlar la humedad requerida por una planta en una matera, dentro del espacio denominado sala de estar, este servicio es prestado por un objeto inteligente llamado de la misma forma, está compuesto por un sensor de humedad XXXX, el actuador hace referencia al sistema de riego controlado por un motor.
- Calidad del aire: Con este servicio es posible regular un nivel de contaminación en el aire y brindar un ambiente más ameno para el usuario en el espacio sala de estar.

Se tiene como controlador de cada servicio a una tarjeta Galileo. Cada servicio presenta las siguientes características:

- Monitoreo constante.
- Alertas y notificaciones.
- Sugerencia de actividades.
- Control inteligente básico.

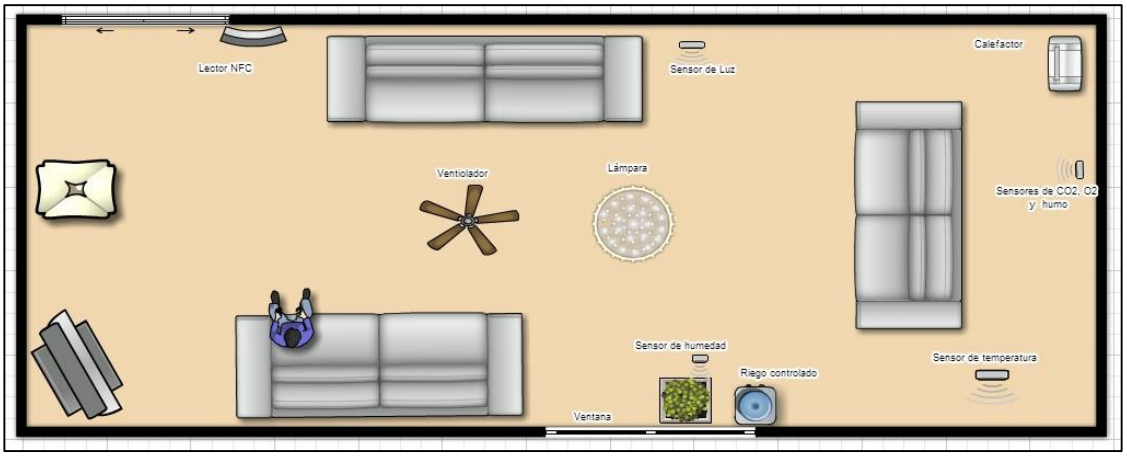


Figura 22. Distribución física de los recursos IoT

Análisis de requerimientos: Se realizara una aplicación que permita al usuario realizar búsquedas de dispositivos de los cuales se pueda elegir un conjunto para la creación de un nuevo servicio.

- La búsqueda del usuario se debe realizar en lenguaje natural.
- Utilización del índice semántico para realizar la búsqueda de dispositivos.
- Es preciso que los resultados de la búsqueda sean precisos y despliegue los dispositivos que pertenecen al contexto.
- La interfaz debe ser amigable al usuario y que le permita una fácil interacción.

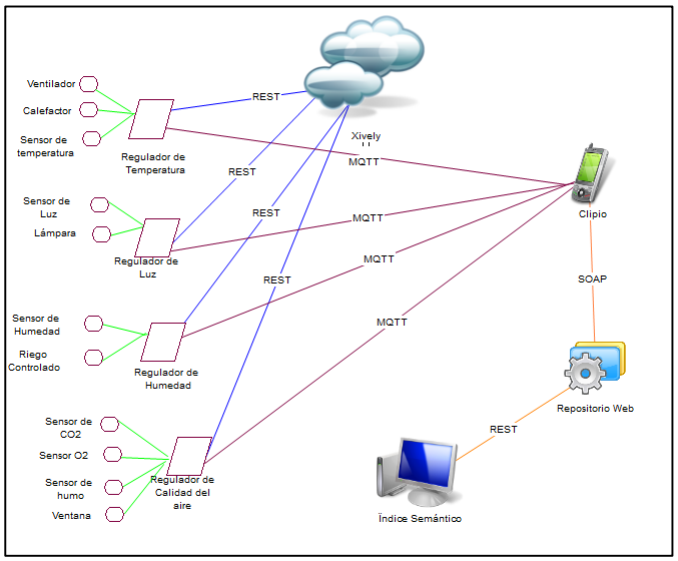


Figura 23. Visión general de los protocolos de comunicación utilizados

Se tiene como protocolos involucrados en la comunicación a:

- MQTT: Para la comunicación entre objetos inteligentes.
- SOAP: Permite la comunicación entre el usuario y el índice semántico.

- REST: Es utilizado para las notificaciones de estado de los objetos inteligentes vía Twiter.

7.2 FASE DE ELABORACIÓN

En esta fase se realizaron los diseños de casos de uso, diagramas conceptuales, diagramas de clase, de interacción, los cuales, permiten mostrar al usuario el funcionamiento del prototipo a crear. Para el desarrollo del escenario de interacción semántica se definen tres módulos importantes, los cuales son: Módulo de recuperación de objetos inteligentes –MROI, Módulo de contexto de objetos inteligentes –MCOI y el Módulo de interacción de objetos inteligentes –MIOI.

7.2.1 Módulo de recuperación de objetos inteligentes –MROI

Este módulo permite almacenar, recuperar y visualizar los datos de los diferentes objetos inteligentes, almacenados en el servidor middleware en la nube. Para ello se hace uso de la aplicación web brindado por el servidor middleware. En cuanto a la interacción realizada con el índice semántico se cuenta con la aplicación de Clipio.

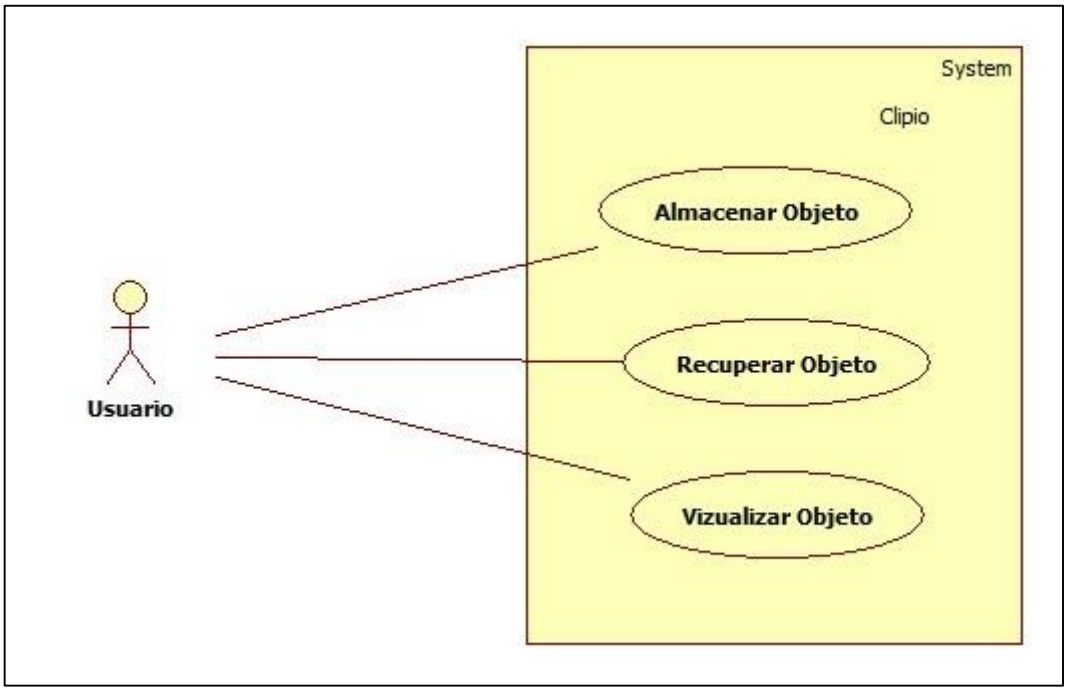


Figura 24. Diagrama de casos de uso del módulo MORI

Caso de uso: Almacenar objeto

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.

Descripción	Un usuario tendrá la opción de interactuar con la aplicación Clipio para almacenar los datos de los objetos inteligentes
Comentarios	Permite la indexación de un nuevo objeto inteligente.

Caso de uso: Recuperar objeto

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	Un usuario tendrá la opción de recuperar la información de un objeto inteligente en particular.
Comentarios	Esta búsqueda de información se realiza mediante la aplicación Clipio.

Caso de uso: Visualizar objeto

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	Un usuario tendrá la opción de visualizar la información de un objeto inteligente en particular.
Comentarios	Esta visualización de información se realiza mediante la aplicación Clipio.

7.2.2 Módulo de Contexto de Objetos Inteligentes – MCOI

Inicialmente se consideraba que éste módulo era el responsable de la asociación del contexto a cada objeto inteligente, para lo cual se tenía a la ontología objeto semántico pero, durante el desarrollo del presente trabajo fue preciso realizar una modificación, la cual toma los conceptos de la ontología DogOnt para realizar la indexación semántica, entonces, cuando el objeto inteligente coordinador, Clipio, pregunta por la entidad de interés (contexto), éste realiza la búsqueda en el índice semántico, el cual consulta la respectiva ontología de dominio para extraer la información. En la Figura 25, se representa el diagrama de casos de uso correspondientes al MCOI, en el cual se puede evidenciar que en el caso de uso almacenar, específicamente, el actuador puede ser un objeto inteligente si se requiere un proceso automático o puede ser el usuario tradicional si el proceso se desea realizar de forma manual.

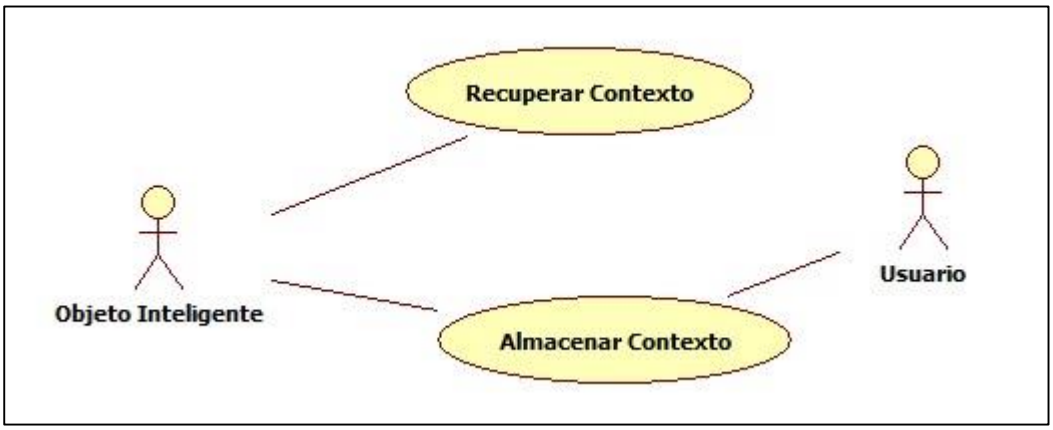


Figura 25. Diagrama de casos de uso MCOI

Caso de uso: Recuperar Contexto

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	Un usuario puede realizar la búsqueda de un objeto descrinándolo por contexto.
Comentarios	El índice semántico tiene la opción de trabajar con varios contextos al mismo tiempo y es éste quien tiene la habilidad de discriminarlos.

Caso de uso: Almacenar Contexto

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	El usuario y objeto inteligente pueden realizar la opción de almacenar los metadatos de los objetos inteligentes.
Comentarios	Cuando este caso de uso es realizado por un objeto inteligente se lleva a cabo de forma automática, en caso contrario se realiza de forma manual.

7.2.3 Módulo de Interacción de Objetos Inteligentes – MIOI

Este módulo permite la creación de los nuevos servicios basados en la interacción, se cuenta con cuatro objetos inteligentes asociados cada uno a un servicio básico y un teléfono inteligente que actúa como objeto inteligente y coordinador del sistema, llamado Clipio.

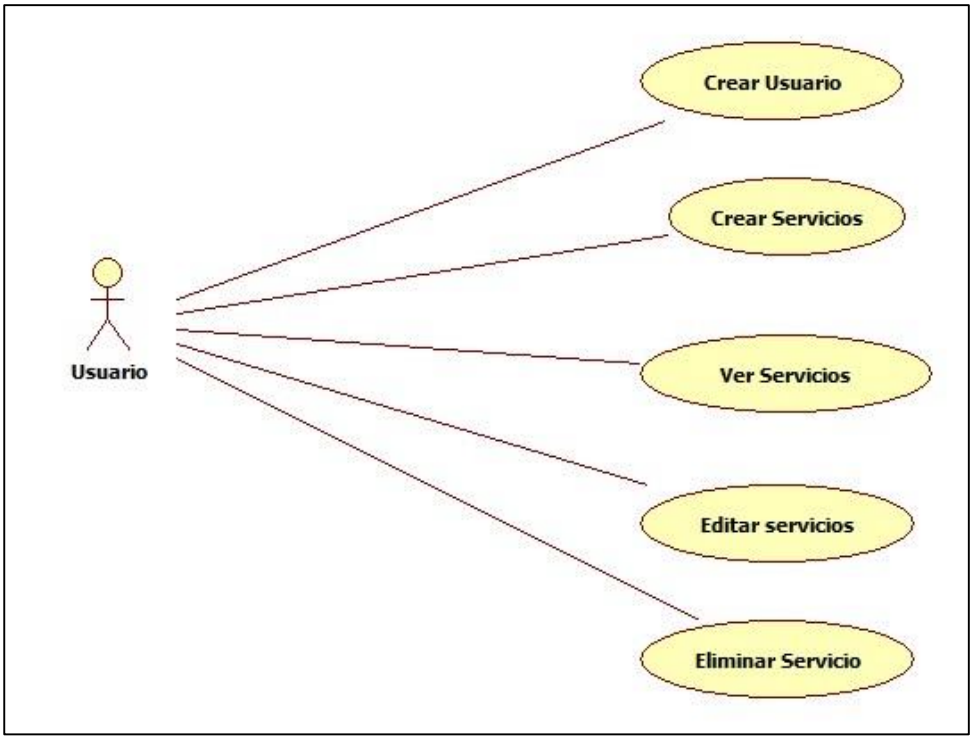


Figura 26. Diagrama de casos de uso MIOI

Caso de uso: Crear usuario

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	Un nuevo usuario tiene la posibilidad de ingresar al sistema para crear sus servicios.
Comentarios	Para la primera versión del escenario de interacción se maneja la precondición de que el nuevo usuario tiene todos los permisos para modificar el contexto al que pertenezca y manipular los objetos.

Caso de uso: Crear servicios

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	Este caso de uso permite crear un servicio a partir del requerimiento personalizado del usuario.
Comentarios	Se tiene como requisitos para la creación de un nuevo servicio la interacción de dos o más objetos inteligentes.

Caso de uso: Ver servicios

Actor ACT-01	Usuario
Autores	Dalila Riobamba

	Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	El usuario puede, en cualquier momento solicitar información acerca de los servicios creados o asociados a éste.
Comentarios	Ver servicio implica preguntar a cada objeto inteligente con comportamiento de actuador acerca de los servicios asociados al usuario.

Caso de uso: Editar servicios

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	Después de ver los servicios el usuario puede modificar un servicio en particular.
Comentarios	Los campos a modificar son: el evento, la condición y la acción. No tanto en el tipo de los mismos sino en los recursos para el caso del evento y la acción; en el caso de la condición se requiere solamente el cambio en la magnitud.

Caso de uso: Eliminar servicio

Actor ACT-01	Usuario
Autores	Dalila Riobamba Santiago Guerrero
Fuentes	Análisis de requerimientos del prototipo software.
Descripción	Si el usuario ya no requiere el servicio listado en el caso de uso ver servicio, puede eliminarlo.
Comentarios	Cuando se elimina un servicio, si se lo requiere en un futuro debe ser construido a partir de la búsqueda en lenguaje natural, no existe la opción de recuperarlo.

Se tiene como requisitos para el sistema:

- Despliegue y puesta en marcha de los objetos inteligentes.
- Indexación de los objetos semánticos haciendo uso de los conceptos de las ontologías para la creación del contexto.
- Definir las entidades de interés mediante TAGs NFC.

El usuario que requiera los servicios ofrecidos por el sistema debe:

- Tener un perfil de usuario, el cual consiste en un nombre, ingresado por el usuario y validado o guardado por el sistema.
- Especificar la entidad y propiedad de interés en la búsqueda en lenguaje natural.

7.3 Fase de construcción

En esta sección se realiza la construcción del prototipo del escenario de interacción que permite la comunicación y cooperación entre los objetos inteligentes para construir nuevos servicios para el usuario a partir de la ontología de dominio indexada.

7.3.1 Iteración 1

En para la realización de esta iteración se realizó la representación digital de los objetos inteligentes, para ello se hace la incorporación de los mismos en Xively, el middleware seleccionado, este proceso se lo había planeado realizar mediante la implementación de un módulo independiente, pero en el transcurso del desarrollo del trabajo se observó que este proceso ya estaba implementado por parte del middleware, este proceso se realizó de manera manual.

Para la implementación del MROI se hace uso de Android Studio 1.2.1.1 para la programación debido a que éste módulo es una funcionalidad de Clipio, aplicación desarrollada para Android 5.1.1 (API 22). El lenguaje de programación utilizado en el desarrollo de la aplicación Clipio, es java.

Las principales funcionalidades de éste módulo son: almacenar, recuperar y visualizar objetos inteligentes.

- Almacenar objeto: Permite realizar la indexación de un objeto inteligente mediante la aplicación Clipio utilizando únicamente el FeedID.
- Recuperar objeto: Es capaz de realizar la búsqueda en lenguaje natural solicitada por parte del usuario y presenta los resultados en forma de lista.
- Visualizar objeto: Presenta a petición del usuario los detalles del objeto solicitado.

7.3.2 Iteración 2

Inicialmente se había previsto que en esta iteración se debía implementar los mecanismos de creación de objetos semánticos,
Para los mecanismos de indexación semántica,
La funcionalidad de búsqueda y uso de los servicios de información expuestos por los objetos. Para esto se implementa el MCOI.

Actualmente se ha desarrollado en esta iteración un módulo que permite realizar la búsqueda de objetos inteligentes y/o recursos en un contexto particular y la funcionalidad de agregar metadatos a cada objeto inteligente.

- Pruebas Alfa y ajustes de las herramientas software.

