Arquitectura para diseñar e implementar Web Services

Architecture to design and implement Web Services

Gabriel Eduardo Duarte Vega.
Ingeniero Mecatrónico

Universidad Nacional de Colombia,
geduartev@unal.edu.co

Artículo deInvestigación

Para citar este artículo: Duarte, G. E. (2015). Arquitectura para diseñar e implementar Web Services, 3(2),

Fecha de recepción: 02-12-2015

Fecha de aceptación: 29-12-2015

Resumen

El diseño de web services no tiene suficiente información bibliográfica que integre una metodología sistemática para generar una arquitectura adecuada que permita tener un web service completo y que se pueda utilizar en cualquier aplicación o investigación en la academia, la organización y la industria. Es por esto que se proponen dos tipos de arquitecturas que permitirán implementar un web service completo, en el cual *completo* se entiende como cumplir una serie de características que permitirá al web service adaptarse en cualquier contexto. Los web services son apoyo fundamental en temas como sistemas adaptativos complejos, realidad aumentada, inteligencia de negocios, inteligencia artifi*cial, videojuegos, automatización, solo por nombrar* algunos. El uso de un web service queda atado a la creatividad de quien decida utilizarlo, so pena de decir que muchos los rechazan o simplemente no los utilizan.

Palabras clave: Servicios web, microservicios, transacciones financieras, sistemas distribuidos, sistemas adaptativos complejos.

Abstract

Web services design has insufficient bibliographical information which integrates a systematic methodology to generate an appropriate architecture that allows to have a full web service and might be used in any application or research in academy, organization or industry. That's why two types of architectures that will implement a full web service is proposed. In it, “full” is defined as fulfill a number of features that allows the web service fit in any context. Web services support with major issues as complex adaptive systems, augmented reality, business intelligence, artificial intelligence, video games, automation, just to name a few. Using a web service remains tied to the creativity of those who decide to use it, on pain of saying that many rejected or simply not used.

Keywords: Web services, microservices, financial transactions, distributed systems, complex adaptive systems.

1. INTRODUCCIÓN

En la actualidad los web services (servicios web) se han convertido en la solución para centralizar operaciones porque son componentes que permiten interactuar entre sí o con otros componentes para realizar operaciones más complejas, de ahí la importancia de la arquitectura del servicio web.

En este artículo se presentarán los dos tipos de arquitectura, una tradicional y otra por microservicios; arquitecturas que se proponen cada una con sus componentes, arquitecturas no propuestas en ninguna de las referencias bibliográficas consultadas. Estas arquitecturas se utilizarán en una de las tres primeras etapas de la investigación y servirán para dar respuesta a las preguntas de investigación, argumentar la hipótesis y solucionar el problema planteado en la investigación.

Actualmente los web services son utilizados en aplicaciones e investigaciones en la academia, la organización y la industria. Son los componentes de software ideales con los que se puede interactuar y permiten a su vez escalabilidad, eficiencia, concurrencia sin costos grandes o tiempos largos de desarrollo.

2. TEMA DE INVESTIGACIÓN

La implementación de web services puede ser de gran interés para la academia y la industria debido a su modularidad, escalabilidad, reutilización, rendimiento, trazabilidad y seguridad. En especial porque cualquier tipo de interfaz (portal web, aplicación móvil, base de datos, máquina e incluso otros web services) puede llegar a tener centralizada las operaciones y compartir información, sin importar el lenguaje o tecnología en el que se hayan creado.

Los web services son importantes en investigaciones, aplicaciones e innovaciones de vanguardia como internet de las cosas, realidad aumentada, realidad virtual, inteligencia de negocios, inteligencia artificial, videojuegos, robótica, portales o aplicaciones web, aplicaciones móviles o transacciones financieras, por nombrar solo algunas.

En la bibliografía consultada se encontraron temas específicos que hablan en forma conjunta o separada acerca de definiciones, conceptos, seguridad, lenguajes de programación, semántica, protocolos de comunicación, etc., pero no se encontró evidencia de una propuesta metodológica, método o modelo sistemático para diseñar e implementar web services.

Se espera entonces aportar como valor a la planeación estratégica del uso de las tecnologías de información y comunicación en la academia, la organización y la industria, una metodología sistemática para diseñar e implementar web services.

