UNIVERSIDAD DEL CEMA
Buenos Aires
Argentina

Serie
DOCUMENTOS DE TRABAJO

Área: Ingeniería Informática

ESTILO ARQUITECTÓNICO PARA APLICACIONES IOT

Gabriel M. Barrera

Noviembre 2018
Nro. 664

www.cema.edu.ar/publicaciones/doc_trabajo.html
UCEMA: Av. Córdoba 374, C1054AAP Buenos Aires, Argentina
ISSN 1668-4575 (impreso), ISSN 1668-4583 (en línea)
Editor: Jorge M. Streb; asistente editorial: Valeria Dowding <jae@cema.edu.ar>

Estilo arquitectónico para aplicaciones IoT

Gabriel M. Barrera*

Abstract

Internet de las Cosas (IoT, Internet of Things) se ha posicionado como una nueva tendencia para el desarrollo de diversas aplicaciones interconectando una gran variedad de dispositivos. Los cuales pueden ser de diversa índole o fabricantes, situación que produce que el desarrollo de estas aplicaciones sea complejo de abordar. Conllevando a que en cada emprendimiento se deban encarar diversas soluciones generando un ecosistema muy variado de arquitecturas a evaluar. Para ello se analizan varios estilos de arquitectura del software relevantes, y se proponen cuáles serían los factores que se habrían de tomar en cuenta al momento de afrontar un desarrollo que utilice IoT.

1. Introducción

Se debe considerar que IoT representa la nueva evolución de Internet, siendo un enorme salto en la capacidad para reunir, analizar y distribuir datos que se convierten en información, conocimiento y en última instancia, sabiduría. Es en este contexto, que IoT se vuelve inmensamente importante.

Partiendo de una definición básica, *Internet of Things* se refiere a la variedad de dispositivos para la detección de información, tales como dispositivos de identificación de radiofrecuencia (RFID), sensores infrarrojos, sistemas de posicionamiento global (GPS), scanners láser y otros dispositivos. Los cuales forman una red enorme, donde el objetivo es tener todos los elementos conectados con la red para facilitar la identificación y gestión de estos.

Entonces, Internet de las cosas o IoT se puede definir como la interconexión entre las personas, los animales o los objetos que tengan la capacidad de intercambiar datos a

* Los puntos de vista del autor no necesariamente representan la posición de la Universidad del CEMA.

través de la red sin la participación de humano a humano o la interacción humano-ordenador. IoT ofrece varios tipos de conectividad, desde dispositivos, sistemas y servicios que trabajan dentro de las comunicaciones máquina-máquina (M2M) y que se cubren con aplicaciones, dominios y protocolos [1]

En la actualidad Internet de las cosas (IoT por su acrónico inglés: Internet of Things) está cambiando todo, incluso como enfrentar el desarrollo de aplicaciones que involucren dispositivos. Por lo cual en las etapas de desarrollo de aplicaciones orientadas a la manipulación de datos se ven fuertemente impactadas por este avance. Siendo la arquitectura de los sistemas la primera barrera a sortear para lograr un software de calidad.

La arquitectura de software investiga métodos para determinar cómo dividir mejor un sistema, cómo los componentes se identifican y se comunican entre sí, cómo se comunica la información, cómo los elementos de un sistema pueden evolucionar de forma independiente y cómo se puede describir todo lo anterior utilizando notaciones formales e informales.

La WWW (World Wide Web) ha tenido éxito en gran parte porque su arquitectura de software se ha diseñado para satisfacer las necesidades de un sistema de hipermedia distribuido a gran escala. La Web se ha desarrollado iterativamente durante años a través de una serie de modificaciones a los estándares que definen su arquitectura.

Dentro de la arquitectura del software, existen muchos estilos arquitectónicos que abordan la problemática de interoperatividad y comunicación entre diferentes componentes. Dentro de esas opciones, el presente trabajo se basa en los estilos existentes para realizar un análisis y posible recomendación sobre cuáles utilizar y la viabilidad de combinación entre ellos.

El presente trabajo se estructura de la siguiente manera, en la sección 2 se presenta una descripción de IoT y su estructura. En la sección 3 se presentan las características de IoT y Cloud of Things. En la sección 4 hay un análisis de

estilos arquitectónicos conocidos para tomar como base. En la sección 5 se presenta una descripción de qué debe contemplar al momento de realizar la arquitectura. Sobre los atributos de calidad relevantes para este tipo de sistema se presentan en la sección 5.1. Y finalmente en la sección 6 se plantea la conclusión y el trabajo a futuro a partir del presente trabajo.