7.3.3 Iteración 3

Debido a que se implementa un solo escenario de interacción con un único usuario, el cual sirve como mecanismo de control, debido a que no se realiza gestión de contextos y de perfiles de usuario. Dentro del escenario se cuenta con un objeto inteligente, Clipio, el cual realiza la tarea de coordinación de las interacciones entre los objetos, es éste mismo objeto el encargado de la monitorización del estado de los demás objetos inteligentes.

El MIOI, es principalmente la aplicación Android llamada Clipio, escrita en java con el soporte de Android Studio versión 1.2.1.1, el cual, además es la interfaz amigable al usuario. Sus principales funcionalidades dentro de esta iteración, son: Crear usuario, crear ver, editar y eliminar servicio.

- Crear usuario: permite al usuario personalizar la aplicación con su nombre y sus servicios asociados.
- Crear servicio: Es la funcionalidad núcleo del presente proyecto, en esta sección el usuario puede ingresar su requerimiento en lenguaje natural, de forma escrita o por comando de voz, seleccionar los recursos IoT deseados, luego de indicar la condición a cump0lir por el sistema, se da por terminado el proceso de creación de un servicio producto de la interacción de objetos inteligentes.
- Ver servicio: Le permite al usuario listar los servicios creados por él; además de acceder a sus correspondientes detalles.
- Editar servicio: Después de visualizar los objetos, el usuario puede, si lo desea, modificar cualquier característica del mismo.
- Eliminar servicio: Esta opción le permite al usuario eliminar los servicios que con los que no esté de acuerdo o que ya no los requiera, eliminar el servicio, implica eliminar el ECA del objeto inteligente con comportamiento de actuador.

  - Implementación de los mecanismos de control, monitorización e interacción de los dispositivos seleccionados en el caso de estudio particular. Para esto se implementa el MIOI.
  - Pruebas Alfa y ajustes de las herramientas software.

7.3.4 Iteración 4

  - Implementación de los parámetros a medir en las interacciones creadas en el caso de estudio particular, que fueron identificados en la fase de inicio.
  - Pruebas Alfa y ajustes de las herramientas software.

ANEXO 8. MARCO DE REFERENCIA PARA LA EVALUACIÓN DE LOS SERVICOS CONSTRUIDOS EN EL ESCENARIO DE ITERACIÓN

Para realizar la evaluación de los servicios prestados por el escenario de interacción semántica de objetos en la WoT, se realizó una revisión bibliográfica, encontrando que la medida más adecuada es la de calidad del servicio – "*Quality of Service*" - QoS en servicios de la IoT. Los trabajos más importantes se relacionan en los siguientes párrafos.

Gubbi, et al. [1] hacen referencia a las redes de la IoT como redes heterogéneas, no solo por su hardware sino también por el tipo de información transmitida y por ende, servicios diferentes con QoS diferentes; razón por la cual es difícil su medición. Proponen un sistema en el que el usuario pueda escoger los parámetros de calidad de servicio deseado. Basados es esta proyección, presentan una aplicación en la nube usando un middleware de servicio llamado Aneka, la cual se basa en la interacción de nubes públicas y privadas y permite la programación del QoS.

El planificador de Aneka es responsable de la asignación de cada recurso a una tarea en una aplicación para la ejecución sobre la base de parámetros de calidad de servicio de los usuarios y el costo global para el proveedor de servicios. Definen unos indicadores de QoS a ser tenidos en cuenta para la construcción del planificador:

- *Optimización multi-objetivo*: Los algoritmos de planificación deben ser capaces de hacer frente a los parámetros de calidad de servicio, tales como el **tiempo de respuesta**, el **costo de uso del servicio**, **número máximo de los recursos disponibles por precio unitario**, y las sanciones por la degradación del servicio.
- La duplicación de tareas de tolerancia a fallos basada en: tareas críticas de una aplicación se replican de forma transparente y ejecutados en diferentes recursos de manera que si un recurso no puede completar la tarea, la versión replicada la pueda utilizar. Esta lógica es crucial en las tareas en tiempo real que necesitan ser procesados para ofrecer servicios de manera oportuna.

Se concluye que la QoS en la computación en la nube es otra área de investigación importante que requerirá cada vez más atención ya que los datos y las herramientas de que se disponga en la nube.

Por otro lado, Zhou and Ma [2] integran los requisitos de la calidad de servicio en los servicios compuestos en la Internet de las cosas (IoT), y plantea métodos eficaces de descomposición y la optimización de los parámetros de calidad de servicio. Se analiza el complejo modelo de cálculo de calidad de servicio en cuatro modelos básicos y cada modelo se da un método computacional, los cuales son:

- Modo de serie: Los requisitos del sistema para los servicios {s1, s2, ..., sn} se realizará de acuerdo con un orden determinado.
- Modo paralelo: el sistema requiere que los servicios {s1 ... sn} ejecuten tareas al mismo tiempo.
- Modo Branch: En este modo la detección de {s1, s2, ..., sn} es el mismo en el modo serie como en el modo paralelo. Mientras una detección sea exitosa, el sistema puede completar la tarea. Este modo puede dividirse en dos modelos en el uso real.

- Modo de circulación: Los servicios {s1, s2, ..., sn} se están ejecutando en serie y se realiza cada ciclo de tiempo de acuerdo al ciclo de tiempo determinado, debido a que el servicio compuesto se construye a partir de múltiples servicios simples y disponibles, que se combinan de acuerdo a ciertas reglas y forman el nuevo servicio. Las métricas de QoS de estos sencillos servicios son diferentes entre sí; los autores proponen un método de cálculo que QoS adecuado para las características de la IoT, e hizo el análisis del algoritmo "*integer programming"* (IP) y la comparación con "genetic algorithm" (GA). Este trabajo deja claro que el algoritmo IP es más adecuado para los servicios de la IoT a gran escala. Se resalta la necesidad de mayor investigación para mejorar el algoritmo de cálculo de la QoS en la composición de servicios IoT en una variedad de escenarios de aplicación.

Otros autores como Jiong, et al. [3], plantean que la calidad de servicio depende de la aplicación que se esté evaluando, aseguran que el tráfico de datos para el monitoreo ambiental es elástica debido a que depende de la naturaleza, la cual tiende a tener un comportamiento continuo. Para estos casos se tiene como principal preocupación el ***ancho de banda***; aunque el retraso y pérdida de paquetes es tolerable hasta cierto punto.

Por lo anterior cuando se cuenta con la ayuda del usuario para realizar la detección, la calidad de servicio se define de acuerdo a la calidad de los datos aportados por el usuario. Así en el trabajo de Vithya and Vinayagasundaram [4], la mayor preocupación se centra en optimizar el costo de la red y en el uso energético de la misma. La prioridad se asigna sobre la base de una secuencia de eventos en las redes para evitar el retraso, la fluctuación de fase en el paquete multimedia, la energía de la batería del nodo y la reducción de la vida útil de la red. Para mejorar la QoS propone la utilización de algoritmos como el Agent based Priority Routing. Los cuales buscan ordenar de acuerdo a la prioridad en una cola con pequeñas modificaciones y ampliaciones. Los paquetes están configurados y alineados para lograr la mejor transmisión en el tiempo con baja latencia. La congestión del tráfico es evitado por el envío de un menor número de transmisiones de paquetes, esto es mediante la comparación de los paquetes y se envía el paquete sólo si hay alguna diferencia.

En Fok, et al. [5], se plantea que un conjunto estático de servicios y limitaciones no puede satisfacer todas las aplicaciones creadas en la IoT, en otras palabras la IoT cuenta con una QoS multi-dimensional, a menudo requiere el cumplimiento activo con ambos lados de una interacción y la propia red. En este trabajo, se revisan los desafíos de equilibrar las demandas de numerosas partes en una red muy diversa, dinámica e impredecible como el IoT. Este trabajo tiene como filosofía el descubrimiento y utilización de los recursos disponibles por parte de los usuarios.
Se considera explícitamente los requerimientos de cada punto final (es decir, la aplicación y el recurso) y la red que los conecta. Dado que la red es una de las partes interesadas, se consideran necesarios los nodos de interconexión.
La calidad de servicio - QoS es inherentemente específico de la aplicación. Por lo tanto, se necesita un mecanismo que permita la especificación de requisitos de QoS específicos de la aplicación mientras se mantiene servicio de aprovisionamiento general. En este trabajo, se utiliza una función QoS que convierte múltiples valores de entrada que representan limitaciones cuantificables pertinentes de cada actor en un solo valor de QoS cuya magnitud representa la inversa de la QoS totales previstos. Sólo las restricciones cuantificables son consideradas para simplificar nuestra exploración del dominio del problema.

El valor de QoS incorpora requisitos de los consumidores, proveedores y participantes de la red necesarios para apoyar la interacción. Los valores de entrada de ejemplo incluyen los dispositivos de eficiencia energética, la disponibilidad de energía, precisión del sensor, ancho de banda y la velocidad del procesador.

Utilizando el mismo mecanismo descrito anteriormente, también se tiene en cuenta las limitaciones de la red, como la disponibilidad de ancho de banda, capacidad de la red, y la latencia.

La energía caracteriza la cantidad de energía de un nodo consume para la interacción con relación a la cantidad de energía disponible. La latencia es la cantidad de retardo introducido por el nodo al servidor como un router o proveedor. Por último, la precisión cuantifica la capacidad del proveedor para prestar el servicio. En este caso, esta es la forma fiable que el servicio puede detectar una caída. La precisión sólo se define por nodos que proporcionan el servicio; otros nodos tienen una precisión de cero, lo que niega su impacto en el valor de calidad de servicio resultante.

En el trabajo de Nef, et al. [6] se plantea la evaluación de la QoS basada en tres factores que definen tres tipos de servicio, definidos en la norma IEEE 802.15.4:

- Interactividad: Se realiza mediante las consultas realizadas por los usuarios, no se evalúa la respuesta en tiempo real y tampoco la misión crítica.
- Retardo: Esta medición puede ser interactiva o no, se realiza en hard real time y en soft real time, esto lo define la suscripción del usuario.
- Criticalidad: No es interactivo ya que existe un flujo continuo de datos. Es en hard real time y en soft real time y es de misión crítica.

Para la simulación se implementó una red PAN, que contaba con un coordinado y N dispositivos de funciones reducidas. Se cuenta con una distancia entre nodos de 25 m, con la salvedad del impedimento del problema del nodo oculto, evitando las colisiones de los datos.

Se cuenta con dos tipos de usuario: single user y multiple user, se tiene como flujos de tráfico a los datos de la transmisión de un salto al coordinador. Se cuenta con escenarios multi saltos y una topología aleatoria. Cada simulación se demora 500 segundos y se repite 15 veces. El tamaño del paquete utilizado es de 40 bytes.

Finalmente se encontraron unas recomendaciones en calidad del servicio. La recomendación D4.2.2 plantea las métricas de calidad de servicio en la red sensor y aplicación, los cuales son:

- Calidad: de los dispositivos físicos.
- Consumo de energía: si es una red con limitaciones energéticas.
- Ancho de banda: hace referencia a la velocidad de los recursos disponibles o consumidos en bits por segundo.
- Volumen de datos: es la cantidad de datos producida por el sensor físico.
- La confiabilidad: hace referencia a la calidad del sensor físico.

En cuanto al sistema se tiene:

- Vida útil del sistema: en función de la vida útil de cada nodo.
- Latencia: es el máximo tiempo de retardo permitido para la comunicación de un mensaje.
- Calidad: parámetro determinado en la calidad de los datos facilitados en la respuesta a la consulta determinada.
- Retardo y variación del retardo: se mide cuando se realiza la recopilación de los datos de los nodos.
- Ancho de banda, capacidad y rendimiento: hace referencia a la capacidad de envío de mensajes en un tiempo de referencia.
- Optimización de recursos y rentabilidad: mide la capacidad del sistema para maximizar el bienestar social el cual se logra con una asignación eficiente de recursos.

La recomendación D5.1.2 plantea que debe realizarse la adquisición de datos basado en la utilidad. Este enfoque maximiza el bienestar social ya que se puede hacer uso óptimo de los objetos conectados a internet y de los datos proporcionados a la plataforma IoT. La norma D.4.6 sugiere un agente monitor de QoS independiente, con el fin de lograr:

1. Cobertura de detección del "*context aware*" y gestión de calidad de datos en la cual se requiere que los datos requeridos sean lo más similares a los entregados. La calidad de los datos se mide en términos de frecuencia y precisión de las lecturas del sensor dentro de una denominada área geográfica. Se requiere adquirir un número suficiente de lecturas de los sensores desde la activación de los mismos dentro de un intervalo de tiempo y en un área geográfica en particular.
2. Gestión de la eficiencia energética: implica maximizar la vida útil de la batería de cada nodo, lo que conlleva a minimizar el consumo de energía pero manteniendo la calidad de los datos.
3. Gestión de la latencia en la difusión de datos: es asegurar la entrega oportuna de los datos del sensor de acuerdo con los requisitos globales y en tiempo real.

El agente de monitorización de QoS sirve como el componente que controla el paso del tiempo de la demanda global de datos de sensor generados por el objeto móvil conectado a internet, además de gestionar el proceso de adquisición y garantizar la cobertura de detección deseada. Sus funciones son:

1. Gestión y seguimiento de suscripciones de QoS: determina los requisitos de la aplicación con respecto al sensor de adquisición de datos.
2. Gestión y seguimiento de las publicaciones de los sensores monitoreados: además de la gestión de datos adquiridos, optimización de la energía y ancho de banda consumidos.

El agente de monitorización de QoS debe contar con dos interfaces:

Servicios web y la de publicación suscripción. Estas recomendaciones se aplican mejor a un sistema con muchos recursos los cuales deben ser inalámbricos; su principal objetivo es la optimización de la energía y la creación de un algoritmo que le permita al sistema escoger los

recursos más confiables cuando se cree un nuevo servicio. Un recurso confiable es aquel que cuenta con niveles de batería altos, estable y que los datos sean precisos.

En cuanto al presente proyecto se tiene como la principal diferencia en el número de sensores desplegados para cada servicio, de n a 1 y con la fuente de alimentación constante para cada objeto inteligente (nodo); razón por la cual se descartan las medidas de eficiencia de los recursos y de energía.

Referencias Bibliográficas

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generation Computer Systems,* vol. 29, pp. 1645-1660, 2013.

[2] M. Zhou and Y. Ma, "QoS-aware computational method for IoT composite service," *The Journal of China Universities of Posts and Telecommunications,* vol. 20, Supplement 1, pp. 35-39, 8// 2013.

[3] J. Jiong, J. Gubbi, L. Tie, and M. Palaniswami, "Network architecture and QoS issues in the internet of things for a smart city," in *Communications and Information Technologies (ISCIT), 2012 International Symposium on*, 2012, pp. 956-961.

[4] G. Vithya and B. Vinayagasundaram, "QOS by Priority Routing in Internet of Things," *Research Journal of Applied Sciences,* vol. 8, pp. 2154-2160, 2014.

[5] C.-L. Fok, C. Julien, G.-C. Roman, and C. Lu, "Challenges of satisfying multiple stakeholders: quality of service in the internet of things," presented at the Proceedings of the 2nd Workshop on Software Engineering for Sensor Network Applications, Waikiki, Honolulu, HI, USA, 2011.

[6] M.-A. Nef, L. Perlepes, S. Karagiorgou, G. I. Stamoulis, and P. K. Kikiras, "Enabling qos in the internet of things," in *Proceedings of the Fifth International Conference on Communication Theory, Reliability, and Quality of Service (CTRQ 2012), Chamonix/Mont Blanc, France*, 2012, pp. 33-38.

ANEXO 9. PRIMERA PRUEBA DE FUNCIONALIDAD DEL ESCENARIO

Pruebas Beta: Para la realización de estas pruebas se emula el comportamiento de los sensores con potenciómetros, para obtener cambios drásticos en la entidad de interés en un corto intervalo de tiempo.

Fecha	Observación
05/11/2015	**Creación ECA servicio inteligente:** Creación exitosa, se encuentra el problema de ejecución; se enciende el ventilador sin cumplir la condición. También se observa que se requiere asociar capacidades de geo posicionamiento para la entidad de interés. Cuando se coloca en el requisito de búsqueda la entidad de interés el resultado presenta todos los recursos IoT asociados a este metadato. Se observó que es necesario presentarle al usuario la información del estado de la comunicación entre los objetos inteligentes, para los dos tipos de servicios ofrecidos por la solución, con una interfaz amigable al mismo. Del mismo modo se requiere la opción de visualizar el tiempo de cada comunicación, los agentes que intervienen en la misma. Se tiene como tiempo de respuesta, 11 segundos para realizar la activación y de 6,58 segundos para la desactivación.
	Caso de uso ver objeto: Esta opción se inicia con la identificación mediante un tag NFC, una vez autenticado Clipio presenta la información de los metadatos y la opción de indexar metadatos faltantes. Se observó que la información presentada no es amigable al usuario.
06/11/2015	**Geo posicionamiento**: El geo posicionamiento de la entidad de interés se encuentra almacenado en un tag NFC, el cual permite a Clipio la identificación de la misma. Para obtener el radio de cobertura de la búsqueda, mediante los datos brindados por la geo posición de los recursos, se implementó el sick bar, mediante las pruebas se observa que presenta resultados con una gran velocidad de respuesta. Se observó que cuando se realiza la solicitud utilizando como palabra clave los recursos el buscador no retorna ningún resultado, cuando la consulta se realiza colocando el nombre del objeto inteligente (Nombre del Feed en Xively) como palabra clave el buscador retorna todos los recursos asociados al Feed correspondiente.
07/11/2015	**Creación ECA I:** Se llevó a cabo la creación del ECA correspondiente al servicio de regulación de temperatura, en esta prueba se apreció que el tiempo estimado para tal proceso era superior a 12 segundos, dejando baja satisfacción en el usuario.
17/11/2015	**Creación ECA II:** En este apartado se busca medir el tiempo de creación de un ECA a partir de las modificaciones realizadas con anterioridad a los procesos de publicación y suscripción a los

Fecha	Observación
	bróker MQTT. El nuevo tiempo estimado para la creación de un ECA es de
18/11/2015	**Creación ECA en objeto inteligente Regulador de luz:** Implementación y prueba exitosa. Se creó el ECA correspondiente a la regulación de luz en la entidad de interés. El tiempo de respuesta para activar el servicio es de 16,505 segundos y de 6,928 para desactivarlo.
	Caso de uso eliminar servicio: Se creó un servicio con el fin de eliminarlo aun estando activo, la prueba fue exitosa, el tiempo aproximado de eliminación es de 5 segundos.
	Creación servicio de interacción: Se creó el servicio de interacción entre el objeto de regulación de temperatura y el de luz, tomando como evento a la temperatura y como acción el encender el bombillo. En este caso los dos objetos inteligentes tienen sus ECAs desactivados. El tiempo de respuesta es de 21.216 segundos para activarlos y 5,92 segundos para desactivarlo. Con los dos ECAs propios de los objetos inteligentes funcionando, pero no el ventilador se crea el ECA que enciende el ventilador a partir de un nivel de luz deseado, el tiempo de respuesta es de 6.984 segundos, los dos actuadores funcionan al mismo tiempo, después de 1.520 se apagó el calefactor y el ventilador para encenderse de nuevo a los 5.017 segundos después; una vez superado el impase inicial permanecen encendidos simultáneamente. Cuando desactivo el servicio de interacción se apagan los dos actuadores después de 6.948 segundos y transcurridos 2.505 segundos después se enciende el calefactor. Se creó el servicio de interacción entre el objeto de regulación de temperatura y el de luz, tomando como evento a la temperatura y como acción el apagar el bombillo. Para esta prueba los dos servicios de interacción están encendidos. Prueba fallida, ni siquiera permite la creación del ECA mediante Clipio; se reinicia el sistema, quitando la fuente de alimentación a los objetos inteligentes; debido el escenario almacena la última configuración establecida, se observa que el primer servicio de interacción en iniciar a funcionar es el de regulación de temperatura a los 75 segundos, mientras que el de regulación de luz inicia a funcionar a los 96 segundos a pesar de tener mayores tiempos de respuesta. En esta prueba se observa que no se puede realizar un ECA si el actuador ya está siendo utilizado por otro ECA.
	Creación ECA usando como evento un actuador: Se creó el servicio venti, el cual enciende el ventilador si el calefactor se encuentra encendido. Prueba exitosa.