Planteamiento del problema

De la experiencia propia y a través de la observación se evidencia que al momento de diseñar web services no se tiene claro qué componentes mínimos deben conformar un servicio, no se tiene una guía sobre cómo utilizar las tecnologías más recientes que se pueden utilizar para desarrollarlo, no se tiene un orden en su implementación, no se realizan pruebas unitarias adecuadas y la documentación para utilizar o consumir el servicio no es clara o diciente. Adicional la bibliografía no trata temas concretos sobre componentes básicos o mínimos que definen un web service funcional.

Todo esto trae como consecuencias errores en la arquitectura y el diseño. Estos errores impiden escalar el servicio, mejorarlo o realizar un mantenimiento de forma fácil y rápida. Lo anterior se traduce en reprocesos, retrasos en las entregas, baja calidad del producto y sobrecostos innecesarios, además del rechazo a utilizar en las innovaciones de desarrollo de software web services.

Por lo tanto, se realiza la propuesta de una metodología sistemática que gradualmente nos introduzca en el diseño, el desarrollo y la implementación de un web service utilizando un diseño y una arquitectura clara de los componentes y el orden en el que se debe llevar a cabo su implementación utilizando las tecnologías de web services más recientes, automatizando las pruebas, generando una documentación entendible y fácil de utilizar y actualizar.

Formulación

¿De qué forma podemos hacer o desarrollar una metodología sistemática para crear web services?

Sistematización

¿Cuál es el mejor diseño o implementación para un web service?

¿Cuál debe ser el proceso de implementación de un web service y qué tecnologías de desarrollo utilizar?

¿Cómo podemos evaluar la calidad o corrección de un web service aplicando la metodología sistemática?

Objetivo general

Diseñar una metodología sistemática mediante el uso de tecnologías recientes, buenas prácticas y pruebas para el diseño e implementación de web services.

Objetivos específicos

Identificar los componentes mínimos que se deben contemplar en un servicio mediante una arquitectura de microservicios para el diseño y arquitectura de web services.

Implementar el web service utilizando tecnologías recientes y de uso libre para codificar, realizar pruebas unitarias y documentar el servicio.

Aplicar la metodología para implementar un web service en un contexto de transacciones bancarias financieras básicas mediante una aplicación web de pruebas para consumirlo.

Justificación teórica y práctica

No se quiere ampliar, refutar o reafirmar un modelo teórico o contrastar la forma de cómo se presenta en una realidad, por lo tanto no se presentarán resultados que sean complemento teórico a ninguna investigación.

Este proyecto de investigación se lleva a cabo porque se quiere demostrar que mediante el uso ordenado y adecuado de métodos, tecnologías y buenas prácticas se puede concebir un web service en muy poco tiempo y de gran calidad.

Se busca que con el diseño, la arquitectura y la implementación de un web service esta investigación se pueda emplear en otras investigaciones posteriores y explicar su validez a través de su aplicación.

El resultado de la presente investigación ayudará en investigaciones o aplicaciones académicas, de organización o de industrias que deseen utilizar o crear web services.

Hipótesis

Si se sigue una metodología para crear un web service entonces se disminuirán errores de diseño, reprocesos y tiempo de implementación.

3. MARCO TEÓRICO Y CONCEPTUAL

¿Qué es un web service?

Un web service (en inglés) o un servicio web (en español) son aplicaciones de software que exponen métodos para consultar, insertar, actualizar o eliminar información.

La información de solicitud y respuesta del web service se realiza con mensajes, mediante Protocolos de comunicación.

Internamente el web service procesa la información e interactúa con otros componentes, por ejemplo, bases de datos, aplicaciones móviles, dispositivos físicos, centros de procesamiento de información, incluso otros web services.

Tecnologías que usan los web services

En los mensajes de solicitud y respuesta que se utilizan para comunicar los web services, se puede utilizar alguna de las siguientes tecnologías:

- *XML (Lenguaje extensible de etiquetas)*

Es un estándar para describir datos y crear etiquetas. Las características especiales son la independencia de datos o de la separación de los contenidos de su presentación. Es un metalenguaje que permite diseñar un lenguaje propio de etiquetas para múltiples clases de documentos.

Los documentos XML se componen de unidades de almacenamiento llamadas entidades (entities), que contienen datos analizados (parsed) o sin analizar (unparsed). Los datos analizados se componen de caracteres, algunos de los cuales forman los datos del documento y el resto forman las etiquetas.