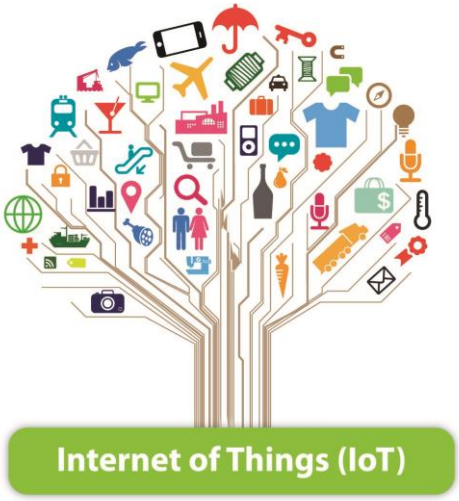

Figura 1. Esquema ilustrativo de los diversos componentes IoT

2. IoT

El núcleo formado por los conceptos de Web 2.0 e Internet of Things (IoT) [2] está cambiando los conceptos de aplicaciones y servicios web hacia una integración y accesibilidad más amplia, a la luz de un paradigma de internetworking en cualquier momento, en cualquier lugar. El futuro de Internet se basa en estos ladrillos para formar una entidad dinámica, produciendo nuevos medios de interacción con los servicios, otros usuarios y el medio ambiente. Las Redes de Sensores Inalámbricos (WSN, Wireless Sensor Networks) han sido reconocidas como un bloque muy importante de este concepto de internetworking.

Existen diversos factores que se deben considerar al momento de realizar desarrollos que impliquen elementos de IoT: la conectividad, los componentes físicos, la arquitectura de conexión, y seguridad entre otros.

2.1. Conectividad

La comunicación "máquina a máquina -M2M-" por cable, inalámbrica ó móvil es una tecnología disponible hace ya varios años. La forma más popular de comunicación en Internet de las Cosas (IoT) actualmente es utilizando tecnologías inalámbricas.

La confiabilidad y privacidad de la información en la red es una de las cuestiones importantes que se están debatiendo. La confiabilidad implica autenticidad de la información en tránsito, de la identificación inequívoca de los actores en la comunicación, como también un reaseguro de la calidad de las mediciones efectuadas por los equipos desplegados.

El manejo de los dispositivos y las comunicaciones debe ser coordinado de una manera tal, que permita la utilización eficiente de la información recolectada, siendo precisas para ello la colaboración y cooperación de las personas, aplicaciones, procesos y servicios a integrarse en el dominio IoT. Las regulaciones en los diferentes escenarios es una tarea complicada toda vez que involucra a las leyes y preceptos locales, gubernamentales e internacionales. La auto-regulación es una propiedad pretendida en los dispositivos de la nueva generación por ser rentable y eficiente; pero en realidad la mayoría de los mismos no soportará esta característica.

La IoT requiere una gran escalabilidad en el espacio de red para manejar la gran y creciente cantidad de dispositivos. IETF 6LoWPAN [1] se utilizaría para conectar dispositivos a redes IP. Con billones de dispositivos [3] siendo agregados al espacio de Internet, IPv6 pasará a tener un rol primordial en el manejo para la escalabilidad en la capa de red. El protocolo de la IETF Constrained

1 IPv6 over Low-Power Wireless Personal Area Networks, RFC 4919

Application, ZeroMQ y MQTT (fig.2) proporcionarán mecanismos ligeros para el transporte de datos.

MQTT (Message Queuing Telemetry Transport) es un protocolo de mensajería basado en la publicación y suscripción estándar ISO. La cual funciona sobre el protocolo TCP/IP. Está diseñado para conexiones con ubicaciones remotas donde se requiere un paquete de datos pequeño o bien el ancho de banda de la red es limitado. En figura 2 se muestra la estructura de los paquetes MQTT. Donde se incluyen todos los datos necesarios para realizar la comunicación, por ejemplo, se incluye el nombre de usuario y contraseña.

Aunque la infraestructura Web se encuentra disponible para dispositivos de IoT y estos la pueden usar, es demasiado pesada para la mayoría de las aplicaciones de IoT. En julio de 2013, IETF lanzó el Protocolo de aplicación restringida (CoAP, Constrained Application Protocol o Protocolo de Aplicación Restringido) para usarlo con nodos y redes de baja potencia y con pérdida [4]. El CoAP, como HTTP, es un protocolo que implementa el estilo de arquitectura RESTful.

El CoAP aborda completamente las necesidades de un protocolo extremadamente ligero y con la naturaleza de una conexión permanente. Tiene conocimiento semántico de HTTP y es RESTful (recursos, identificadores de recursos y manipula esos recursos a través de una Interfaz de Programación de Aplicaciones [API] uniforme).

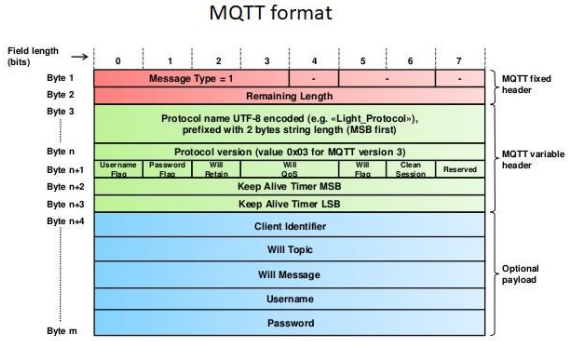

Figura 2. Encabezado de los paquetes en MQTT

2.2. Componentes

Existe una gran variedad de dispositivos. Existen algunos que tienen algún tipo de interfaz de usuario (UI), pero en general los dispositivos IoT están centrados en ofrecer sensores, actuadores o una combinación de ambos. El requerimiento principal del sistema es que se pueda recolectar datos e información desde una multitud de dispositivos y poder almacenarla, analizarla y actuar sobre esta información.