Fecha	Observación
	Problema: Cuando se desactiva este servicio, el calefactor también se desactiva durante 3 segundos y se enciende de nuevo. Como este servicio depende de otro ECA, el cual es el encargado de encender el calefactor, cuando éste se desactiva, el ventilador también debe desactivarse, el tiempo de respuesta para esta prueba es de 19 segundos, cuando se activa el ECA desencadenante el tiempo de activación de este es de 19 segundos. El servicio dependiente se inicia luego de 19 segundos después de haberse encendido el calefactor.
	Creación de un ECA usando un sensor como acción: Es preciso probar todas las opciones que se le presentan al usuario, el ECA se inicia con el evento como un sensor o actuador que pertenece a otro objeto inteligente y como acción se toma el sensor de temperatura, Clipio presenta una alerta de error y no permite la creación del ECA. Lo mismo ocurre cuando el evento es un sensor o actuador que pertenece al mismo objeto.
27/11/2015	**Prueba a Clipio I:** Prueba exitosa, el sistema se congestionó cuando se crearon 37 ECAs diferentes y dejó de funcionar. Se realizó eliminación manual de todos los archivos xml tanto en Clipio como en el objeto inteligente y se reiniciaron los componentes del escenario.
27/11/2015	**Prueba a Clipio II**: Cuando se activa o desactiva un servicio, Clipio no almacena el estado actual del mismo. En la base de datos no queda almacenado el tiempo de respuesta de desactivación de los ECAs. Como tampoco el cambio de estado de la propiedad de interés, y el tiempo de respuesta del sistema para ejecutar el ECA correspondiente. Cuando Clipio no puede realizar una tarea, éste no presenta una notificación de fallo o error, simplemente se queda en la interfaz de espera, por tiempos incluso mayores a 4 minutos, cuando el usuario se cansa presiona el botón hacia atrás y tiene que volver a empezar la construcción del ECA. Estos datos no son reportados en la base de datos.

ANEXO 10. TAREAS BASE DE DATOS

10.1 Leer Clipio con el tag al objeto (9 registros)

Precondiciones:
1. El dispositivo móvil debe estar ejecutando Clipio en la actividad principal (MainActivy.java).
2. El tag debe estar anotado con el id correspondiente al objeto, el cual es el mismo que el del servidor middleware.

metadata_query: Permite consultar los metadatos de un objeto. Inicia cuando el usuario acerca un tag NFC al dispositivo móvil y termina cuando se le presenta la información de metadatos al usuario. Todos los objetos inteligentes están suscrito al canal coordinador.

Post condición: La interfaz de usuario entrega además de los metadatos del objeto la opción para activar o desactivar el servicio inteligente.

ID	Proceso
4311	Iniciando Proceso de Consulta De MetaDatos: lee el tag NFC. Prepara el xml correspondiente a la petición de metadatos a través del metadata_query, identificado con el id del objeto. Realiza la conexión con el bróker y permite inicializar los canales MQTT.
4312	Publicando Solicitud, Clipio publica el xml metadata_query en el bróker MQTT, mediante el canal id_objeto/coordinador. Esta acción se lleva a cabo hasta que el objeto inteligente asociado al tag NFC, le entregue la respuesta.
4313	Solicitud Recibida: El objeto reconoce el mensaje como metadata_query para este objeto.
4314	Publicando Respuesta; el objeto inteligente busca en su directorio MetaData, toma el MetaData.xml y lo publica en el bróker mediante el canal id_objeto/MetaData. Esta acción se lleva a cabo 10 veces para garantizar que el mensaje sea recibido.
4315	Procesando Respuesta, cuando Clipio recibe los metadatos y los presenta al usuario en su interfaz.
4316	Proceso De Consulta De Metadatos Terminado Con Éxito

10.2 Inicio servicio inteligente: set_basic_state

Precondición: Realizar la actividad anterior: metadata_query.
servicio_inteligente: Le permite al usuario activar o desactivar el servicio inteligente.

ID	**Proceso**
4417	Inicio Proceso De Cambio De Estado Del Servicio Inteligente, prepara el archivo SetBasicState.xml, correspondiente a la petición de cambio de estado del servicio inteligente. Realiza la conexión con el bróker y permite inicializar los canales MQTT.
4418	Publicando Solicitud, Clipio publica el archivo SetBasicState.xml en el bróker, mediante el canal id_objeto/coordinador. Esta acción se lleva a cabo hasta

ID	Proceso
	que el objeto inteligente al que se le hace el cambio de estado entregue la respuesta.
4419	Solicitud Recibida, El objeto reconoce el mensaje como solicitud de cambio para este objeto.
4420	Publicando Respuesta, el objeto inteligente crea un archivo llamado SimpleResponse.xml, el cual informa que la solicitud se ha recibido correctamente y lo publica en el bróker mediante el canal id_objeto/XXXX. Esta acción se lleva a cabo 10 veces para garantizar que el mensaje sea recibido.
4421	Procesando Respuesta, Clipio recibe el SimpleResponse.xml y cambia la interfaz, según la modificación realizada.
4422	Servicio Inteligente Modificado En Objeto, el estado asociado a la solicitud se ha modificado correctamente.
4423	Proceso De Cambio De Estado Del Servicio Inteligente Terminado Con Éxito

Post condición: Se mantiene la interfaz de usuario inicial, desde la tarea metadata_query, sólo se observa la actualización del estado del servicio inteligente.

10.3 Creación de un ECA: eca_gen

Temperatura riego: cuando la temperatura es mayor a 50 º, encender el riego.
-temperatura riego: apaga el riego cuando la temperatura es menor a 50º

Precondición: Seleccionar los recursos con los cuales se quiere realizar el ECA, para lo cual Clipio realiza el proceso de metadata_query antes de presentarle la interfaz de usuario para definir la condición.

Proceso
Iniciando Proceso De Creación y Publicación De Un ECA, Clipio prepara el archivo NombreECA.xml, este archivo toma su nombre de acuerdo a la solicitud ingresada por el usuario. Inicializa los hilos para el manejo del objeto: evento y acción.
Iniciando Publicación En El Objeto Evento; Realiza la conexión con el bróker y permite inicializar los canales MQTT con el objeto evento.
Iniciando Publicación En El Objeto Acción, Realiza la conexión con el bróker y permite inicializar los canales MQTT con el objeto acción.
Publicando Solicitud En El Objeto Evento, Clipio publica el archivo NombreECA.xml en el bróker, mediante el canal id_objeto_evento/coordinador. Esta acción se lleva a cabo hasta que el objeto evento que recibe el ECA, entregue la respuesta
Publicando Solicitud En El Objeto Acción, Clipio publica el archivo NombreECA.xml en el bróker, mediante el canal id_objeto_accion/coordinador. Esta acción se lleva a cabo hasta que el objeto acción que recibe el ECA, entregue la respuesta.
Solicitud Recibida, El objeto evento reconoce el mensaje como solicitud de creación de ECA para este objeto.
Solicitud Recibida, El objeto acción reconoce el mensaje como solicitud de creación de ECA para este objeto.
Publicando Respuesta, el objeto evento crea un archivo llamado SimpleResponse.xml, el cual informa que la solicitud se ha recibido correctamente y lo publica en el bróker mediante el canal id_objeto_evento/XXXXX. Esta acción se lleva a cabo 10 veces para garantizar que el mensaje sea recibido.
ECA Creado En Objeto, el ECA recibido se crea exitosamente en el objeto evento.
Procesando Respuesta Del Objeto Acción, Clipio recibe el SimpleResponse.xml del

Proceso
objeto acción.
Publicando Respuesta, el objeto acción crea un archivo llamado SimpleResponse.xml, el cual informa que la solicitud se ha recibido correctamente y lo publica en el bróker mediante el canal id_objeto_accion/XXXXX. Esta acción se lleva a cabo 10 veces para garantizar que el mensaje sea recibido.
ECA Creado En Objeto, el ECA recibido se crea exitosamente en el objeto acción.
Procesando Respuesta Del Objeto Evento, Clipio recibe el SimpleResponse.xml del objeto evento.
Confirmada Respuesta Del Objeto Acción, Clipio confirma la creación ECA en el objeto acción y finaliza el hilo.
Confirmada Respuesta Del Objeto Evento, Clipio confirma la creación ECA en el objeto evento y finaliza el hilo.
Proceso De Creación y Publicación De Un ECA Terminado Con Éxito

Post condición
Clipio presenta al usuario una notificación emergente de confirmación exitosa de la creación del ECA.

Para que el ECA funcione se debe realizar su activación.

Precondiciones:
Se debe ingresar a la interfaz: detalleECA.
Antes de activar un ECA, es preciso crearlo con anterioridad.
Activación ECA: Permite colocar en funcionamiento un ECA.
set_eca_state
Establecer Enlace, el objeto acción se suscribe al canal del objeto evento; id_objeto_evento/id_recurso

Esto aparece en el panel log

Proceso
Iniciando Proceso De Cambio De Estado De Un ECA, Clipio prepara el archivo SetECAState.xml. Inicializa los hilos para el manejo del objeto: evento y acción.
Iniciando Publicación En Objeto Evento, Realiza la conexión con el bróker y permite inicializar los canales MQTT con el objeto evento
Iniciando Publicación En Objeto Acción, Realiza la conexión con el bróker y permite inicializar los canales MQTT con el objeto acción.
Publicando Solicitud En Objeto Evento, Clipio publica el archivo SetECAState.xml en el bróker, mediante el canal id_objeto_evento/coordinador. Esta acción se lleva a cabo hasta que el objeto evento que recibe la solicitud, entregue la respuesta
Publicando Solicitud En Objeto Acción, Clipio publica el archivo SetECAState.xml en el bróker, mediante el canal id_objeto_accion/coordinador. Esta acción se lleva a cabo hasta que el objeto acción que recibe la solicitud, entregue la respuesta.
Solicitud Recibida, El objeto evento reconoce el mensaje como solicitud de cambio de estado del ECA para este objeto.
Solicitud Recibida, El objeto acción reconoce el mensaje como solicitud de cambio de estado del ECA para este objeto.
Publicando Respuesta, el objeto evento crea un archivo llamado SimpleResponse.xml, el cual informa que la solicitud se ha recibido correctamente y lo publica en el bróker

Proceso
mediante el canal id_objeto_evento/XXXXX. Esta acción se lleva a cabo 10 veces para garantizar que el mensaje sea recibido.
Publicando Respuesta el objeto acción crea un archivo llamado SimpleResponse.xml, el cual informa que la solicitud se ha recibido correctamente y lo publica en el bróker mediante el canal id_objeto_accion/XXXXX. Esta acción se lleva a cabo 10 veces para garantizar que el mensaje sea recibido.
ECA Modificado En Objeto, la solicitud recibida fue modificada exitosamente en el objeto evento.
Procesando Respuesta Del Objeto Evento, Clipio recibe el SimpleResponse.xml del objeto evento.
ECA Modificado En Objeto, la solicitud recibida fue modificada exitosamente en el objeto acción.
Procesando Respuesta Del Objeto Acción, Clipio recibe el SimpleResponse.xml del objeto acción.
Confirmada Respuesta Del Objeto Acción, Clipio confirma la modificación del ECA en el objeto acción y finaliza el hilo.
Confirmada Respuesta Del Objeto Evento, Clipio confirma la modificación del ECA en el objeto evento y finaliza el hilo.
Proceso De Cambio De Estado De Un ECA Terminado Con Éxito

Post condición: la interfaz detalleECA presenta la actualización del estado del ECA.

Eliminar ECA

Precondición:
Para acceder a esta funcionalidad es preciso ingresar a la interfaz, detalleECA.
Eliminar ECA
Eca_delete

Proceso
Iniciando Proceso De Eliminación De Un ECA, Clipio prepara el archivo ECAdelete.xml. Inicializa los hilos para el manejo del objeto: evento y acción.
Iniciando Publicación En Objeto Evento, Realiza la conexión con el bróker y permite inicializar los canales MQTT con el objeto evento
Iniciando Publicación En Objeto Acción, Realiza la conexión con el bróker y permite inicializar los canales MQTT con el objeto acción.
Publicando Solicitud En Objeto Evento, Clipio publica el archivo ECAdelete.xml en el bróker, mediante el canal id_objeto_evento/coordinador. Esta acción se lleva a cabo hasta que el objeto evento que recibe la solicitud, entregue la respuesta
Publicando Solicitud En Objeto Acción, Clipio publica el archivo ECAdelete.xml en el bróker, mediante el canal id_objeto_accion/coordinador. Esta acción se lleva a cabo hasta que el objeto acción que recibe la solicitud, entregue la respuesta.
Solicitud Recibida, El objeto evento reconoce el mensaje como solicitud de eliminación del ECA para este objeto.
Solicitud Recibida, El objeto acción reconoce el mensaje como solicitud de eliminación del ECA para este objeto.
Publicando Respuesta, el objeto evento crea un archivo llamado SimpleResponse.xml, el cual informa que la solicitud se ha recibido correctamente y lo publica en el bróker mediante el canal id_objeto_evento/XXXXX. Esta acción se lleva a cabo 10 veces para garantizar que el mensaje sea recibido.
Publicando Respuesta el objeto acción crea un archivo llamado SimpleResponse.xml, el cual informa que la solicitud se ha recibido correctamente y lo publica en el bróker mediante el canal id_objeto_accion/XXXXX. Esta acción se lleva a cabo 10 veces para

Proceso
garantizar que el mensaje sea recibido.
ECA Eliminado En Objeto, la solicitud recibida fue ejecutada exitosamente en el objeto evento.
Procesando Respuesta Del Objeto Evento, Clipio recibe el SimpleResponse.xml del objeto evento.
ECA Eliminado En Objeto, la solicitud recibida fue ejecutada exitosamente en el objeto acción.
Procesando Respuesta Del Objeto Acción, Clipio recibe el SimpleResponse.xml del objeto acción.
Confirmada Respuesta Del Objeto Acción, Clipio confirma la eliminación del ECA en el objeto acción y finaliza el hilo.
Confirmada Respuesta Del Objeto Evento, Clipio confirma la eliminación del ECA en el objeto evento y finaliza el hilo.
Proceso De Eliminar Un ECA Terminado Con Exito

post condición: Se presenta una notificación emergente de la eliminación del ECA.

ANEXO 11. ITERACIONES PRUEBA DE FUNCIONAMIENTO DEL ESCENARIO

A continuación se llevó a cabo una serie de interacciones para realizar la prueba del funcionamiento del escenario:

Prueba realizada el 26 de Noviembre del 2015

Para la fase de la evaluación del prototipo del escenario, se llevó a cabo en las instalaciones de la universidad del Cauca, en la oficina 422. La conexión a la red se realiza mediante el punto de acceso ubicado en esta oficina, el cual no está dedicada a la realización de las pruebas, las velocidades son de 9.76 Mbps para descarga y de 17,9 Mbps para la velocidad de carga; la red es compartida aproximadamente con 70 usuarios constantes.

Se construyó un ECA, tal que si la intensidad de luz era superior a 20 lux, se debía encender la lámpara y si la intensidad de luz es inferior a 30 lux, la lámpara debía apagarse, en el intervalo de 20 a 30, el sistema se obedecía las dos condiciones, pero después de unos instantes de tiempo, el sistema trabaja con la última condición cumplida.

Existe problemas para eliminar un ECA de la interfaz de Clipio, aparecen listados ECAs ya eliminados. Incluso para eliminar un ECA, el sistema se queda en la interfaz de proceso en ejecución. Después de crear 30 ECAs el sistema no permite creación ni eliminación de los mismos, para superar este impase se recurrió a la eliminación manual, tanto de la aplicación Android como de cada objeto inteligente. La mayor cantidad de ECAs almacenados por un objeto antes de bloqueo del sistema es de 17.

Como el problema para la creación del ECA persiste, se desinstaló la aplicación, se borraron sus datos y se la reinstaló, pero el problema persistía, se probó como última medida reiniciar los objetos inteligentes para actualizar su carpeta de ECAs, el sistema no permite la creación de un nuevo ECA.

Las interacciones planteadas se presentan en la siguiente tabla, para el funcionamiento del servicio que requiere más de una condición, es preciso realizar un ECA para cada condición.