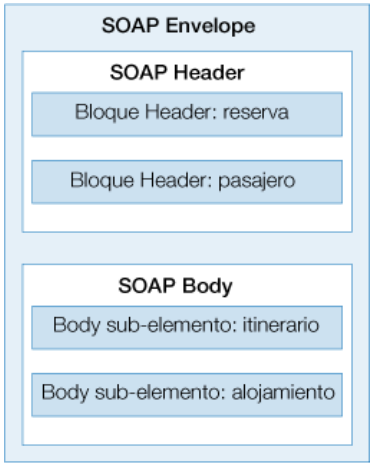
Las etiquetas codifican la descripción de la estructura lógica y de almacenamiento del documento. XML proporciona un mecanismo para imponer restricciones en la estructura lógica y de almacenamiento.

- *SOAP (Protocolo Simple de Acceso a Objetos)*

Se trata de un protocolo basado en XML, que permite la interacción entre varios dispositivos y tiene la capacidad de transmitir información compleja. Los datos pueden ser transmitidos a través de HTTP, SMTP, etc.

Un SOAP especifica el formato de los mensajes. El mensaje SOAP está compuesto header (cabecera) y body (cuerpo). (Ver Figura 1)

Figura 1. Estructura de los mensajes.



Fuente: Guía Breve de Servicios Web W3C, 2105.

- *WSDL (Web Service Definition Language, Lenguaje de descripción de web services)*

Especificación XML para la formación del documento de descripción de un web service. Identifica los métodos, funciones y parámetros necesarios para invocar un determinado servicio. Así, un usuario puede crear una aplicación cliente que comunica con el web service.

- *UDDI (Universal Description, Discovery and Integration - Descripción, Descubrimiento e Integración)*

UDDI son las siglas del catálogo de negocios de Internet denominado Universal Description, Discovery and Integration. El registro en el catálogo se hace en XML. UDDI es una iniciativa industrial abierta (sufragada por la OASIS) entroncada en el contexto de los servicios Web (XML, 2015).

Es un elemento básico sobre el que se asientan los web services hace posible que las empresas puedan encontrar y compartir web services.

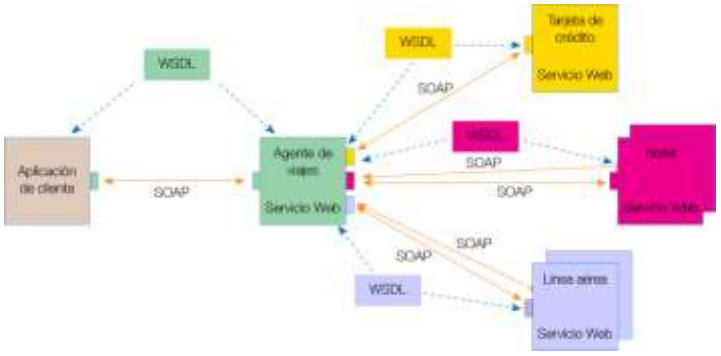
UDDI provee un mecanismo que permite que se "describan" a sí mismos y los tipos de servicios que proporcionan para luego registrar y publicarse en un Registro UDDI. Tales negocios publicados pueden ser buscados, consultados o "descubiertos" por otros negocios utilizando mensajes con SOAP.

¿Para qué sirven?

Los web services proporcionan un componente de comunicación estándar, interactúan entre sí para presentar información a quien la solicita, proporcionan interoperabilidad y extensibilidad entre diferentes aplicaciones, sin importar su tecnología, se pueden combinar para crear operaciones más complejas, centralizan operaciones.

El siguiente ejemplo se toma completo de la página oficial de W3C, ya que explica cómo funciona un web service. (Guía Breve de Servicios Web W3C, 2105)

Figura 2. Ejemplo de varios services en funcionamiento.



Fuente: Guía Breve de Servicios Web W3C, 2105.

En el ejemplo se aprecia uno de los componentes que interactúa con los servicios web. Por ejemplo, las aplicaciones clientes (aplicación móvil, plataforma web, dispositivo físico, etc.).

Accediendo a la URL del servicio se consumirán los métodos expuestos de información de la agencia de viaje (Ver Figura 2).

La agencia de viajes ofrecerá información a sus clientes, por ejemplo, en relación con el hotel y la compañía aérea. Información que a su vez la convierte en cliente de esos otros web services que le van a proporcionar la información solicitada sobre el hotel y la línea aérea. Por último, el usuario realizará el pago del viaje a través de la agencia de viajes que servirá de intermediario entre el usuario y el web service que gestionará el pago.