Internet de las cosas ofrece características que hace posible que las personas y cosas puedan comunicarse a través de Internet. Comunican y alcanzan Internet para el almacenaje de datos y las aplicaciones usando diversa infraestructura WIFI o alámbrica.

En el presente ecosistema de IoT, varios componentes de IoT pueden categorizarse ampliamente en tres clases: nodos receptores, nodos de puerta de enlace (gateways) y servicios IoT. Los nodos receptores típicos consisten en electrodomésticos o sensores que observan el entorno físico, que poseen pocos recursos computacionales, restricciones de energía estrictas y recursos limitados de comunicación. El nodo de puerta de enlace funciona como un agregador de datos de sensores y proporciona conectividad con otros nodos de receptores y proveedores de servicios. Los nodos de la puerta de enlace tienen más recursos informáticos en comparación con los nodos del receptor y ocasionalmente proporcionan reemplazo para los nodos del receptor. Los servicios de IoT recopilan datos de los distintos nodos de puerta de enlace y proporcionan servicios específicos de usuario o evento mediante una interfaz gráfica, una notificación o una aplicación.

2.3.Arquitectura del hardware

Una arquitectura típica de una solución IoT consiste en dispositivos restringidos, gateway (pasarelas) o enrutadores fronterizos (routers) y la plataforma en la

nube. En una perspectiva de arquitectura de alto nivel, existen dos tipos de dispositivos: dispositivos restringidos y dispositivos de tipo gateway.

Los dispositivos de tipo pasarela utilizan potentes procesadores, memorias extensibles y sin restricciones en la fuente de alimentación. Pueden enrutar los datos a los servidores de la nube o agregar/almacenar datos para hacer frente a las latencias de la red. Típicamente funcionan con el sistema operativo de Linux con las aplicaciones y disposición para la administración remota.

Los dispositivos restringidos son nodos finales con sensores/actuadores que pueden manejar un propósito de aplicación específico. Por lo general, están conectados a dispositivos de tipo pasarela, red de baja potencia con pérdidas y a su vez, se comunican con las plataformas de IoT. Normalmente se comunican a través de protocolos inalámbricos de baja potencia como BLE, 802.15.4 (6LoWPAN, Zigbee, Hilo, WirelessHART, etc.) o LPWAN, la mayoría con uso de baterías y con baja velocidad de datos. [5]

2.4. Seguridad

La administración de la seguridad es un obstáculo con los que podría chocar el crecimiento de IoT. En primera instancia se siguen los principios de seguridad utilizados en la informática empresarial, lo cual puede ayudar a eliminar ese obstáculo. Con lo cual en conjunto, las siguiente medidas aumentarían significativamente la seguridad para Internet de las cosas:

- Mecanismos de seguridad de extremo a extremo
- Los datos cifrados de extremo a extremo
- Acceso y control de autorizaciones
- Actividad de auditoría
- Infraestructura de nube endurecido

Igual protección a través de múltiples protocolos

3. Arquitecturas IoT y CoT

Con el aumento de la repercusión de Internet de las Cosas aparece el concepto de Computación en la Nube (CoT, Cloud of Things), que consiste en llevar a un servidor externo, el cual se podría identificar como la nube, archivos con cualquier tipo de información hasta la capacidad de procesamiento eliminando las barreras que aporta el hardware local. Según los niveles de seguridad, protección de datos y manipulación hay tres variantes de nubes: pública, privada e híbrida.

En el momento de construir sistemas CoT (Cloud of Things) se tienen ciertas expectativas y condiciones que se desean alcanzar. Y también, ciertos retos y detalles técnicos que se deben controlar. En esta sección se desea explicar estos detalles encontrados por la industria y la academia.

Primero, se espera que los Sistemas CoT (esto incluye los IoT) no solo perciban el entorno en el que se despliegan, sino que puedan tomar decisiones inteligentes y estén conscientes sobre el dominio. Es decir, sean adaptables al cambio y mantengan un ciclo de control cerrado del sistema [6].

Además, deben ser capaces de manejar la heterogeneidad tecnológica, que pueden ser, pero no se limitan a: actualizaciones, sistemas legados sin soporte, nuevas funcionalidades, formatos, protocolos, etc. Y deben ser administrados adecuadamente [6].

Para lograr esto, se recomienda utilizar módulos que abstraigan y faciliten los aspectos técnicos más complicados que deban ser implementados. Se recomienda un administrador de dispositivos (para controlar los sensores activos), de ontologías (para manejar la configuración de sensores y su propósito), de comunicaciones o red (para conectar los componentes) y un administrador de contexto o dominio (que entienda el contexto de trabajo y maneja las entidades, reglas y eventos).