Servicio	Evento	Condición	Acción	
			Actuador	Estado
1	Temperatura	T>50 ºC	Riego	ON
	Riego	ON	Ventilador	ON
2	Luz	L<30	Calefactor	ON
	Temperatura	T<50 ºC	Bombillo	ON
3	Humedad	H>60%	Ventilador	ON
	Ventilador	ON	Riego	ON
4	Servicio regulador de temperatura activo y se toma como condición a la temperatura	T>10	Riego	ON
	Servicio regulador de temperatura activo y se toma como condición a la temperatura	T<10	Riego	OFF
5	Servicio regulador de humedad activo y se toma como condición a la humedad	h>50	Ventilador	ON

Servicio	Evento	Condición	Acción	
			Actuador	Estado
5.1	Servicio regulador de humedad activo y se toma como condición a la humedad	h<50	Calefactor	ON
6	Temperatura	t>50 ºC	Riego	ON
6.1	Humedad	H>60%	Ventilador	ON
6.2	Riego	ON	Calefactor	ON

Como resultado se observó que no existe variación de tiempos para la creación activación y desactivación de ECAs, así como también se observó que si un ECA tiene como evento un actuador, éste debe esperar que se cumpla la condición de cambio del actuador para pasar a realizar el siguiente ECA, en conclusión, el tiempo de respuesta no cambia, solo se percibe como mayor debido a que se requiere cumplir dos condiciones, para percibir el cambio.

ANEXO 12. PRUEBA PARA LAS ITERACIONES

Para las pruebas realizadas del 28 de Noviembre al 02 de diciembre del 2015, se analizaron todas las posibles combinaciones para llevar a cabo la interacción entre objetos inteligentes en la web de las cosas. Como se cuenta con cuatro actuadores en tres objetos diferentes, para realizar la interacción, se toma el actuador de otro objeto para activar uno de los dos actuadores que corresponden al mismo objeto inteligente, con el fin de evitar que la comunicación se lleve a cabo entre ellos, aunque el sistema lo permite y el tiempo de respuesta es el mismo.

Para la primera parte de la prueba se deja activo uno de los servicios inteligentes y se toma los actuadores de los otros dos para realizar las posibles combinaciones, tomando como evento tanto al sensor como el o los actuadores. La siguiente fase se crea interacciones en cascada, tomando cada uno de los sensores como evento y en los casos que correspondan los actuadores, pero con la condición de que sea un sensor el que cambia su estado.

Estas pruebas se llevaron a cabo en una red domiciliaria, externa a las instalaciones de la universidad del Cauca, con un ancho de banda de 5 Megas, una velocidad de bajada de 400 Kbps y una velocidad de subida de 1 M. como también se realizaron pruebas conectados a la red de la oficina 422, la cual es compartida.

Nombre Interacción	Evento	Condición	Acción	Estado
Servicios inteligentes activos				
Servicio regulador de temperatura				
01_servint_tem_may_rie	Temperatura	>50	Riego	ON
01_servint_tem_men_rie	Temperatura	<50	Riego	OFF
02_servint_tem_may_bom	Temperatura	>50	Bombillo	ON
02_servint_tem_may_bom	Temperatura	<50	Bombillo	OFF
03_servint_cal_on_rie	Calefactor	ON	Riego	ON
03_servint_cal_off_rie	Calefactor	OFF	Riego	OFF
04_servint_cal_on_bom	Calefactor	ON	Bombillo	ON
04_servint_cal_off_bom	Calefactor	OFF	Bombillo	OFF
05_servint_ven_on_rie	Ventilador	ON	Riego	ON
05_servint_ven_off_rie	Ventilador	OFF	Riego	OFF
06_servint_ven_on_bom	Ventilador	ON	Bombillo	ON
06_servint_ven_off_bom	Ventilador	OFF	Bombillo	OFF
Servicio regulador de humedad				
07_servint_hum_may_cal	Humedad	>50	Calefactor	ON
07_servint_hum_men_cal	Humedad	<50	Calefactor	OFF
08_servint_hum_men_ven	Humedad	<50	Ventilador	ON
08_servint_hum_may_ven	Humedad	>50	Ventilador	OFF
09_servint_hum_may_bom	Humedad	>50	Bombillo	ON
09_servint_hum_men_bom	Humedad	<50	Bombillo	OFF
10_servint_rie_on_cal	Riego	ON	Calefactor	ON
10_servint_rie_off_cal	Riego	OFF	Calefactor	OFF
11_servint_rie_on_ven	Riego	ON	Ventilador	ON
11_servint_rie_off_ven	Riego	OFF	Ventilador	OFF
12_servint_rie_on_bom	Riego	ON	Bombillo	ON
12_servint_rie_off_bom	Riego	OFF	Bombillo	OFF
Servicio regulador de luz				
13_servint_luz_may_cal	Luz	>50	Calefactor	ON
13_servint_luz_men_cal	Luz	<50	Calefactor	OFF

Nombre Interacción	Evento	Condición	Acción	Estado
14_servint_luz_may_rie	Luz	>50	Riego	ON
14_servint_luz_men_rie	Luz	<50	Riego	OFF
15_servint_luz_may_ven	Luz	>50	Ventilador	ON
15_servint_luz_men_ven	Luz	<50	Ventilador	OFF
16_servint_bom_on_cal	Bombillo	ON	Calefactor	ON
16_servint_bom_off_cal	Bombillo	OFF	Calefactor	OFF
17_servint_bom_on_ven	Bombillo	ON	Ventilador	ON
17_servint_bom_off_ven	Bombillo	OFF	Ventilador	OFF
18_servint_bom_on_rie	Bombillo	ON	Riego	ON
18_servint_bom_off_rie	Bombillo	OFF	Riego	OFF
Interacciones en cascada				
1:1				
Servicio regulador de temperatura				
19_cas_tem_may_rie	Temperatura	>50	Riego	ON
19_cas_tem_men_rie	Temperatura	<50	Riego	OFF
20_cas_tem_may_bom	Temperatura	>50	Bombillo	ON
20_cas_tem_men_bom	Temperatura	<50	Bombillo	OFF
Servicio regulador de humedad				
21_cas_hum_may_cal	Humedad	>50	Calefactor	ON
21_cas_hum_men_cal	Humedad	<50	Calefactor	OFF
22_cas_hum_may_ven	Humedad	>50	Ventilador	ON
22_cas_hum_men_ven	Humedad	<50	Ventilador	OFF
23_cas_hum_may_bom	Humedad	>50	Bombillo	ON
23_cas_hum_men_bom	Humedad	<50	Bombillo	OFF
Servicio regulador de luz				
24_cas_luz_may_cal	Luz	>50	Calefactor	ON
24_cas_luz_men_cal	Luz	<50	Calefactor	OFF
24_cas_luz_may_ven	Luz	>50	Riego	ON
24_cas_luz_men_ven	Luz	<50	Riego	OFF
24_cas_luz_may_bom	Luz	>50	Ventilador	ON
24_cas_luz_men_bom	Luz	<50	Ventilador	OFF
1:2				
25_1_a_cas_tem_may_rie	Temperatura	>50	Riego	ON
25_1_a_cas_tem_men_rie	Temperatura	<50	Riego	OFF
25_1_b_cas_rie_on_bom	Riego	ON	Bombillo	ON
25_1_b_cas_rie_off_bom	Riego	OFF	Bombillo	OFF
25_2_a_cas_tem_may_bom	Temperatura	>50	Bombillo	ON
25_2_a_cas_tem_men_bom	Temperatura	<50	Bombillo	OFF
25_2_b_cas_bom_on_rie	Bombillo	ON	Riego	ON
25_2_a_cas_bom_off_rie	Bombillo	OFF	Riego	OFF
26_1_a_cas_hum_may_cal	Humedad	>50	Calefactor	ON
26_1_a_cas_hum_men_cal	Humedad	<50	Calefactor	OFF
26_1_b_cas_cal_on_bom	Calefactor	ON	Bombillo	ON
26_1_b_cas_cal_off_bom	Calefactor	OFF	Bombillo	OFF
26_2_a_cas_hum_may_bom	Humedad	>50	Bombillo	ON
26_2_a_cas_hum_men_bom	Humedad	<50	Bombillo	OFF
26_2_b_cas_bom_on_cal	Bombillo	ON	Calefactor	ON

Nombre Interacción	Evento	Condición	Acción	Estado
26_2_b_cas_bom_off_cal	Bombillo	OFF	Calefactor	OFF
26_2_c_cas_bom_on_ven	Bombillo	ON	Ventilador	ON
26_2_c_cas_bom_off_ven	Bombillo	OFF	Ventilador	OFF
26_3_a_cas_hum_may_ven	Humedad	>50	Ventilador	ON
26_3_a_cas_hum_men_ven	Humedad	<50	Ventilador	OFF
26_3_b_cas_ven_on_bom	Ventilador	ON	Bombillo	ON
26_3_b_cas_ven_off_bom	Ventilador	OFF	Bombillo	OFF
26_3_c_cas_ven_on_cal	Ventilador	ON	Calefactor	ON
26_3_c_cas_ven_off_cal	Ventilador	OFF	Calefactor	OFF
27_1_a_cas_luz_may_cal	Luz	>50	Calefactor	ON
27_1_a_cas_luz_men_cal	Luz	<50	Calefactor	OFF
27_1_b_cas_cal_on_rie	Calefactor	ON	Riego	ON
27_1_b_cas_cal_off_rie	Calefactor	OFF	Riego	OFF
27_1_b_cas_cal_on_ven	Calefactor	ON	Ventilador	ON
27_1_b_cas_cal_of_ven	Calefactor	OFF	Ventilador	OFF
27_2_a_cas_luz_may_rie	Luz	>50	Riego	ON
27_2_a_cas_luz_men_rie	Luz	<50	Riego	OFF
27_2_b_cas_rie_on_cal	Riego	ON	Calefactor	ON
27_2_b_cas_rie_off_cal	Riego	OFF	Calefactor	OFF
27_2_b_cas_rie_on_ven	Riego	ON	Ventilador	ON
27_2_b_cas_rie_off_ven	Riego	OFF	Ventilador	OFF
27_3_a_cas_luz_may_ven	Luz	>50	Ventilador	ON
27_3_a_cas_luz_men_ven	Luz	<50	Ventilador	OFF
27_3_b_cas_ven_on_rie	Ventilador	ON	Riego	ON
27_3_b_cas_ven_off_rie	Ventilador	OFF	Riego	OFF
27_3_b_cas_ven_on_cal	Ventilador	ON	Calefactor	ON
27_3_b_cas_ven_off_cal	Ventilador	OFF	Calefactor	OFF
1:3				
28_1_a_cas_tem_may_bom	Temperatura	>50	Bombillo	ON
28_1_a_cas_tem_men_bom	Temperatura	<50	Bombillo	OFF
28_1_b_cas_bom_on_ven	Bombillo	ON	Ventilador	ON
28_1_b_cas_bom_off_ven	Bombillo	OFF	Ventilador	OFF
28_1_c_cas_ven_on_rie	Ventilador	ON	Riego	ON
28_1_c_cas_ven_off_rie	Ventilador	OFF	Riego	OFF
28_2_a_cas_tem_may_rie	Temperatura	>50	Riego	ON
28_2_a_cas_tem_men_rie	Temperatura	<50	Riego	OFF
28_2_b_cas_rie_on_bom	Riego	ON	Bombillo	ON
28_2_b_cas_rie_off_bom	Riego	OFF	Bombillo	OFF
28_2_c_cas_bom_on_ven	Bombillo	ON	Ventilador	ON
28_2_c_cas_bom_off_ven	Bombillo	OFF	Ventilador	OFF
29_1_a_cas_luz_may_cal	Luz	>50	Calefactor	ON
29_1_a_cas_luz_men_cal	Luz	<50	Calefactor	OFF
29_1_b_cas_cal_on_bom	Calefactor	ON	Riego	ON
29_1_b_cas_cal_off_bom	Calefactor	OFF	Riego	OFF
29_1_c_cas_bom_on_ven	Riego	ON	Ventilador	ON
29_1_c_cas_bom_off_ven	Riego	OFF	Ventilador	OFF
29_2_a_cas_luz_may_ven	Luz	>50	Ventilador	ON

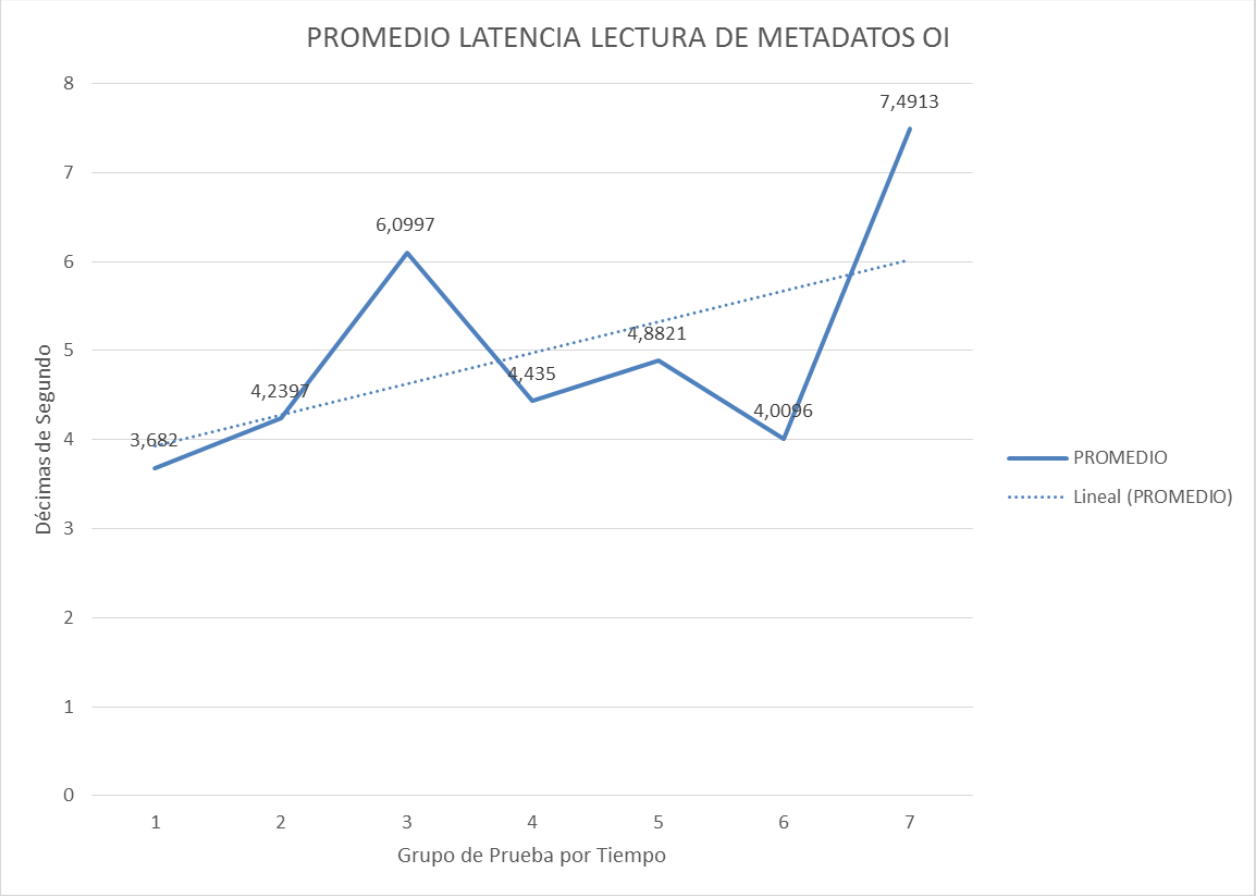
Nombre Interacción	Evento	Condición	Acción	Estado
29_2_a_cas_luz_men_ven	Luz	<50	Ventilador	OFF
29_2_b_cas_ven_on_rie	Ventilador	ON	Riego	ON
29_2_b_cas_ven_off_rie	Ventilador	OFF	Riego	OFF
29_2_c_cas_rie_on_cal	Riego	ON	Calefactor	ON
29_2_c_cas_rie_off_cal	Riego	OFF	Calefactor	OFF
1:4				
30_a_cas_hum_may_cal	Humedad	>50	Calefactor	ON
30_a_cas_hum_men_cal	Humedad	<50	Calefactor	OFF
30_b_cas_cal_on_bom	Calefactor	ON	Bombillo	ON
30_b_cas_cal_off_bom	Calefactor	OFF	Bombillo	OFF
30_c_cas_bom_on_ven	Bombillo	ON	Ventilador	ON
30_c_cas_bom_off_ven	Bombillo	OFF	Ventilador	OFF
30_d_cas_ven_on_rie	Ventilador	ON	Riego	ON
30_d_cas_ven_off_rie	Ventilador	OFF	Riego	OFF

ANEXO 13. EVALUACIÓN DEL ESCENARIO DE ITERACIÓN SEMÁNTICA EN LA IOT

13.1 Prueba- 1

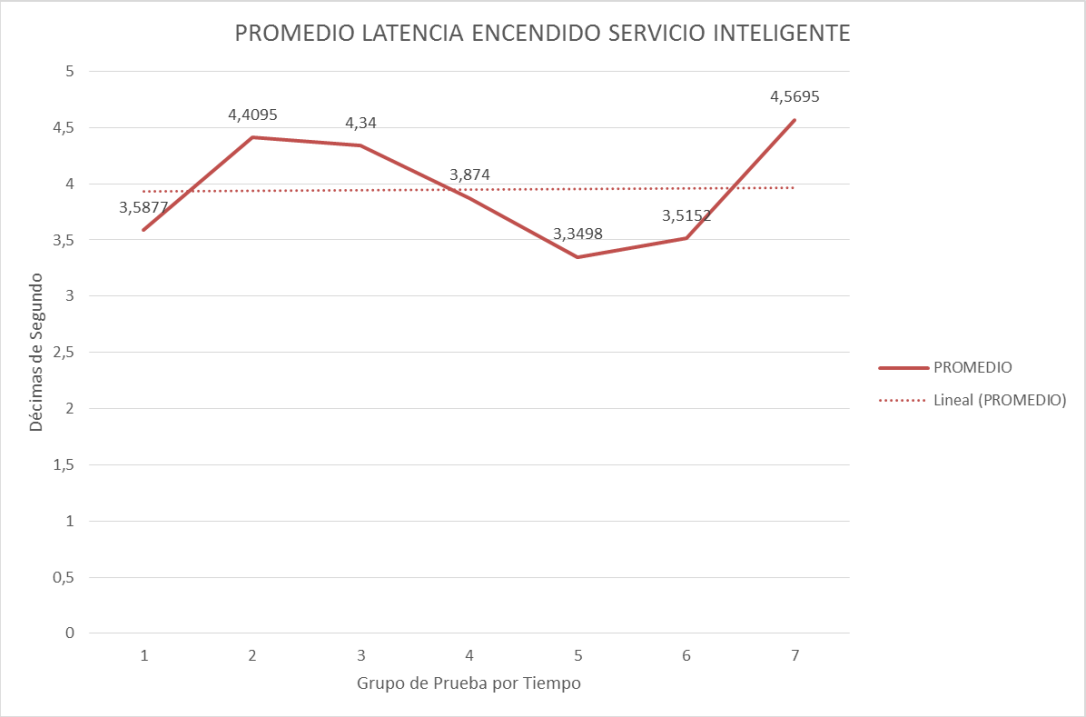
Latencia Promedio Lectura de Metadatos de Objetos Inteligentes