4. ARQUITECTURAS PROPUESTAS PARA DISEÑAR E IMPLEMENTAR WEB SERVICE

En este artículo se presentan los resultados de una de las tres etapas del proyecto de investigación planteado al inicio, el cual propone una metodología sistemática para crear web services.

El documento IEEE Std 1471-2000, dice que:

> *La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.*

Los resultados de la primera etapa de investigación arrojan dos arquitecturas con diferentes distribuciones de componentes para el diseño e implementación de web services que se presentan en este artículo.

Web service completo

Cuando un web service cumple su función principal: recibir y responder a las solicitudes, además de validar y verificar la información, tiene una persistencia de información y se deja configurar y auditar, se considera como un web service completo.

La siguiente es la lista de características que usamos en este trabajo para considerar un web service completo:

- Permite exponer métodos a través de una URL.
- Recibe solicitudes y devuelve respuestas en un formato definido. Por ejemplo: JSON o XML.
- Se puede administrar la configuración del web service a través de una base de datos relacional o no relacional.
- Valida tipo de datos, obligatoriedad, formatos, longitudes en los parámetros de la solicitud.
- Verifica los datos de los parámetros cuando sea necesario contra una base de datos relacional, no relacional u otro servicio.
- Permite configurar el servicio almacenando la información en una base de datos relacional o no relacional.
- Administra roles y permisos sobre los métodos a consumir.
- Cumple por seguridad el proceso de autenticación y autorización para consumir cualquier método del servicio.
- Permite hacer auditoría mediante la consulta o trazabilidad de los eventos realizados en el servicio.
- Administra las excepciones que se presentan durante la ejecución del servicio.
- Permite administrar y configurar un protocolo de comunicación mediante códigos de respuesta.
- Maneja unos tiempos de espera y respuesta cuando se consumen los métodos antes de lanzar excepciones.
-

Arquitectura general para un web service completo

Teniendo en cuenta las características que debe cumplir un web service completo, se propone, diseña y presenta una arquitectura para un web service.

Figura 3. Arquitectura general para un web service completo.

En la Figura 3 se observa una arquitectura general para un web service completo, la cual consta de los siguientes componentes:

WEB SERVICE: el diagrama con forma de nube representa el conjunto de métodos privados y públicos que se expondrán mediante una URL.

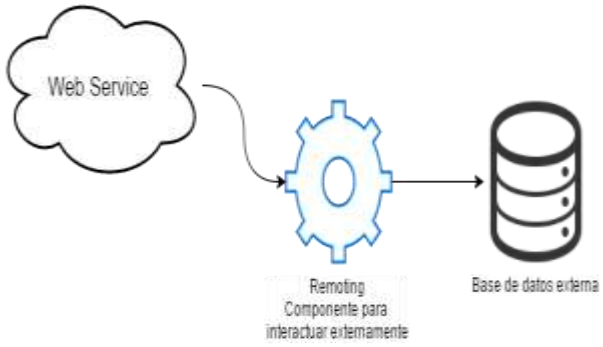
REMOTING: es un componente middleware[1] con el cual interactuarán externamente otros componentes u otros web services.

DB No SQL: esta base de datos no relacional se utiliza para dejar una trazabilidad de cada uno de los eventos que ocurren en el servicio, sirve de auditoría.

DB SQL: esta base de datos relacional se utilizará para administrar las configuraciones propias del servicio y de cada uno de los métodos del servicio.

Internamente el componente con forma de nube o "Web Service" contiene las propiedades y métodos que se utilizarán para realizar solicitudes. Los métodos serán públicos, ocultos[2] y privados. El Web Service se encarga de la lógica de validación y verificación. Para verificar datos es probable que tenga que interactuar en muchas ocasiones con componentes externos, como por ejemplo, una base de datos externa, tal como se puede apreciar en la Figura 4.

Figura 4. Verificación de información del web service a través de Remoting hacia una base de datos externa.



Ejemplo de aplicación de un web service con arquitectura general en el contexto de las transacciones bancarias financieras

El servicio "Web Service" también puede interactuar con otros componentes, centraliza las conexiones directas que pueden realizar los usuarios a través de aplicaciones móviles, aplicaciones web, portales, CMS, puntos de atención en comercios mediante cajeros automáticos o datafonos (POS).