Integrando totalmente los componentes reales y virtuales del sistema se crean espacios ciber-fisicos (CPS), los cuales son representaciones virtuales adecuadas del contexto de negocio y se utilizan para tomar decisiones que mejoren las

propiedades del sistema (tiempo de respuesta, eficiencia, productividad, etc.) [6].

Por otro lado, para crear un software IoT o CoT se requiere tener un grupo de personas con diversos perfiles profesiones, que trabajen juntos y tengan un alto nivel de conocimiento en sus respectivos campos [7].

Para lograr un sistema adecuado se debe tener en cuenta la información de los sensores, los algoritmos utilizados para procesar los datos, el dominio donde se utilizarán los algoritmos (información del negocio) y la infraestructura tecnológica que será utilizada para desplegar el sistema (servidores, IPs, Máquinas Virtuales, VPNs, etc.) [8]. Estos cuatro aspectos son expuestos en la Imagen 4 y permiten proponer servicios que abstraigan los componentes concretos del sistema (ej.: termómetro, barómetro, acelerómetro, GPS, etc.) a funcionalidades de bajo y medio nivel para procesar datos (ej.: calcular la velocidad, posición de un vehículo, detectar colisión, detectar exceso de velocidad), las cuales serán utilizadas por funcionalidades de alto nivel para tomar una decisión (ej.: detectar excesos de velocidad, predecir probabilidad de accidentes, cálculo de riesgo para un seguro de vida)[8].

Por otro lado, ya que existe una gran oferta de tecnología en el mercado (sensores, actuadores, periféricos, etc.). El sistema debe tener cierto nivel de independencia tecnológica. Lo cual se logra implementando prácticas aceptadas (reglas ECA, patrones MVC, Ontologías, APIs) y protocolos estándares (RESTFull, SOAP, html, JSON, MQTT, etc.) lo que evita llegar a puntos muertos [6].

Para crear servicios adecuados se debe pasar por tres etapas. Primero se debe abstraer y estandarizar las fuentes y estructuras de datos utilizadas. Segundo se deben crear los componentes básicos de soporte, de comunicación y lógicos que manejen las entidades del negocio (Ej.: un CRUD). Y por último se debe configurar los comportamientos e interacciones del sistema basado en el conocimiento del dominio que se posee [9].

4. Estilos Arquitectónicos

La arquitectura de software es considerada un puente entre la fase de diseño y la ingeniería de requerimientos [10] dado que tiene una relación directa entre decisiones de arquitectura y los requerimientos [11]. Se define a un estilo arquitectónico como un conjunto de principios que proveen un framework abstracto para una familia de sistemas que promueve la reutilización de componentes y diseños mediante soluciones a problemas recurrentes. Es decir, un estilo arquitectónico es un conjunto de decisiones y principios que encajan como solución en determinados tipos de problemas que suelen repetirse en varios sistemas. [12]

La arquitectura de software de un sistema informático es una descripción del sistema que ayuda a comprender cómo se comportará el sistema. Sirve como un plan para el sistema y el proyecto que lo desarrolla. La arquitectura es el portador principal de las cualidades del sistema, como el rendimiento, la modificabilidad y la seguridad entre otros. Es un artefacto para el análisis asegurarse de que un enfoque de diseño produzca un sistema aceptable. Al construir una arquitectura efectiva, se puede identificar los riesgos del diseño y mitigarlos al principio del proceso de desarrollo[13].

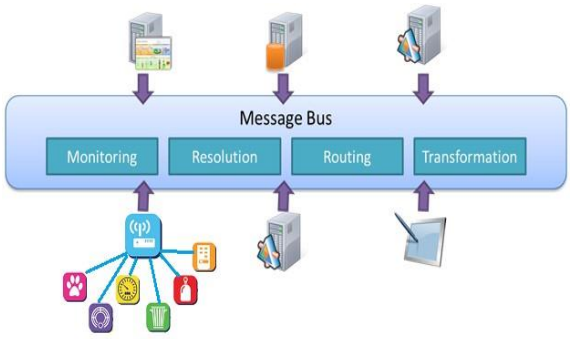

Figura 3. Enterprise Service Bus

El caso de las aplicaciones orientadas a IoT, deben ser diseñadas tomando en consideración una arquitectura idónea para que este pueda cumplir satisfactoriamente sus objetivos y además alcanzar los requerimientos actuales de

modularización, expansión y adaptación a nuevos requerimientos e incrementos en la demanda de uso.

El IoT es una infraestructura siempre cambiante y en constante evolución. Dado que la movilidad forma parte de muchas ”Cosas”, es imposible concebir que los objetos estacionarios basados en la topología de red de hoy en día y las conexiones dedicadas sean funcionalmente sólidas y permitan el desarrollo de IoT. Lo que es necesario la capacidad de mover objetos desde un punto de comunicación a otro y guiarlos según su ubicación o estado mientras permanece conectado al dominio adecuado. En las plataformas estándar XMPP, el enrutamiento de objetos es la configuración de facto, que requiere que todas las aplicaciones y puntos finales se conozcan entre sí y cada filtrado a través de mensajes de presencia para determinar qué elementos son realmente significativos. Entonces la selección y el diseño de la arquitectura es crucial para que el proyecto tenga el éxito deseado.