FECHA	Grupo de Prueba	PROMEDIO
21/11/2015 0:00	1	3,682
23/11/2015 0:00	2	4,2397
24/11/2015 0:00	3	6,0997
25/11/2015 0:00	4	4,435
26/11/2015 0:00	5	4,8821
27/11/2015 0:00	6	4,0096
28/11/2015 0:00	7	7,4913
	TOTAL	4,7242



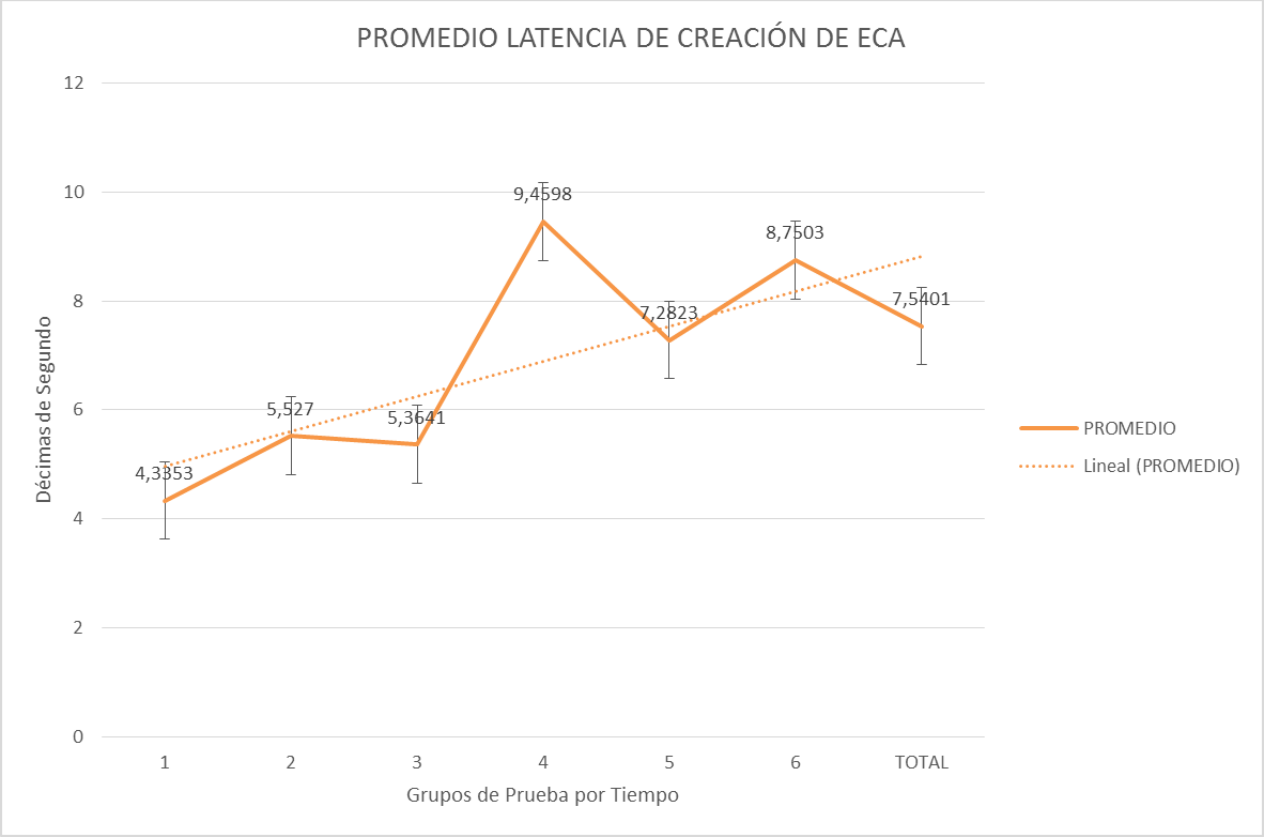
Latencia Promedio Encendiendo el Servicio Inteligente

FECHA	Grupo de Prueba	PROMEDIO
21/11/2015 0:00	1	3,5877
23/11/2015 0:00	2	4,4095
24/11/2015 0:00	3	4,34
25/11/2015 0:00	4	3,874
26/11/2015 0:00	5	3,3498
27/11/2015 0:00	6	3,5152
28/11/2015 0:00	7	4,5695
	TOTAL	**3,7651**



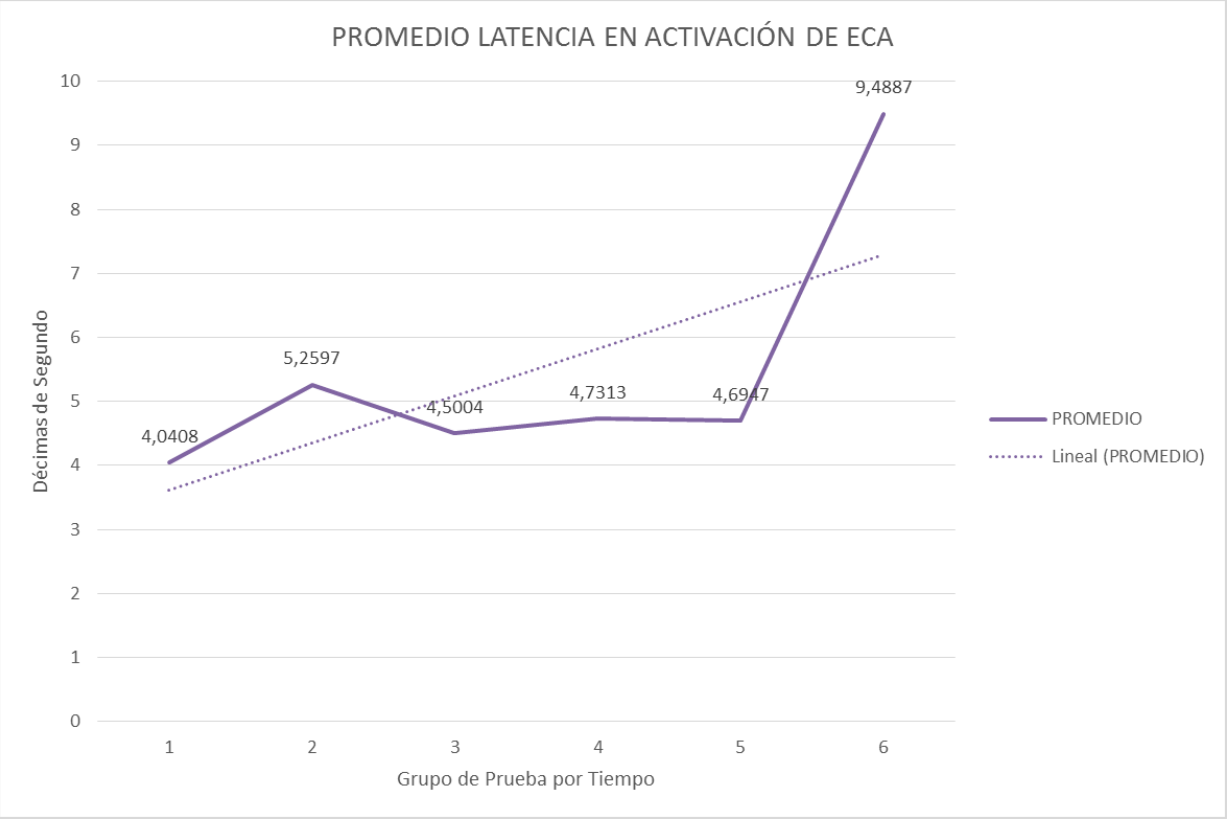
Latencia Promedio Creación ECA

FECHA	Grupo de Prueba	PROMEDIO
21/11/2015 0:00	1	4,3353
23/11/2015 0:00	2	5,527
25/11/2015 0:00	3	5,3641
26/11/2015 0:00	4	9,4598
27/11/2015 0:00	5	7,2823
28/11/2015 0:00	6	8,7503
	TOTAL	**7,5401**



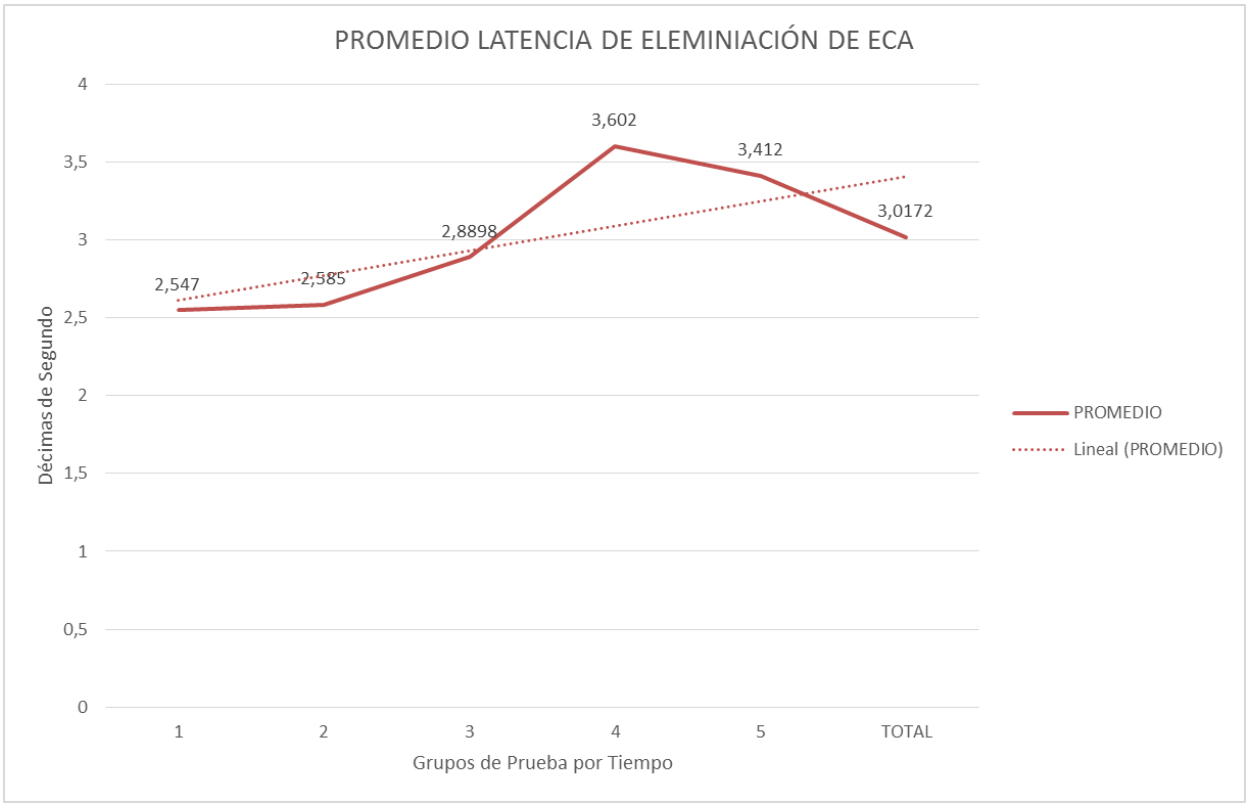
Latencia Promedio de Activación de ECA

FECHA	Grupo de Prueba	PROMEDIO
21/11/2015 0:00	1	4,0408
23/11/2015 0:00	2	5,2597
25/11/2015 0:00	3	4,5004
26/11/2015 0:00	4	4,7313
27/11/2015 0:00	5	4,6947
28/11/2015 0:00	6	9,4887
	TOTAL	**4,8128**



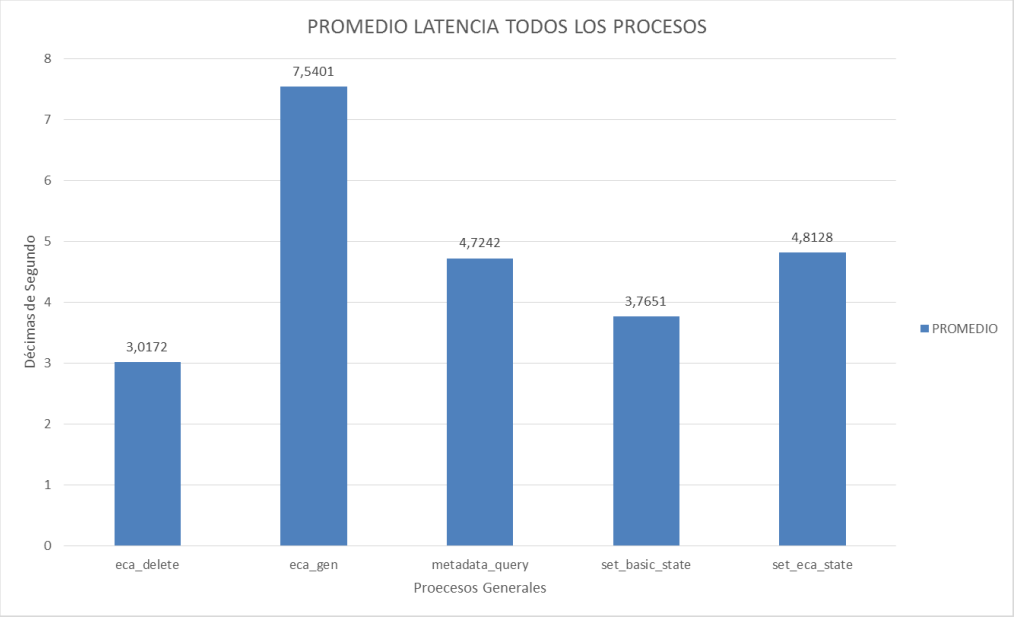
Latencia Promedio de Eliminación de ECA

FECHA	Gupo de Prueba	PROMEDIO
24/11/2015 0:00	1	2,547
25/11/2015 0:00	2	2,585
26/11/2015 0:00	3	2,8898
27/11/2015 0:00	4	3,602
28/11/2015 0:00	5	3,412
	TOTAL	**3,0172**



Promedio Latencia Todos los Procesos

INDICADOR	PROMEDIO
eca_delete	3,0172
eca_gen	7,5401
metadata_query	4,7242
set_basic_state	3,7651
set_eca_state	4,8128
TOTAL	**5,0535**



13.2 Prueba 2

Latencia Promedio Lectura de Metadatos de Objetos Inteligentes

FECHA	Grupo de Prueba	PROMEDIO
29/11/2015 0:00	1	3,605
	TOTAL	**3,605**

Latencia Promedio Encendiendo el Servicio Inteligente

FECHA	Grupo de Prueba	PROMEDIO
29/11/2015 0:00	1	3,737
	TOTAL	**3,737**

Latencia Promedio Creación ECA

FECHA	Grupo de Prueba	PROMEDIO

Latencia Promedio de Activación de ECA

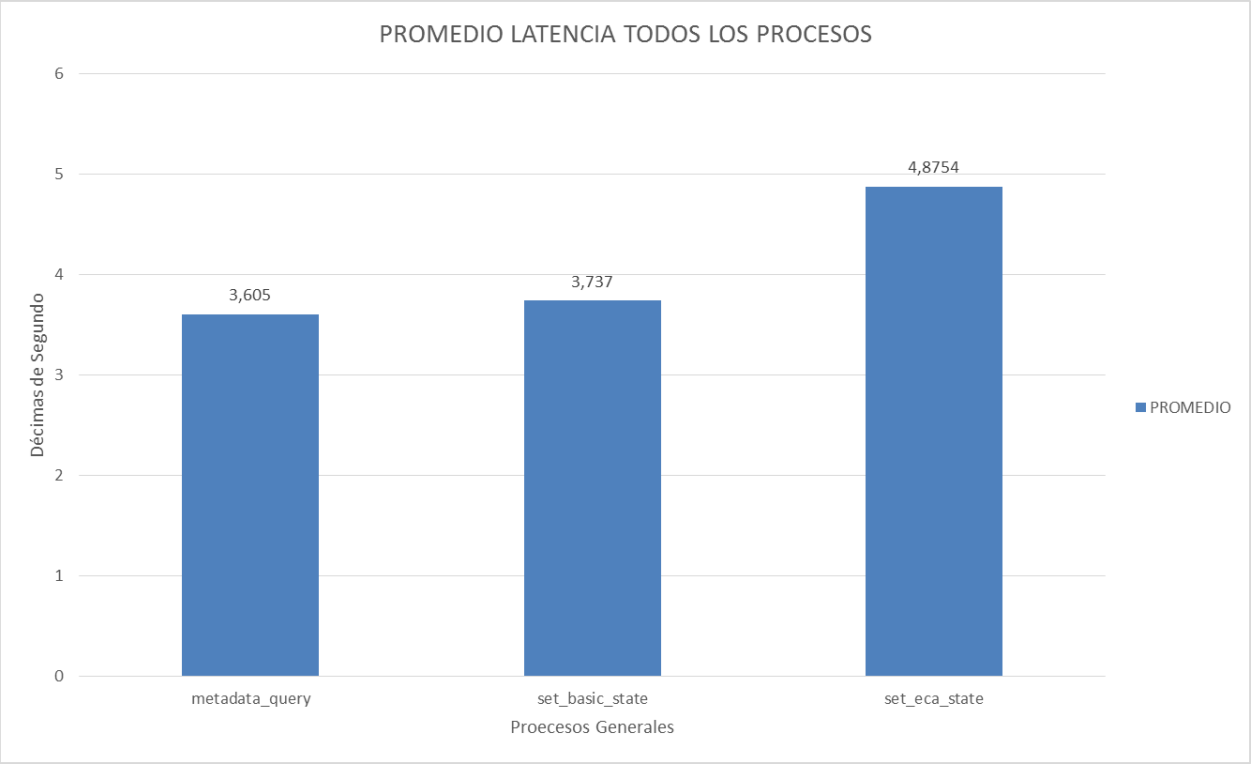
FECHA	Grupo de Prueba	PROMEDIO
29/11/2015 0:00	1	4,8754
	TOTAL	**4,8754**