Por ejemplo, en el contexto de las transacciones bancarias financieras, una solicitud realizada por un usuario a través de una aplicación móvil deberá ser validada por "Componente autorizador del banco" (Ver Figura 5).

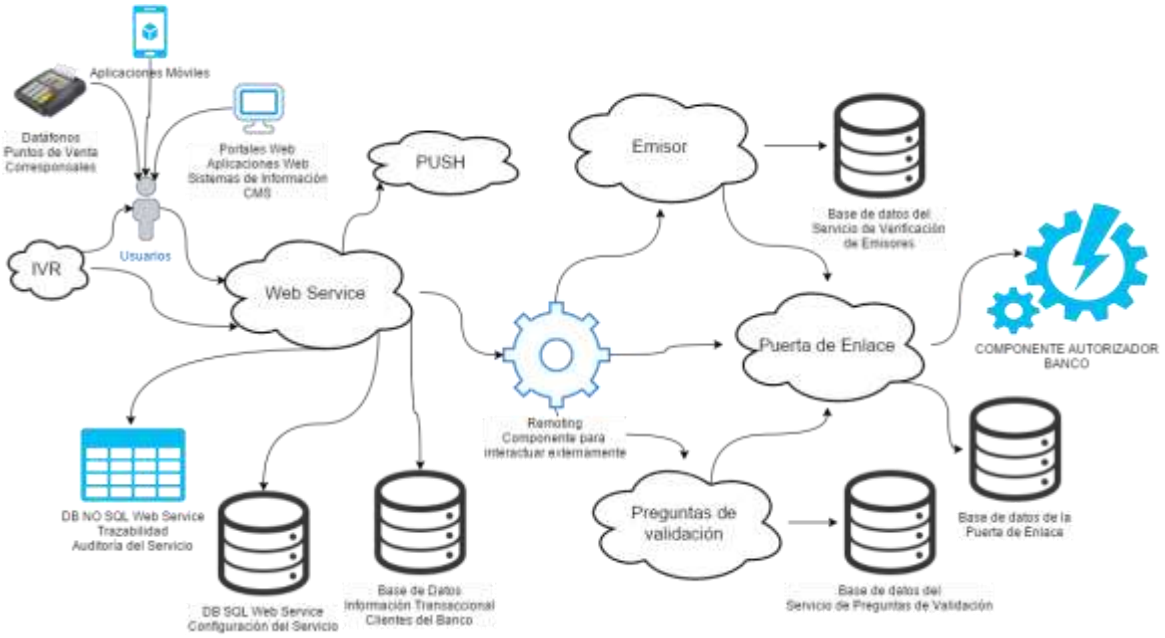
La información del banco al que pertenece el usuario deberá traerse a través del componente "Emisor", que además se encarga de administrar y configurar parámetros independientes para cada transacción en cada emisor o banco configurados (Ver Figura 5).

El servicio "Preguntas de validación" es un web service que se encarga de generar y convalidar las preguntas de validación al consumir cualquier método en el servicio con el propósito de garantizar la seguridad (Ver Figura 5).

Notemos que los componentes anteriores interactúan todos a través de la "Puerta de enlace", que actúa como un Switch para que de acuerdo a quien le haga peticiones (emisor, web service o preguntas de validación) sepa cómo redirigir sus solicitudes hasta los "Componentes autorizadores del Banco" (Ver Figura 5). Los usuarios realizan sus transacciones mediante solicitudes consumiendo los métodos expuestos en el web service (Ver Figura 5).

[1] Middleware es un software para interactuar con otros softwares o aplicaciones simplificando la tarea de realizar varias conexiones o sincronizaciones por cada aplicación con la que se desea interactuar.
[2] Los métodos ocultos se pueden acceder a través de la URL del servicio pero no dan a conocer mediante el WSDL.

Figura 5. Ejemplo de arquitectura general de un web service aplicado en el contexto de transacciones bancarias financieras.



Estas solicitudes las pueden realizar, como se había dicho, mediante datafonos, aplicaciones móviles, portales web, etc.

Los eventos quedan registrados uno a uno en una base de datos no relacional, la cual servirá más adelante para realizar seguimiento o reportes.

Servicios de telefonía automática como los IVR interactúan con el usuario y con el web service de forma directa.