Uno de los primeros estilos arquitectónicos al necesitar interoperabilidad, es el Enterprise Service Bus (ESB), representado su estructura en la figura 3. Pensado para poder realizar conexiones desde y hasta diversos componentes. El ESB permite a las empresas ser más ágiles en la entrega de nuevos productos y servicios digitales, tanto internamente como a través de ecosistemas digitales. Esta agilidad está respaldada por la capacidad del ESB de integrar sin problemas aplicaciones, servicios, datos y procesos en los sistemas en las instalaciones, la nube, los dispositivos móviles y el IoT. Ayuda a hacer que los datos empresariales sean accesibles mediante la integración de servicios en la nube y software legacy (heredado), y almacenes de datos, así como la transformación de datos sin problemas a través de diferentes formatos y transportes con ESB y las capacidades de servicios de datos. [14]

Existen diversos motivos por las que tomar el camino de aplicar el estilo arquitectonico Enterprise Service Bus sea una buena práctica: Se puede conectar diversos dispositivos y se puede tener un único punto de entrada que pueda enrutar la solicitud al componente especifico. Es posible que se requiera alguna transformación de datos de los

sensores a otros formatos para que puedan ser conectado a otros dispositivos. Implementar alguna lógica de autenticación para que solo el cliente autenticado pueda acceder a los datos. Implementar cierta lógica empresarial antes de devolver los datos a los actuadores conectados. No exponer los dispositivos directamente en Internet. Usando un ESB, se puede mover parte de lógica de los dispositivos IoT a la ESB, dejando a los dispositivos solo las tareas ”electrónicas”. Por ejemplo, se podría usar sensores para monitorear la temperatura interior, pero no agregar lógica de autenticación u otros controles.

Un desafío importante será desarrollar una infraestructura que pueda soportar la gran cantidad de conexiones concurrentes requeridas por el crecimiento exponencial en dispositivos y aplicaciones IoT, que pueden incluir computadoras, teléfonos inteligentes, aviones, GPS, termostatos, controladores de aire, marcapasos, etc.

Al utilizar XMPP y WebSockets, se lograría proporcionar un mecanismo de comunicación poderoso y seguro que incluye el request/response estándar y la mensajería de una sola dirección, pero también incluye presencia. La presencia es una forma única y eficiente de entregar datos y el estado de cualquier punto final a las aplicaciones, las personas y otros dispositivos que lo necesitan de forma automática.

En contrapartida, en muchas aplicaciones de IoT se está buscando que los datos sean procesados en tiempo real, forzando que los diseños de los dispositivos y software lleguen a tener que ser altamente optimizados. Es acá donde ESB podría tener su problema por el principio del uso de conectores, los cuales muchas veces son de terceros, siendo además una capa adicional de procesamiento que si la transformación de los datos fuera realizada de manera nativa.

Los mecanismos de IoT actuales solo proporcionan la entrega de mensajes de extremo a extremo y no tienen acceso a los datos semánticos. Organizaciones como IETF, que administra los estándares de CoAP, y XMPP están trabajando en la estandarización de modelos de datos para sensores como pasos hacia la anotación de datos semánticos [15].

En el proceso de resolver el problema de interoperabilidad del modelo de datos en los sistemas que contemplen IoT, estos esfuerzos avanzan en la dirección de crear estructuras alrededor de estos protocolos, donde estos modelos de datos están centrados en el protocolo e incompatibles con otros modelos de datos.

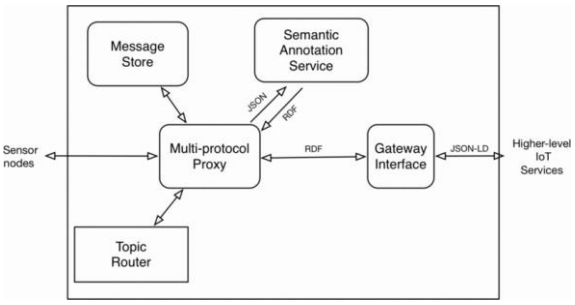

Figura 4. Arquitectura de Semantic Gateway as Service

La anotación semántica de los datos del sensor utilizando un mecanismo y un vocabulario estándar puede proporcionar interoperabilidad entre silos verticales IoT. La comunidad de Web Semántica ha creado y optimizado ontologías estándar para la observación, descripción, descubrimiento y servicios de sensores a través de O&M, SensorML, SOS y SSN. Al integrar estos datos anotados y proporcionar una interfaz de mensajería habilitada para la Web Semántica, un servicio de terceros puede convertir observaciones de sensores heterogéneos en abstracciones de mayor nivel[16].