Latencia Promedio de Eliminación de ECA

FECHA	Gupo de Prueba	PROMEDIO

Promedio Latencia Todos los Procesos

INDICADOR	PROMEDIO
metadata_query	3,605
set_basic_state	3,737
set_eca_state	4,8754
TOTAL	**4,4421**



13.3 Prueba 3

Latencia Promedio Lectura de Metadatos de Objetos Inteligentes

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	3,741
1/12/2015 0:00	2	3,9303
	TOTAL	3,8167

Latencia Promedio Encendiendo el Servicio Inteligente

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	3,6062
1/12/2015 0:00	**2**	**3,621**
	TOTAL	3,6087

Latencia Promedio Creación ECA

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	5,5208
1/12/2015 0:00	2	5,7134
	TOTAL	5,6629

Latencia Promedio de Activación de ECA

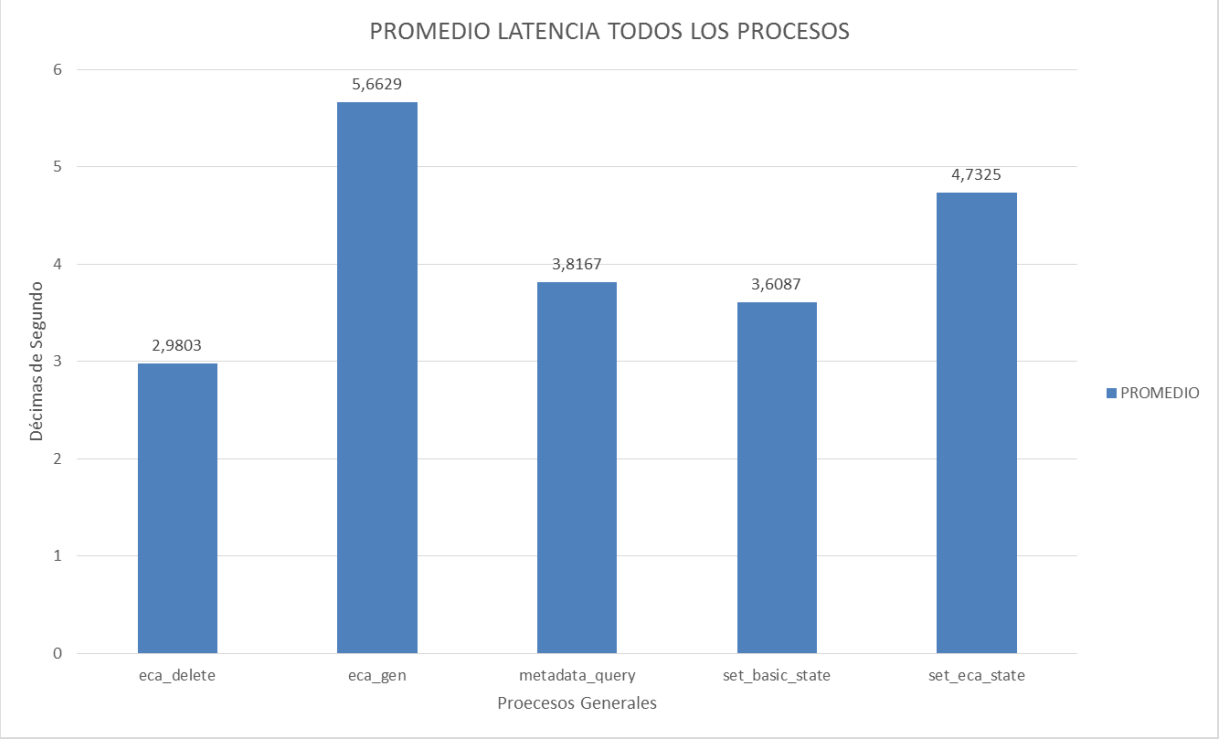
FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	5,0884
1/12/2015 0:00	**2**	**4,2529**
	TOTAL	4,7325

Latencia Promedio de Eliminación de ECA

FECHA	Gupo de Prueba	PROMEDIO
30/11/2015 0:00	1	2,827
1/12/2015 0:00	2	3,057
	TOTAL	2,9803

Promedio Latencia Todos los Procesos

INDICADOR	PROMEDIO
eca_delete	2,9803
eca_gen	5,6629
metadata_query	3,8167
set_basic_state	3,6087
set_eca_state	4,7325
TOTAL	5,0534



13.4 Prueba 4

Latencia Promedio Lectura de Metadatos de Objetos Inteligentes

FECHA	Grupo de Prueba	PROMEDIO
29/11/2015 0:00	1	4,839
30/11/2015 0:00	2	4,0954
	TOTAL	4,1698

Latencia Promedio Encendiendo el Servicio Inteligente

FECHA	Grupo de Prueba	PROMEDIO
29/11/2015 0:00	1	3,785
30/11/2015 0:00	**2**	**3,8411**
	TOTAL	3,8258

Latencia Promedio Creación ECA

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	6,08
	TOTAL	6,08

Latencia Promedio de Activación de ECA

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	5,0716
	TOTAL	**5,0716**

Latencia Promedio de Eliminación de ECA

FECHA	Gupo de Prueba	PROMEDIO
30/11/2015 0:00	1	7,1095

	TOTAL	7,1095

Promedio Latencia Todos los Procesos

INDICADOR	PROMEDIO
eca_delete	7,1095
eca_gen	6,08
metadata_query	4,1698
set_basic_state	3,8258
set_eca_state	5,0716
TOTAL	4,799

13.5 Prueba 5

Latencia Promedio Lectura de Metadatos de Objetos Inteligentes

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	3,741
1/12/2015 0:00	2	3,9303
	TOTAL	3,8167

Latencia Promedio Encendiendo el Servicio Inteligente

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	3,6062
1/12/2015 0:00	**2**	**3,621**
	TOTAL	3,6087

Latencia Promedio Creación ECA

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	5,5208
1/12/2015 0:00	2	5,7134
	TOTAL	5,6629

Latencia Promedio de Activación de ECA

FECHA	Grupo de Prueba	PROMEDIO
30/11/2015 0:00	1	5,0884
1/12/2015 0:00	**2**	**4,2529**
	TOTAL	4,7325

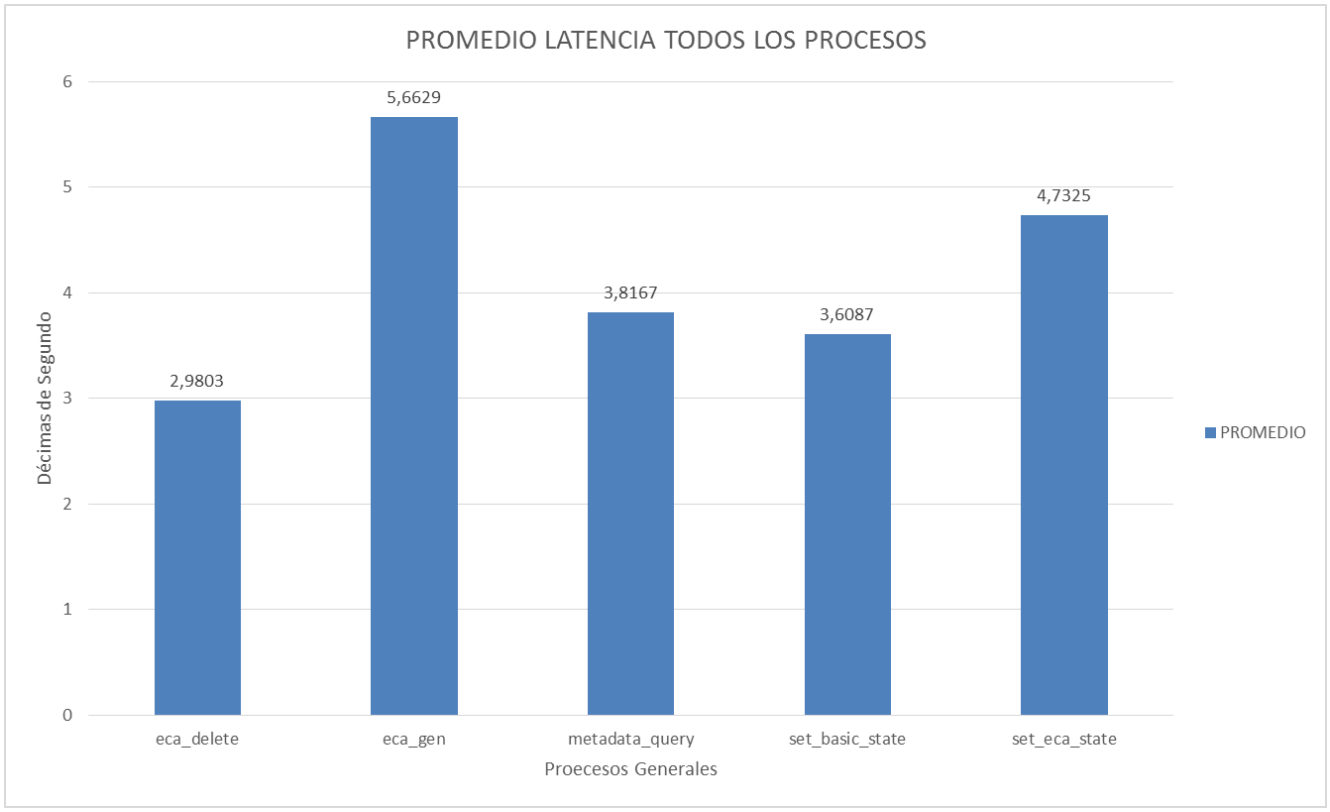
Latencia Promedio de Eliminación de ECA

FECHA	Gupo de Prueba	PROMEDIO
30/11/2015 0:00	1	2,827
1/12/2015 0:00	2	3,057

TOTAL	2,9803

Promedio Latencia Todos los Procesos

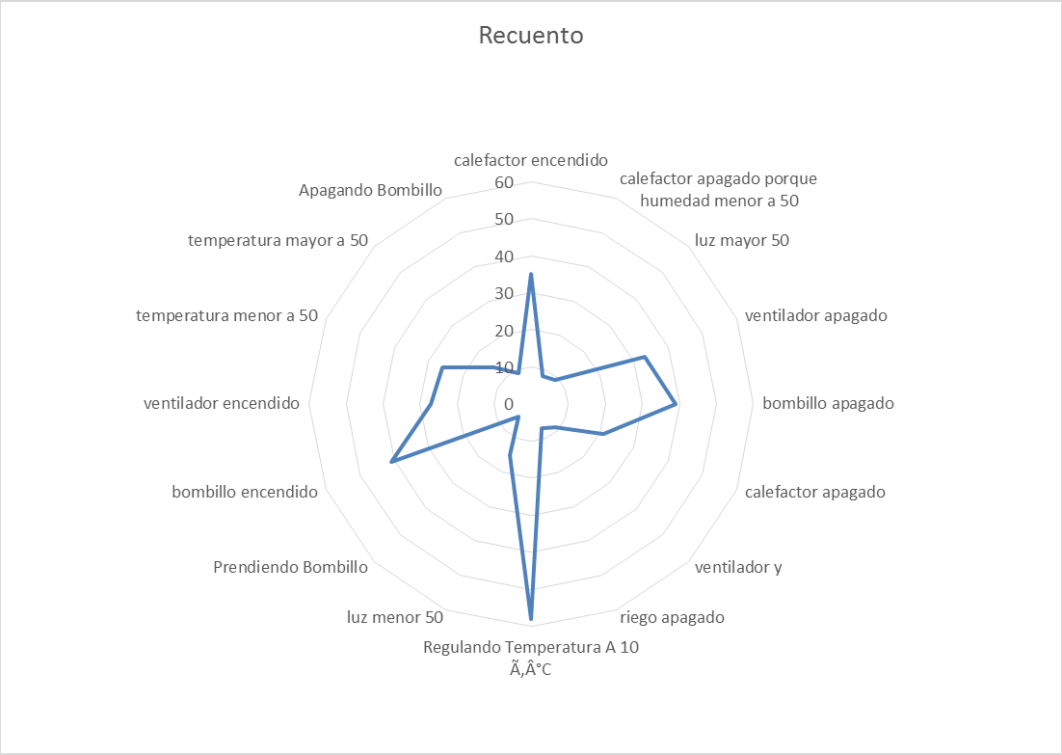
INDICADOR	PROMEDIO
eca_delete	2,9803
eca_gen	5,6629
metadata_query	3,8167
set_basic_state	3,6087
set_eca_state	4,7325
TOTAL	5,0534



13.6 Prueba 6

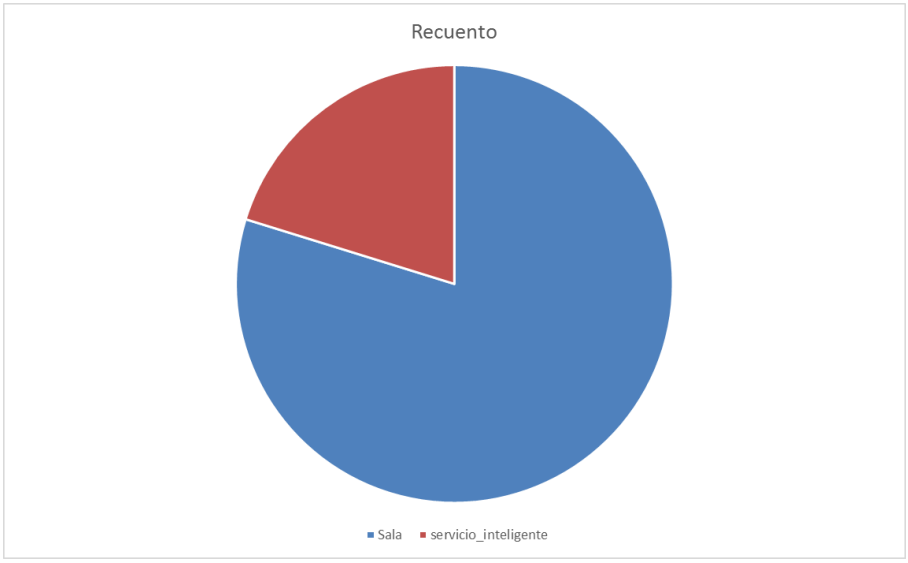
GRUPOS POR CAMBIO DE ESTADO

ESTADO	Recuento
calefactor encendido	35
calefactor apagado porque humedad menor a 50	8
luz mayor 50	9
ventilador apagado	33
bombillo apagado	39
calefactor apagado	21
ventilador y	9
riego apagado	7
Regulando Temperatura A 10 Ã‚Â°C	58
luz menor 50	15
Prendiendo Bombillo	5
bombillo encendido	41
ventilador encendido	27
temperatura menor a 50	26
temperatura mayor a 50	14
Apagando Bombillo	9



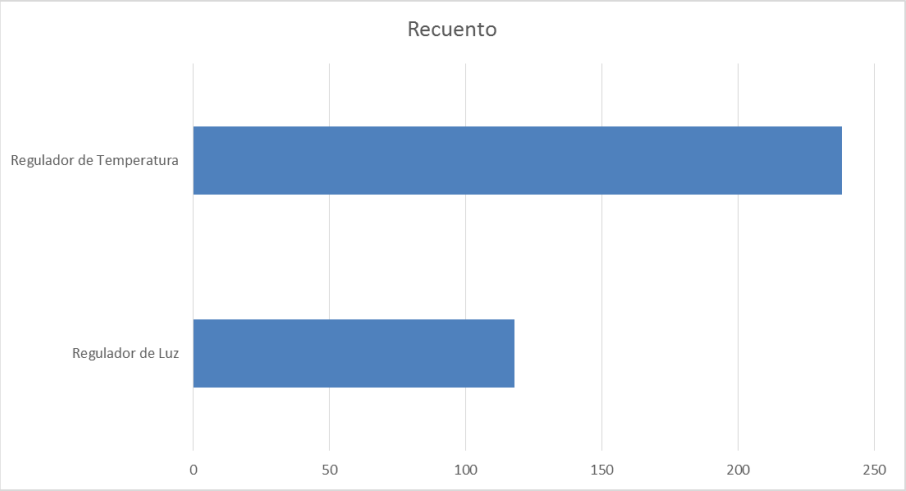
Llamados a las Entidades de Interes

ENTIDAD_INTERES	Recuento
Sala	284
servicio_inteligente	72



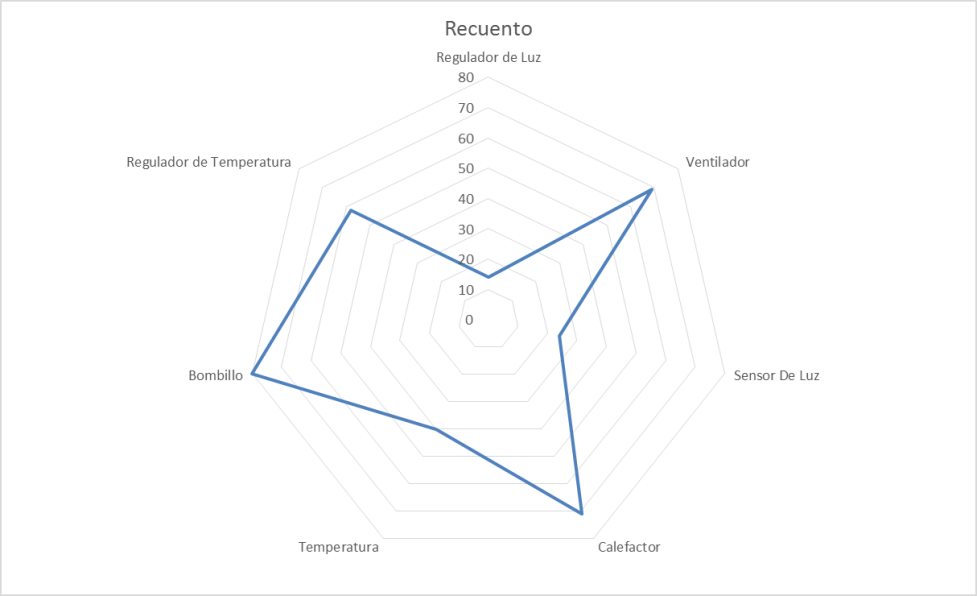
Llamados al Objeto Inteligente

OBJETO	Recuento
Regulador de Luz	118
Regulador de Temperatura	238



Llamados a los recursos Inteligentes

RECURSO	Recuento
Regulador de Luz	14
Ventilador	69
Sensor De Luz	24
Calefactor	71
Temperatura	40
Bombillo	80
Regulador de Temperatura	58


Recuento
Regulador de Luz
80
70
60
50
40
30
20
10
0
Ventilador
Sensor De Luz
Calefactor
Temperatura
Bombillo
Regulador de Temperatura

"Hablando con las Cosas" Un mundo mediado por el Internet de las Cosas, vacíos y retos de los próximos 10 años

Por:
MIGUEL ÁNGEL NIÑO ZAMBRANO
PhD(c) en Ingeniería Telemática
Universidad del Cauca
manzamb@unicauca.edu.co
Tutor: PhD. Gustavo Adolfo Ramírez González

AGENDA

- Escenario de Motivación
- Historia y Conceptos de la IoT
- Tecnologías Hardware y Software de la IoT
- Creación de Productos y Servicios en la IoT
- Demostraciones de Creación de Objetos de la IoT
- Tendencias en Investigaciones en la IoT
- Preguntas

¿Cuál es el futuro a 10 años de la IoT?