Arquitectura con microservicios

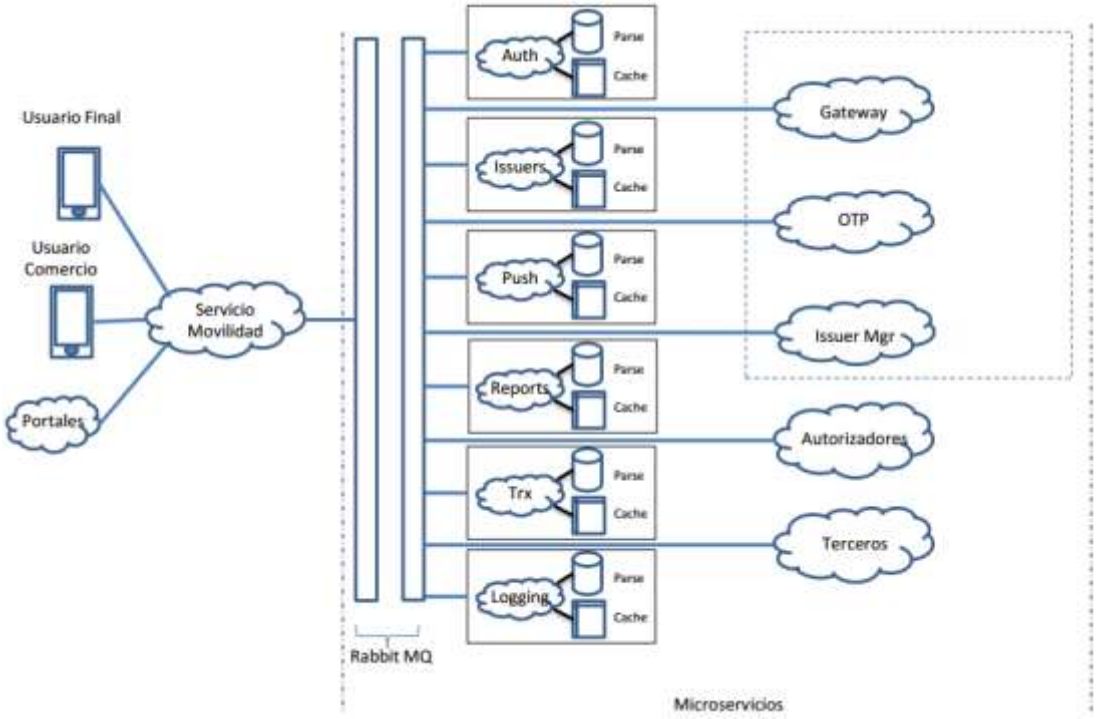
La arquitectura de microservicios no es un concepto nuevo, se ha venido aplicando recientemente en la innovación de aplicaciones.

De acuerdo a sus pioneros Martín Fowler y James Lewis en su artículo "Microservices"[8], se definen como un "estilo arquitectural en el que múltiples servicios, cada uno corriendo de manera individual y desplegados de la forma más automatizada posible, se comunican entre sí mediante mecanismos ligeros, generalmente un recurso API basado en HTTP".

Se decide entonces proponer una segunda arquitectura con microservicios, la cual tendrá otra perspectiva frente a la arquitectura general anterior.

Ejemplo de aplicación de un web service con arquitectura de micro servicios en el contexto de las transacciones bancarias financieras

Figura 6. Ejemplo de arquitectura con microservicios de un web service aplicado en el contexto de transacciones bancarias financieras.



Esta arquitectura tiene grandes ventajas, ya que cada componente se trata como una unidad independiente y única, lo que permite modificaciones y actualizaciones sobre un componente sin afectar otros, en menos tiempo, puesto que no se tiene toda una lógica compleja en un todo.

En la Figura 6 tenemos una arquitectura de microservicios, el middleware es una nueva tecnología conocida como RabbitMQ[8], que se encarga de realizar la comunicación entre el web service "Servicio Movilidad" y los microservicios.

Por ejemplo, cuando un usuario final realiza una solicitud, esta es recibida por el servicio de movilidad que a través de RabbitMQ[8] determina el camino y las conexiones al microservicio correspondiente.

Cada microservicio tiene su correspondiente cache, lo que evita redundar o realizar demasiadas peticiones a la persistencia de datos.