Debido a que los nodos de la puerta de enlace tienen suficientes recursos computacionales, se puede implementar las tecnologías necesarias para proporcionar interoperabilidad. Del mismo modo, la utilización de tecnologías semánticas en el nivel de servicio también puede permitir la interconexión entre ellos. Aparece el concepto de *Semantic Gateway as Service* (SGS) como un puente entre los nodos receptores y los servicios IoT (Fig. 4)[17]. En la arquitectura de IoT semántica, la puerta de enlace actuá como el centro de comunicación de datos entre el mundo físico y la nube. Esta arquitectura se puede categorizar como Arquitectura Semántica Orientada a

Servicios (SSOA) para sistemas IoT ya que cumple con requisitos técnicos tales como arquitectura orientada a servicios, diseño basado en estándares y agentes de aplicación para aprovechamiento informático basado en semántica con el fin de interpretar de forma autónoma los datos de sensores e interactuar mutuamente [18].

Un micro-servicio es una pequeña aplicación que se puede implementar, escalar y probar de forma independiente y con una sola responsabilidad. La arquitectura de micro-servicios se define como el desarrollo de una aplicación como un conjunto de pequeños servicios independientes, donde cada uno de los servicios se ejecuta en su propio proceso independiente.

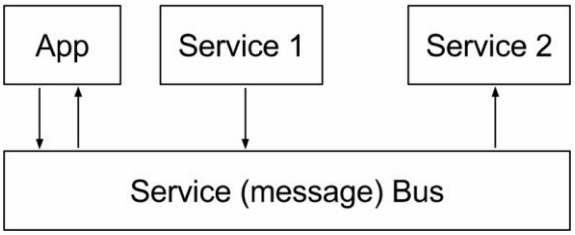

Figura 5. Bus de Servicio/Mensaje

Tres patrones de comunicación de micro servicios se presentan y discuten en [19]. Al considerar la naturaleza asíncrona de las aplicaciones de IoT, los autores sugieren el uso del bus de servicio/mensaje (fig. 5). Se basa en el modelo Publish/Subscribe y permite la adición de nuevos componentes sin cambiar los componentes existentes del sistema.

La Arquitectura dirigida por eventos, Event-driven architecture o EDA, es un patrón de arquitectura software que promueve la producción, detección, consumo de, y reacción a eventos.

Un evento puede ser definido como “un cambio significativo en un estado”. Desde una perspectiva formal, lo que es producido, publicado, propagado, detectado o consumido es un mensaje (típicamente asíncrono) llamado notificación del evento, y no el evento en sí mismo, el cual

es el cambio de estado que disparó la emisión del evento. Por otro lado, el término evento es frecuentemente usado para denotar el mensaje de notificación en sí mismo, lo cual puede llevar a algún tipo de confusión.

Este patrón arquitectónico puede ser aplicado por el diseño e implementación de aplicaciones y sistemas que transmitan eventos entre componentes de software que estén emparejados libremente y servicios. Un sistema dirigido por eventos está compuesto típicamente de emisores de eventos (o agentes) y consumidores de eventos (o receptores). Los consumidores tienen la responsabilidad de llevar a cabo una reacción tan pronto como el evento esté presente. La reacción puede o no puede ser completamente proporcionada por el consumidor en sí mismo. Por ejemplo, el consumidor debe tener solamente la responsabilidad de filtrar, transformar y reenviar el evento a otro componente o debe proporcionar una reacción propia a algún evento.

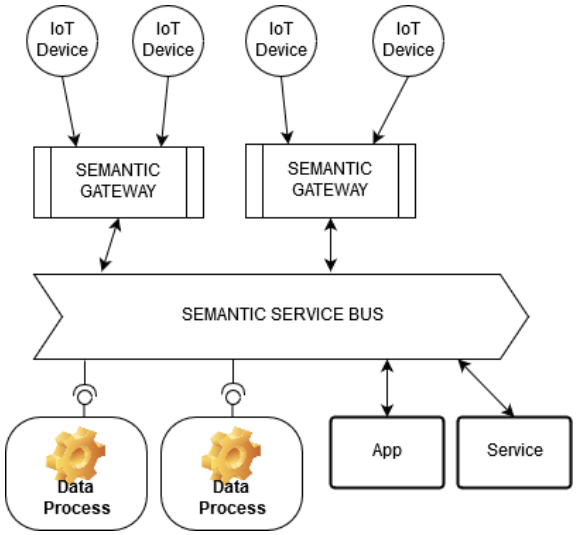

Figura 6. Modelo de arquitectura para IoT

Construir aplicaciones y sistemas alrededor de una arquitectura dirigida por eventos permite a estas aplicaciones y sistemas ser construidos de una manera que facilita un mayor grado de reacción, debido a que los

sistemas dirigidos por eventos están, por el diseño, más normalizados para entornos no predecibles y asíncronos.

La arquitectura dirigida por eventos puede complementar la arquitectura orientada a servicios (SOA) porque los servicios pueden ser activados por disparadores que se encuentran en eventos entrantes. Este paradigma es particularmente útil cuando el consumidor no proporciona algún contenedor ejecutivo propio.