Escenario de Motivación

Miguel Ángel Niño Zambrano

Escenario de Motivación – Un día en la vida de Alam Brito (2025)





Sensores & RFID: Falla

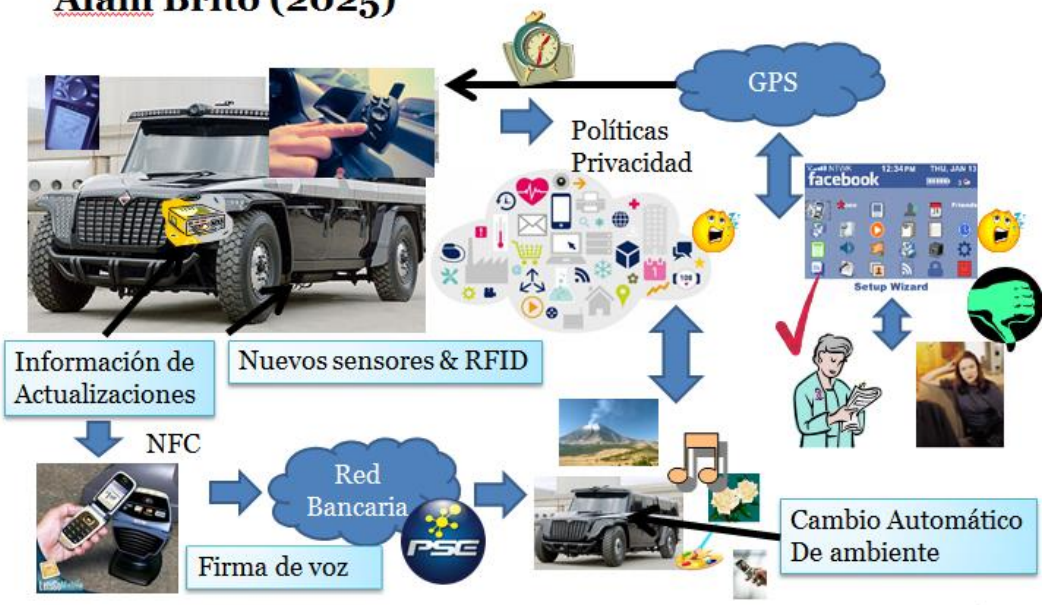
Google Trips: Mapa, rutas, estado, noticias

Radio Inteligente: Itunes Lista: "despecho"

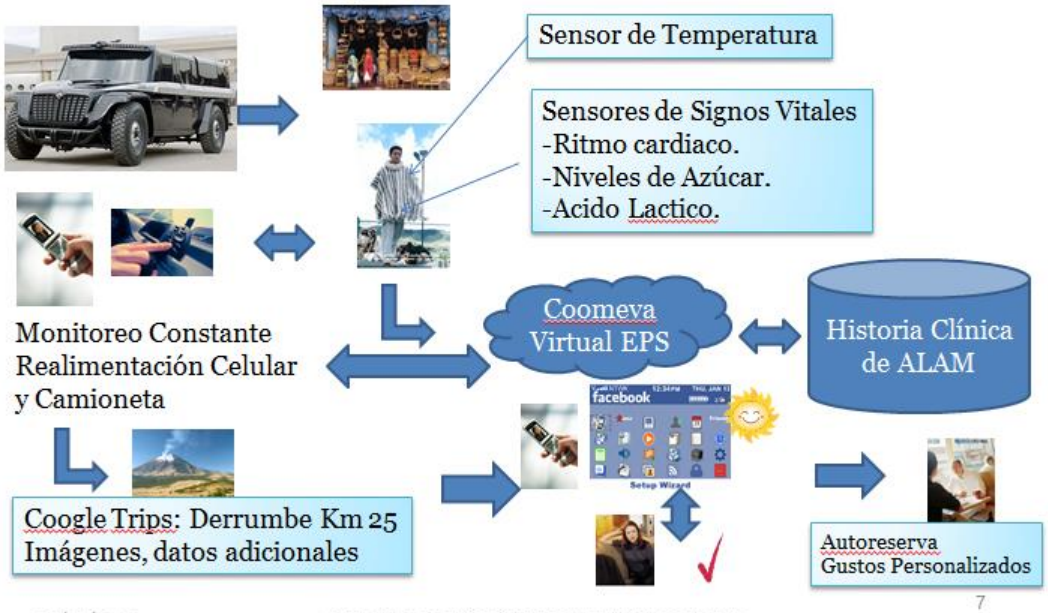
KIZ967: Proveedor, Entrega, Arreglos, Fallo RFID ¿Garantía?

Yenra Smart Truck modelo 2025

Escenario de Motivación – Un día en la vida de Alam Brito (2025)



Escenario de Motivación – Un día en la vida de Alam Brito (2025)



Reflexiones del Escenario de Motivación

- **¿Qué tecnologías, servicios identificamos?**
  - Sensores RFID (todo tipo de objetos)
  - Sensores & Nanotecnología.
  - Productos Inteligentes.
  - Sensores y productos con capacidad de procesamiento
  - Redes Sociales de Objetos.
  - Aplicaciones Web 2.0 interactuando con objetos
  - Convergencia de Servicios de Telecomunicaciones y la Información.
  - Objetos interconectados y compartiendo información.
  - Reglamentación legal, privacidad y protección.
  - Servicios y productos personalizados al cliente

¿Cómo se Desarrolló la IoT?

Historia y Conceptos

Historia – Antes del término IoT

- **(1832):** Un ***telégrafo electromagnético*** fue creado por el ***barón Schilling*** en Rusia, y en 1833 *Carl Friedrich Gauss y Wilhelm Weber* inventó su propio código para comunicarse a través de una distancia de 1200 m en Göttingen, Alemania.

- **(1926)** : ***Nikola Tesla*** en una entrevista con la revista Colliers:

"Cuando inalámbrico se aplique perfectamente toda la tierra se convertirá en un enorme cerebro, ... Un hombre será capaz de llevar a uno en el bolsillo del chaleco ".

Historia – Antes del término IoT

- **(1950):** ***Alan Turing*** en su artículo *"Computing Machinery and Intelligence"* in the Oxford Mind Journal.

"... lo mejor es proporcionar a la máquina con los mejores órganos de los sentidos que el dinero pueda comprar, y luego enseñar a entender y hablar Inglés. Este proceso podría seguir la enseñanza normal de un niño."

- **(1990): John Romkey** creó el primer "dispositivo" de Internet, una tostadora que podría ser encendido y apagado a través de Internet.

Credit: Living Internet

Historia – Antes del IoT

(1991) Marck Weiser: "El incremento de la *disponibilidad* del poder de procesamiento vendrá acompañada por el decremento de la *visibilidad*"

- **Ubiquitous computing (ubicomp)**: La integración de la Informática en el entorno de la persona, así no es visible las tecnologías que lo hacen posible y esta embebida en todo lo que nos rodea.

Historia - IoT

- **(1999)**Primero en acuñar el término de Internet de Objetos – "***Internet of Things***" fue Kevin Ashton, RFID Journal, 22 Junio de 1999. Director de:

Credit: RFID Journal

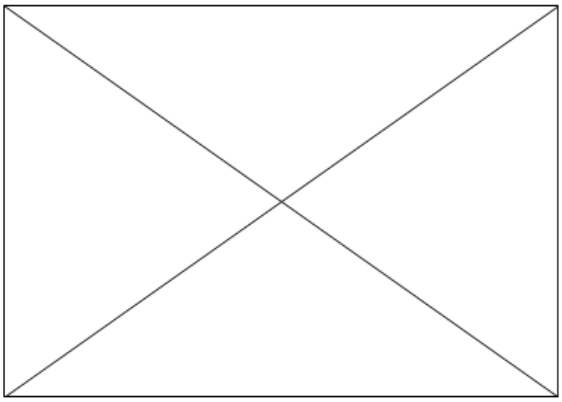
- **Auto-ID Labs**: Es un grupo de investigación en el campo de redes usando "radio – frecuence identification" – RFID, como la pionera de las tecnologías de redes de sensores.

AUTO-ID LABS

- **EPCglobal**: Organización que implementó y desarrolló la estandarización del "electronic product code" – EPC tecnology.
- **(2004): GERSHENFELD, N., KRIKORIAN, R., & COHEN:** Atributos de la IO (Internet Zero).

Historia – Internet o

- **(2005):** Adelantado a su tiempo, el ***Nabaztag*** (ahora parte de Aldebaran Robotics) fue fabricado originalmente por la compañía de Violet y creado por Rafi Haladjian y Olivier Mével . La declaración fue "***si usted puede incluso conectar conejos, a continuación, se puede conectar cualquier cosa***"



Historia – ITU plantea IoT formalmente

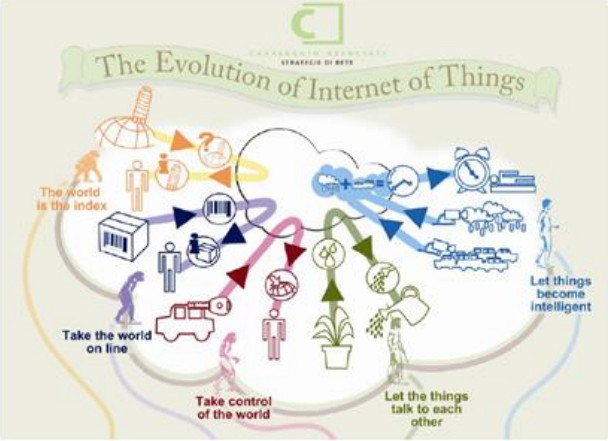
(2005) International Telecommunication Union (ITU):

- **Tecnologías de la IoT:**
  - RFID
  - Tecnologías Sensor
  - Inteligencia Embebida

- **Retos en la IoT:**
  - Crear estándares abiertos para los objetos como EPC Global (RFID) o ZigBee Alliance.
  - Protección de datos y privacidad.
  - Creación de un nuevo ecosistema para los humanos.

Concepto de Internet de Objetos

- (ITU – 2005) La IoT es una red de ***objetos físicos o virtuales*** con acceso a Internet, junto con ***servicios*** online que interactúan con dichos objetos y sus eventos en el espacio y tiempo.
- Los ***objetos*** del Internet de Objetos tienen ***identidad, atributos físicos y personalidades*** virtuales y usan interfaces inteligentes, y están así mismo integradas dentro de la información de la red (Vermesan, Friess et al., 2008).



Características de los Objetos Inteligentes de la IoT

1. Un objeto tiene ***identidad electrónica*** y puede ser consultado de manera remota.
2. Es capaz de ***detectar cambios*** físicos a su alrededor.
3. Es capaz de almacenar ***datos sobre si mismo***.
4. Pueden trabajar conjuntamente para ***crear nuevos servicios*** con otros.

Evolución de los Objetos Inteligentes

Miguel Ángel Niño Zambrano

¿Qué maneja la IoT?

Tecnologías Hardware y Software de la IoT

Miguel Ángel Niño Zambrano

Tecnologías de Comunicación

http://postscapes.com/internet-of-things-technologies

Tecnologías para el manejo de objetos en la IoT

- **Tecnologías clave: IPv4 y IPv6, UDP , TCP, 6LoWPAN**
- **Capacidades SO**: procesamiento, autonomía energética, almacenamiento, comunicación, localización, origen, estado, utilización, id único.
- **Cooperación de Objetos**: Dominio particular, SOA
- **Tipos de Objetos**: físico y virtual.
- **Tipos de Interoperabilidad**: Técnica (red), Sintáctica(EMML,WADL, IDL,WSDL) y Semántica (RDFS, OWL, RF, HTTP, XMPP).
- **Servicios Dinámicos**: Bajo acoplamiento, dependencias de ejecución, reutilización y recambio de servicios
- **IoT vs. WoT**: IoT (Red de objetos, protocolos, arquitecturas), WoT(Web de Objetos, Objetos Semánticos, servicios web, representación digital de lo físico.

Hardware de la IoT

- Sensores y Dispositivos

RFID

RFI Textil

HP Tarjeta Inteligente

RFID FEDex

RFID Implantes

RFID Pets

RFID Tráfico

Arduino
Netduino
Raspberry Pi,
BeagleBone,…

Software

- **Middleware:** Software que provee una capa que abstrae la infraestructura del IoT y las aplicaciones .
- **Servidores IoT (Brokers):** Servidores web que permiten conectar cualquier tipo de dispositivos mediante acceso web y una base centralizada. Ofreciendo diferentes servicios.
- **Sistemas Operativos: Ej.** RIOT OS. Contiki
- **Protocolos de Comunicación:** COAP, MQTT, XMPP.
- **Arquitecturas:** REST, SOA, WSN.
- **Tecnologías de la Web Semántica y la Recuperación de Información:** Ontologías, Sistemas de Búsqueda, PLN, Realidad Aumentada.

Miguel Ángel Niño Zambrano

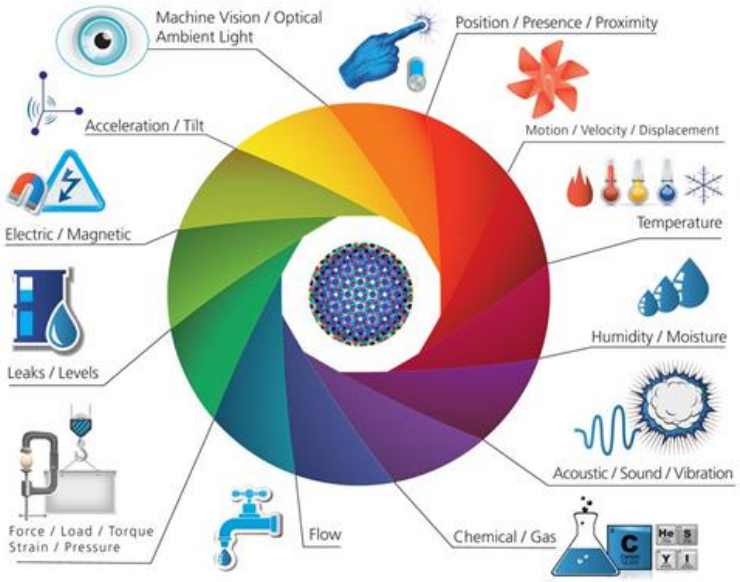
¿Cómo crear la IoT?

Creación de Productos y Servicios en la IoT

Miguel Ángel Niño Zambrano

Aplicaciones de la IoT (Sensores-Conectividad-Personas & Procesos)

1. Dotar los objetos con sensores y Actuadores

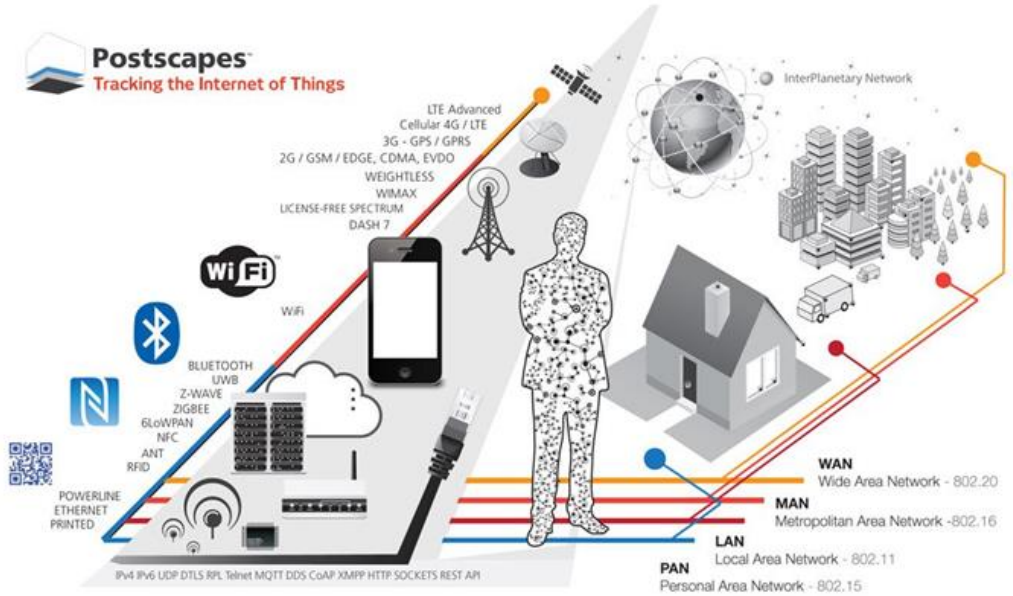


Miguel Ángel Niño Zambrano

http://postscapes.com/internet-of-things-technologies

Aplicaciones de la IoT (Sensores-Conectividad-Personas & Procesos)

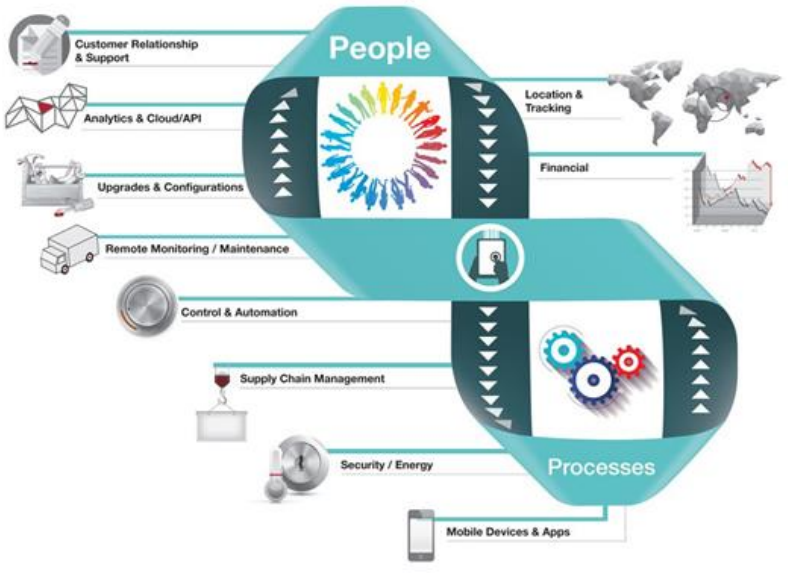
2. Conectar los Dispositivos a Internet



Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

Aplicaciones de la IoT (Sensores-Conectividad-Personas & Procesos)

3. Combinar Datos, Personas & Procesos



Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

Aplicaciones y Servicios Inteligentes de la IoT

Starting with popular connected devices already on the market

SMART THERMOSTATS

nest

Save resources and money on your heating bills by adapting to your usage patterns and turning the temperature down when you're away from home.