Cada microservicio tiene su propia persistencia de datos en la nube, lo que nos desliga de administración de servidores, motores de bases de datos, licencias, administración y gestión. Además es una base de datos no relacional, lo que nos permite tener ganancias de tiempo en las consultas y sus resultados, mucho más que por el manejo de cache anterior.

La puerta de enlace "Gateway", las preguntas de validación "OTP" y los emisores "Issuer Mgr" siguen cumpliendo el papel mencionado en el ejemplo anterior de la arquitectura general, solo que esta vez interactúan a través de RabbitMQ[8] (Ver Figura 6).

Algo diferente está en que estos tres componentes ya no interactúan con los componentes autorizadores de un banco, ya que el componente autorizador del banco y terceros son individuales y se comunican directamente con RabbitMQ[8]. (Ver Figura 6).

Otros componentes utilizados en las transacciones bancarias o financieras como por ejemplo, el autorizador de inicio de sesión "Auth", el conocimiento de emisores "Issuers", las notificaciones "Push", los reportes o "Reports", la trazabilidad y auditoría de las transacciones "Trx" y los usuarios registrados y autorizados "Logging" se manejarán a través de microservicios, cada uno con

su propia base de datos y caché. Estos ya no se encontrarán, todos, agrupados en una misma base de datos lo que hace más eficiente al servicio web.

Al momento de implementar un web service con una arquitectura de microservicios se tratarán como proyectos independientes, se puede dividir el trabajo, se pueden realizar actualizaciones, modificaciones, correcciones y mantenimiento de forma más rápida, ya que no afectará otros componentes. El seguimiento o depuración de problemas se facilitará.

5. CONCLUSIONES

No se encontró en la bibliografía consultada un conjunto de elementos completos para definir una arquitectura adecuada, por lo que se propone la arquitectura general y se da un ejemplo de cómo utilizar la misma.

Este trabajo es importante porque permite servir de guía para futuras investigaciones y aplicaciones en cualquier rama del conocimiento.

Se espera que este trabajo pueda ser comparado con otras propuestas de arquitecturas.

La modularidad de la arquitectura de microservicios tiene la ventaja de permitir realizar correcciones, mejoras, mantenimientos y actualizaciones de forma más eficiente que la arquitectura general.

La arquitectura de microservicios si la implementa una sola persona tendrá la experiencia de repetir código en cada microservicio, esto no es una desventaja, ya que cada servicio debe observarse como una unidad o componente independiente, pero como es una sola persona quien desarrolla varios microservicios se dará cuenta de tal hecho.

Utilizar bases de datos no relacionales para la trazabilidad, la configuración y almacenar otra información donde se aplique el web service nos da la ventaja en la rapidez con la que se realizan y devuelven las consultas.

La arquitectura de microservicios se puede extender con otras tecnologías de almacenamiento de información en la nube, como por ejemplo RabbitMQ.

6. REFERENCIAS

[1] Bhatia, R.; Singh, M. (2015). Minimizing privacy disclosure in web services paradigm. *Procedia Computer Science, 48*: 781–788.

[2] Casado, R.; Younas, M.; Tuya, J. (2013). Multi-dimensional criteria for testing web services transactions. *Journal of Computer and System Sciences, 79*(7): 1057–1076.

[3] Austin, D.; Barbir, A.; Ferris, C.; Garg, S. (Editores). (2004). Web services architecture requirements. *W3C Working Group Note*. Recuperado de: http://www.w3.org/TR/wsa-reqs/

[4] Solis, D.; Roque, W.; Morilla, M. L. (2013). Pasarela de pagos para la seguridad de transacciones bancarias en línea. *3c Empresa, 15*. Recuperado de: http://dialnet.unirioja.es/descarga/articulo/4817913.pdf

[5] Silva, J. P.; de Miguel, M.; Briones, J.; Alonso, A. (2014). Estrategia guiada por modelos para incluir aspectos de seguridad en sistemas empotrados basados en servicios web. *Revista Iberoamericana de Automática e Informática Industrial RIAI, 11*(1): 86–97.

[6] Consorcio World Wide Web. (s.f.). Guía breve de servicios web. *W3C*. Recuperado de: http://www.w3c.es/divulgacion/guiasbreves/serviciosweb

[7] Jaca, M. C.; Serrano, N. (2010). Aplicaciones de la Web 2.0 en las Pymes como herramienta para la innovación y mejora. *Revista de Ingeniería Dyna, 85*(8): 662–666.