SOA 2.0 engloba las implicaciones de las arquitecturas SOA y EDA proporcionando a un más rico y más robusto nivel, creando un nuevo patrón de eventos. Este nuevo concepto de disparadores de patrones de inteligencia promueve a humanos autónomos o procesamiento automático que añade valor exponencial al negocio. Esto se debe a que se inyecta información de valor añadido en patrón reconocido que no podía haber sido obtenido previamente.

La maquinaría computacional y los sensores (como sensores de cualquier tipo, actuadores, controladores, ...) pueden detectar cambios de estado de objetos o condiciones y crear eventos que pueden ser procesados por un servicio o un sistema. Los disparadores de eventos son condiciones que tienen como resultado la creación de un evento.

5. Estilo Arquitectónico para IoT

Basado en el análisis de la sección anterior, se logra proponer un estilo arquitectónico orientado a aplicaciones IoT basado en las fortalezas de estilos actuales.

Una arquitectura de referencia cumple un papel fundamental en los dispositivos IoT. Su función es vital para la correcta integración de los dispositivos con diferentes sistemas. Aunque sea difícil encontrar una arquitectura de referencia que cumpla con las necesidades de cada uno de los diferentes tipos de dispositivos IoT, sabemos que si se trabaja en una arquitectura escalable y modular se podrá cumplir todos nuestros objetivos.

Este tipo de arquitecturas es la que ayuda a adaptar los requerimientos específicos de los proyectos y también a

evolucionar, añadir servicios y hacer más compleja la arquitectura cuando sea necesario.

La arquitectura propuesta debe ser escalable, lo que ayuda al IoT a desarrollarse y expandirse de manera más fácil y eficiente. Al configurar e implementar los agentes de aplicación correspondientes, se pueden desarrollar nuevas aplicaciones para cumplir con los requisitos emergentes e integrarlas en el dominio de aplicación existente de manera eficiente sin considerar las características o cambiar otras aplicaciones existentes. Específicamente, no importa qué tecnologías aplique una nueva aplicación, puede introducirse en el dominio de aplicación implementando un agente de aplicación correspondiente, que debe suscribirse al agente de dominio y establecer las conexiones de comunicación con otros agentes de aplicación si es necesario basándose en un agente unificado protocolos de comunicación.

Además, si un dominio de aplicación es demasiado grande y debe dividirse en múltiples subdominios, o si se debe construir un nuevo dominio en el presente dominio de aplicación según las necesidades, se puede lograr de manera eficiente introduciendo nuevos agentes de dominio y reconstruyendo el dominio. las relaciones entre todos los agentes, incluida la cancelación de suscripción, la suscripción y el reencaminamiento en función del protocolo de comunicación del agente unificado.

5.1. Atributos de Calidad

Dentro de los atributos de calidad deseados en cualquier estilo arquitectónico, para la situación de aplicaciones orientadas a IoT, se debe pensar en interoperabilidad, modificabilidad, escalabilidad y seguridad.

5.1.1. Interoperabilidad

Estos sistemas deben de tener capacidad de intercambiar datos y posibilitar la puesta en común de información y conocimientos. La interoperatividad es la condición mediante la cual sistemas heterogéneos pueden intercambiar procesos o datos. Por eso se podría decir que

es uno de los atributos con mayor relevancia. En los sistemas orientados o que hacen uso de IoT son conjuntos de diversos componentes que deben poder intercambiar información. En el contexto de las aplicaciones IoT existe un gran abanico de opciones, diferentes plataformas, lenguajes de programación, compañias proveedoras de servicios y componentes con sus propios protocolos, eso hace que este atributo sea complejo de conseguir, pero a su vez el más importante de lograr.

5.1.2. Modificabilidad

Otro factor importante para tener en consideración en la modificabilidad, la habilidad del sistema para ser flexible frente a cambios durante el desarrollo y durante su vida en producción. Puesto que los diversos componentes están en constante evolución y actualización, y además, el mismo sistema debe poder soportar los agregados a nuevos dispositivos, sin perder su funcionamiento. Por ejemplo, si la aplicación controla un aire acondicionado marca X, el cual luego es reemplazado por la marca Y, el sistema debería ser los suficientemente versátil para soportar el cambio. Incluso, debería poder soportar el agregado de nuevas funcionalidades, por ejemplo, que el sistema pueda además controlar la iluminación.

5.1.3. Flexibilidad

La flexibilidad es la habilidad del sistema para adaptarse a ambientes y situaciones variables y para soportar cambios en políticas de negocios y reglas de negocio. Un sistema flexible es uno que es fácil de reconfigurar o que se adapta en respuesta a los diferentes requerimientos de usuarios y del sistema. Dentro del contexto de Internet de las cosas, esto puede convertirse vital, para que una aplicación sea fácilmente adaptada cuando el entorno en el que se encuentra es cambiado.