CONNECTED CARS

CAR 2GO

Tracked and rented using a smartphone. Car2Go also handles billing, parking and insurance automatically.

ACTIVITY TRACKERS

BASIS

Continuously capture heart rate patterns, activity levels, calorie expenditure and skin temperature on your wrist 24/7.

SMART OUTLETS

belkin

Remotely turn any device or appliance on or off. Track a device's energy usage and receive personalized notifications from your smartphone.

PARKING SENSORS

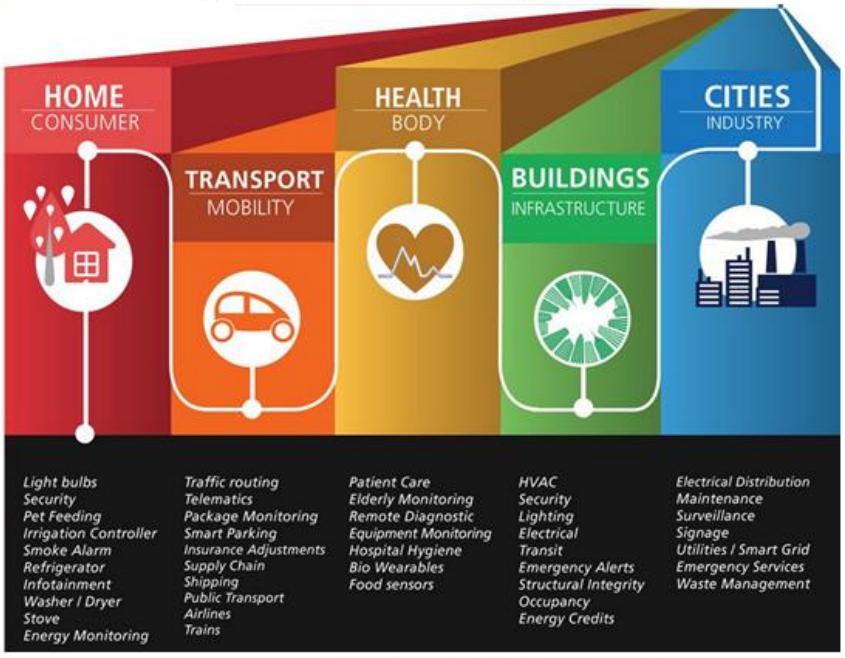
STREETLINE

Using embedded street sensors, users can identify real-time availability of parking spaces on their phone. City officials can manage and price their resources based on actual use.

Whereables: Ejemplo Manilla – SmatPhone y Smartwacth

Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

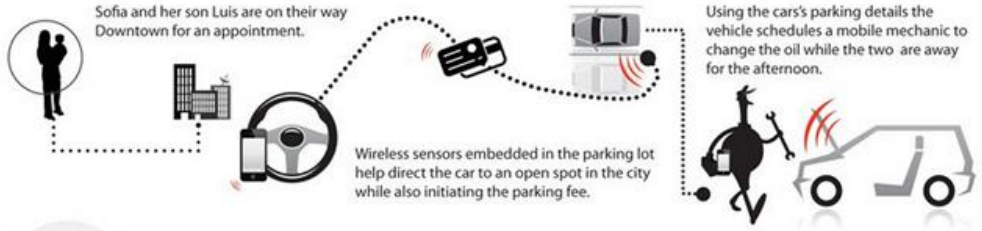
Aplicaciones y Servicios Inteligentes de la IoT



Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

Aplicaciones Complejas con interrelaciones verticales y horizontales en empresas del IoT.

TRANSPORTATION + SMART CITIES



In Downtown San Francisco 20-30% of all traffic congestion is caused by people hunting for a parking spot.

- San Francisco Municipal Transportation Agency (SFMTA)

Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

Aplicaciones Complejas con interrelaciones verticales y horizontales en empresas del IoT.

HEALTHCARE + SMART HOME

40 million adults age 65 and over will be living alone in the U.S, Canada and Europe.

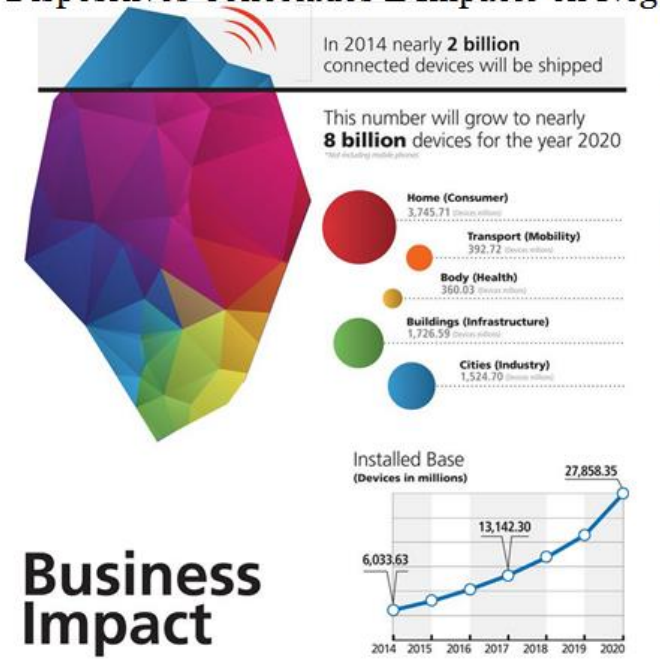
- U.S. Department of Health and Human Services: Administration for Community Living (ACL)

Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

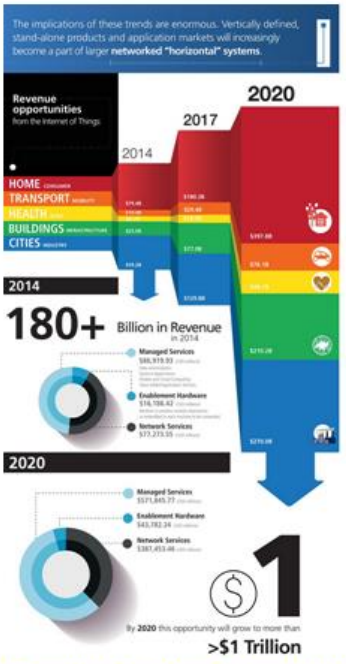
Aplicaciones Complejas con interrelaciones verticales y horizontales en empresas del IoT.

Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

Dispositivos Conectados E Impacto en Negocios



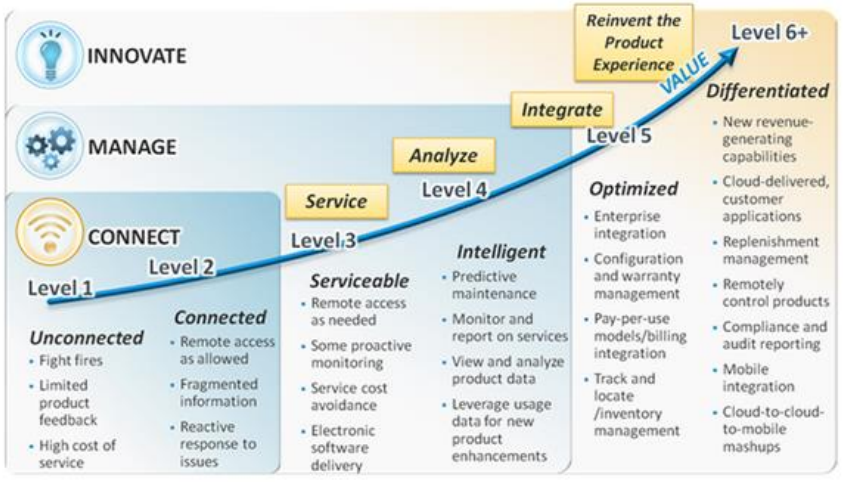
Business Impact



Miguel Ángel Niño Zambrano http://postscapes.com/internet-of-things-technologies

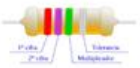
Ciclo de Vida y Desarrollo IoT

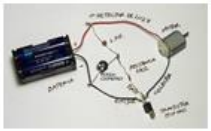
ected Product Maturity Curve as a guide during the workshop (see below), participants from across your organization will share ideas and understand how to achieve greater innovation through connec



Implementando la IoT – Conceptos Básicos

- Conceptos Básicos de electrónica
  - Resistencias
  - Divisores de Voltaje
  - Sensores y Potenciómetros
  - Reguladores de Voltaje
  - Entradas Digitales y Análogas
  - Motores y Transistores





Implementando la IoT – Creando un control de temperatura.

1. Leyendo Sensores
   - Leyendo Interruptores
   - Leyendo la temperatura
   - Leyendo la Luminosidad.
2. Controlando Actuadores
   - Encender Led.
   - Mover un motor.
   - Crear sonido.
3. Interacción Objetos
   - Controlador de Temperatura.
   - Visualizar Datos Procesing.
4. Subir a la Web
   - Subir la variación a Xively

Tendencias de la Tecnología e Investigaciones Futuras

	Before 2010	2010-2015	2015-2020	Beyond 2020
Vision society People	• Socially acceptable RFID	• Pervasive RFID	• Interacting objects	• Personalised objects
	• Realising benefits (food safety, anti counterfeiting, health care) • Consumer concerns (privacy) • Changing ways to work	• Changing business (processes, models, ways to work) • Smart appliances • Ubiquitous readers • Access rights • New retail and Logistics	• Integrated appliances • Smart transportation • Energy & Resource conservation	• Mastered ambient intelligence • Interaction of physical and virtual worlds • Search the physical world (google of things) • Virtual Worlds
Politics & Governance	• De-facto governance • Privacy legislation • Address cultural barriers • Future Internet governance	• EU governance • Frequency spectrum Governance • Sustainable Energy Consumption guidelines	• Authentication, trust and verification • Security, social well-being	• Authentication, trust and verification • Security, social well-being
Standards	• RFID security and Privacy • Radio frequency use	• Sector specific standards	• Interaction Standards	• Behavioural Standards

	Before 2010	2010-2015	2015-2020	Beyond 2020
Vision technology	• Connecting objects	• Networked objects	• Executable objects /semi-intelligent objects	• Intelligent objects
Use	• RFID adoption in logistics, retail and pharmaceutics.	• Increased interoperability	• Decentralised code execution • Global applications	• Unified network that connects people, things and services • Integrated industries
Devices	• Smaller and cheaper tags, sensors and active systems	• Increasing memory and sensing capacities	• Ultra high speed	• Cheaper materials • New physical effects
Energy	• Low power chipsets • Reduced energy consumption	• Improved energy management • Better batteries	• Renewable energy • Multiple sources	• Elements of energy harvesting

Internet of Thing in 2020, A Roadmap for the future, Gérald Santucci and Sebastian Lange

Miguel Ángel Niño Zambrano

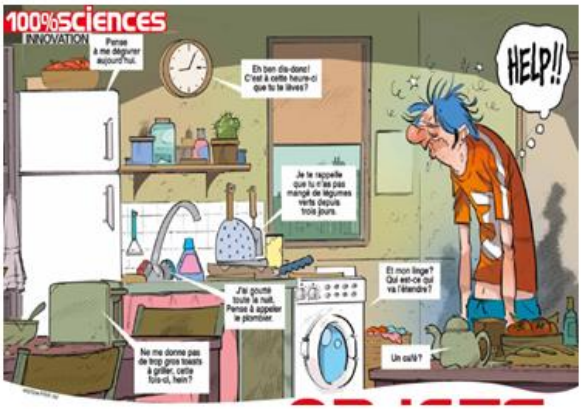
Tópicos que requieren investigación nueva

	Before 2010	2010-2015	2015-2020	Beyond 2020
Vision society	• Wide take up of RFID	• Integration of objects	• Internet of Things	• Unlocked full potential of the Internet of Things
People	• Socially acceptable RFID	• Ambient assisted living • Biometric IDs • Industrial ecosystems	• Smart living • In-vivo health • Security based living	• Mastered continuum of people, computers and things • Automated healthcare
Polities	• First global guidance • Standardisation	• First global governance • Unified open interoperability	• Authentication, trust and verification	• Inclusive Internet of Things
Standards	• Network security • Ad-hoc sensor networks • Protocols for distributed control and processing	• Interoperability protocols and frequencies • Power and fault resilient protocols	• Intelligent devices cooperation	• Health security

	Before 2010	2010-2015	2015-2020	Beyond 2020
Vision technology	• Low power and low cost	• Ubiquitous integration of tags and sensor networks	• Code in tags and objects	• Smart objects everywhere
Use	• Interoperability framework (protocols and frequencies)	• Distributed control and databases • Ad-hoc hybrid networks • Harsh environments	• Global applications • Self-adaptive systems • Distributed memory and processing	• Heterogeneous systems
Devices	• Smart multi-band antennas • Smaller and cheaper tags • Higher frequency tags • Miniaturised and embedded readers	• Extended range of tags and readers and higher frequencies • Transmission speed • On-chip antennas • Integration with other materials	• Executable tags • Intelligent tags • Autonomous tags • Collaborative tags • New materials	• Biodegradable devices • Nano-power processing units
Energy	• Low power chip sets • Thin batteries • Power optimised systems (energy management)	• Energy harvesting (energy conversion, photovoltaic) • Printed batteries • Ultra low power chip sets	• Energy harvesting (biology, chemistry, induction) • Power generation in harsh environments • Energy recycling	• Biodegradable batteries • Wireless power

Internet of Thing in 2020, A Radmap for the future, Gérald Santucci and Sebastian Lange

Problemas y Desafíos Actuales de la Interacción en la IoT

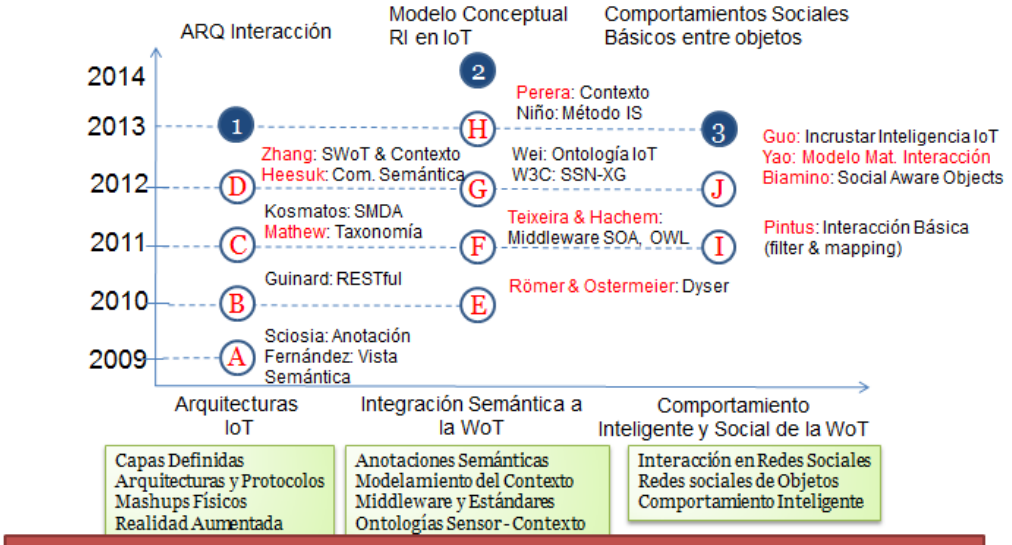


Problemas
- Heterogeneidad HW & SW
- Capacidades de las "Cosas"
- Conectividad: Interacción inteligente.

Desafíos
- Niveles adecuados de abstracción
- Servicios útiles para las personas

Los objetos deberían interactuar entre ellos de manera inteligente

Pregunta y Estado del Arte



¿Cuáles son los elementos, estructuras y mecanismos que permiten modelar la interacción semántica entre los objetos de la WoT?.

Objetivo General

Definir las características, funcionalidades y restricciones que se deben tener en cuenta para realizar **interacción semántica entre objetos inteligentes** en la Web de las Cosas, con el fin de proveer **servicios semánticos de información** a los usuarios de dichos objetos.

Son objetos que poseen capacidades de: Procesamiento, almacenamiento y comunicación (Smart Thing) o implementa estrategias que permiten emular éstas características (Ej. Smart Gateways).

Son servicios web que permiten interacción entre aplicaciones de manera inteligente. Para este proyecto inicialmente se enfoca a proveer servicios de información de la IoT.

Resultados

1. Propone una **modelo para la interacción semántica:** Arquitectura, Vistas Semánticas, Modelo Ontológico IoT, indexación semántica y servicios web semánticos (https://sites.google.com/site/websemanticaiot/home)
2. Define un **método de indexación semántica en la WoT**. (http://semanticsearchiot.net/sswot/simiot/Publish/index.htm).
3. Define un **Índice Semántico en Contaminación Medioambiental** (http://semanticsearchiot.net/sswot/WSSemanticSearch/WSSemanticSearch.asmx)
4. **Ontología de Contaminación Medioambiental**: (http://semanticsearchiot.net/Ontologies/OntologiaContaminacionAmbiental.xml.).
5. **Buscador Semántico para la IoT**: Caso de estudio Contaminación medioambiental. (http://semanticsearchiot.net/sswot/SearchWoT/).

Muchas Gracias

Contacto:
Miguel Ángel Niño Zambrano
Programa de Doctorado en Ingeniería Telemática de la Universidad del Cauca
Afiliación: Profesor Titular Universidad del Cauca, Facultad de Ingeniería Electrónica y Telecomunicaciones – Grupo de Investigación en Ingeniería Telemática - GIT y Grupo de Investigación en Tecnologías de la Información –GTI
Correo: manzamb@unicauca.edu.co, manzamb@hotmail.com
Web del Proyecto: https://sites.google.com/site/websemanticaiot/

¿Preguntas?