[8] Lewis, J.; Fowler, M. (2014). *Microservices*. Recuperado de: http://martinfowler.com/articles/microservices.html

[9] Alfonso, J.; Lizcano, D.; Martínez, M. A.; Pazos, J. (2013). Developing frontend Web 2.0 technologies to access services, content and things in the future Internet. *Future Generations Computer Systems FGCS, 29*(5): 1184–1195.

[10] Maamar, Z.; Costantino, G.; Petrocchi, M.; Martinelli, F. (2015). Business reputation of social

networks of web services. *Procedia Computer Science, 56*: 18–25.

[11] Scheihing, V. A. (2003). *Diseño e implementación de una solución de business intelligence sobre información de transacciones bancarias electrónicas de la empresa Redbanc*. (Tesis de grado). Facultad de Ciencias de la Ingeniería, Escuela de Ingeniería Civil en Informática, Universidad Austral de Chile. Recuperado de: http://cybertesis.uach.cl/tesis/uach/2003/bmfcis318d/sources/bmfcis318d.pdf

[12] Ramírez, N. D. (2008). El fraude en la actividad bancaria. *El Cuaderno – Escuela de Ciencias Estratégicas, 2*(4): 279–296.

[13] Nacer, H.; Aissani, D. (2014). Semantic web services: Standards, applications, challenges and solutions. *Journal of Network and Computer Applications, 44*: 134–151.

[14] Lassala-Navarré, C.; Ruiz, C.; Sanz, S. (2010). Implicaciones de la satisfacción, confianza y lealtad en el uso de los servicios bancarios online: un análisis aplicado al caso español. *Revista Europea de Dirección y Economía de la Empresa, 19*(1): 27–46.

[15] Niknam, M.; Karshenas, S. (2015). Sustainable design of buildings using semantic BIM and semantic web services. *Procedia Engineering, 118*: 909–917.

[16] Olvera-Lobo, M. D. (2000). Rendimiento de los sistemas de recuperación de información en la web: evaluación de servicios de búsqueda (search engines). *Revista Española de Documentación Científica, 23*(3): 302–316.

[17] Ran, S. (2003). A model for web services discovery with QoS. *ACM SIGecom Exchanges, 4*(1): 1-10.

[18] Rassan, I. A.; Haifa, A. (2015). Tagged-Sub Optimal Code (TSC) compression for improving performance of web services. *Procedia Computer Science, 62*: 167–169.

[19] RabbitMQ. (s.f.). Recuperado de: https://w ww.rabbitmq.com

[20] Ruíz del Olmo, Francisco Javier. (2012). Dispositivos móviles y servicios web. Características sociales y comunicativas de su convergencia. *Icono14. Revista Científica de Comunicación y Tecnologías Emergentes, 8*(1): 220-237.

[21] Samper, J.; Llidó, D.; Soriano, F.; Martínez, J. J. (2015). Semantic web service discovery system for road traffic information services. *Expert Systems with Applications, 42*(8): 3833–3842.

[22] Lara, B. A.; Rodríguez, A. M. (2004). *Guía metodológica para la generación de servicios en línea a partir de los estándares WFS y WMS basados en visualización con tráfico liviano y manejo de seguridad*. (Proyecto de grado para optar al título de Ingeniero de Sistemas). Facultad de Ingeniería, Pontificia Universidad Javeriana. Recuperado de: http://javeriana.edu.co/biblos/tesis/ingenieria/Tesis209.pdf

[23] Talon, A. F.; Madeira, E.R.M. (2015). Comparison between light-weight and heavy-weight monitoring in a web services fuzzy architecture. *Procedia Computer Science, 64*: 862–69.

[24] Valero, V.; Macià, H.; Pardo, J. J.; Cambronero, M. E.; Díaz, G. (2012). Transforming web services choreographies with priorities and time constraints into prioritized-time colored Petri nets. *Science of Computer Programming, 77*(3): 290–313.

[25] Lamarca, M. J. (s.f.). *Hipertexto: el nuevo concepto de documento en la cultura de la imagen*. XML. Recuperado de: http://www.hipertexto.info/documentos/xml.htm

[26] Zou, G.; Gan, Y.; Chen, Y.; Zhang, B. (2014). Dynamic composition of Web services using efficient planners in large-scale service repository. *Knowledge-Based Systems, 62*: 98–112