5.1.4. Escalabilidad

La escalabilidad es la habilidad del sistema para funcionar bien cuando se presentan cambios en la demanda o en la carga del mismo. Típicamente el sistema será capaz de extenderse a un número mayor o más poderoso de servidores al incrementarse la demanda o la carga. Si bien no todos los sistemas orientados a IoT deberán contemplar este atributo de calidad, puesto que no todos están pensado para grandes volúmenes de datos, si es a tener en cuenta para aquellas que puedan contemplar un creciente número de dispositivos.

5.1.5. Seguridad

Sin duda alguna, la seguridad es la base de toda tecnología. Realmente no se puede construir una arquitectura totalmente segura. No existen las máquinas seguras, lo único que se puede garantizar es que si se pone un ordenador encerrado en una habitación, blindado con paredes de plomo de un metro de grosor, sin puerta y sin conexión a ninguna red, podemos asegurar que está relativamente seguro. No existe la seguridad informática 100% segura, sin embargo, se busca poner todas las trabas para evitar ataques.

La seguridad en el IoT es algo que se debe tener en cuenta desde el principio. Ya no solo están en peligro los datos, sino también está en peligro la integridad física de las personas y bienes.

6. Conclusión

El trabajo presenta una visión de que debería comprender un estilo arquitectónico para aplicaciones con componentes IoT. Varios patrones y principios de diseño de arquitectura de software se prevén y se discuten en el documento. La arquitectura de software propuesta sigue la regla de dependencia de la arquitectura limpia, la responsabilidad única y los principios de segregación de la interfaz para facilitar la capacidad de ampliación y la capacidad de prueba del sistema.

En este trabajo se presenta que pautas se deberían tener en consideración al momento de crear una arquitectura de software para un sistema orientado a IoT, sin embargo, queda para futuros trabajos el poder realizar un desarrollo utilizando estos principios y analizando las diferentes etapas involucradas y evaluando si los atributos de calidad deseados llegaron a cumplirse y en qué medida.

Referencias

[1] J. Holler, V. Tsiatsis, C. Mulligan, S. Avesand, S. Karnouskos, and D. Boyle, *From Machine-tomachine to the Internet of Things: Introduction to a New Age of Intelligence*. Academic Press, 2014.

[2] E. F. I. Portal. Future internet assembly. [Online]. Available: http://www.future-internet.eu

[3] G. Says, "6.4 billion connected "things" will be in use in 2016, up 30 percent from 2015," *Gart. Inc*, 2015.

[4] Z. Shelby, K. Hartke, and C. Bormann, "The constrained application protocol (coap)," 2014.

[5] F. Khodadadi, A. V. Dastjerdi, and R. Buyya, "Internet of things: An overview," *arXiv preprint arXiv:1703.06409*, 2017.

[6] D. Conzon, P. Brizzi, P. Kasinathan, C. Pastrone, F. Pramudianto, and P. Cultrona, "Industrial application development exploiting iot vision and model driven programming," in *Intelligence in Next Generation Networks (ICIN), 2015 18th International Conference on*. IEEE, 2015, pp. 168–175.

[7] P. Grace, B. Pickering, and M. Surridge, "Modeldriven interoperability: engineering heterogeneous iot systems," *Annals of Telecommunications*, vol. 71, no. 3-4, pp. 141–150, 2016.

[8] A. Pal, A. Mukherjee, and P. Balamuralidhar, "Model-driven development for internet of things: Towards easing the concerns of application developers," in

Internet of Things. User-Centric IoT. Springer, 2015, pp. 339–346.

[9] H. Cai, Y. Gu, A. Vasilakos, B. Xu, and J. Zhou, "Model-driven development patterns for mobile services in cloud of things," *IEEE Transactions on Cloud Computing*, 2016.

[10] I. Sommerville, *Ingeniería del software*. Pearson Educación, 2011.

[11] X. Ferre, N. Juristo, A. Moreno, and I. Sánchez, "A software architectural view of usability patterns," in *2nd Workshop on Software and Usability Cross-Pollination (at INTERACT'03) Zurich (Switzerland)*, 2003.

[12] M. P. . P. Team, *.NET application architecture guide*. Microsoft, 2009.

[13] Defining software architecture. [Online]. Available: https://www.sei.cmu.edu/architecture/

[14] D. Chappell, *Enterprise service bus*. .ºReilly Media, Inc.", 2004.

[15] P. Waher. Xep-0323: Internet of things - sensor data. [Online]. Available: https://xmpp.org/extensions/xep-0323.html

[16] H. Patni, C. Henson, and A. Sheth, "Linked sensor data," in *Collaborative Technologies and Systems (CTS), 2010 International Symposium on*. IEEE, 2010, pp. 362–370.

[17] P. Desai, A. Sheth, and P. Anantharam, "Semantic gateway as a service architecture for iot interoperability," in *Mobile Services (MS), 2015 IEEE International Conference on*. IEEE, 2015, pp. 313–319.

[18] M. Deriaz and G. D. M. Serugendo, "Semantic service oriented architecture," *Switz. Univ. Geneva*, 2004.

[19] D. Namiot and M. Sneps-Sneppe, "On microservices architecture," *International Journal of Open Information Technologies*, vol. 2, no. 9, pp. 24–27, 2014.