UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

MODELO DE COMPOSICIÓN DE MICROSERVICIOS PARA LA IMPLEMENTACIÓN DE UNA APLICACIÓN WEB DE COMERCIO ELECTRÓNICO UTILIZANDO KUBERNETES

MICROSERVICE COMPOSITION MODEL FOR THE IMPLEMENTATION OF AN E-COMMERCE WEB APPLICATION USING KUBERNETES

Donia Alizandra Ruelas Acero[1,*]

[1]Universidad Nacional del Altiplano, Escuela Profesional de Ingeniería de Sistemas, Av. Floral N° 1153, Ciudad Universitaria, Puno, Perú, donializ7@gmail.com

RESUMEN

El comercio electrónico ha crecido en los últimos años y a consecuencia de ello ha recibido una mayor atención por de las empresas, las cuáles vienen invirtiendo en la implementación de aplicaciones web; incrementado su demanda y la existencia de usuarios concurrentes en determinados tiempos, ocasionando un mayor nivel de exigencia; que conlleva a problemas como la disponibilidad limitada, tiempo de respuesta excedida, y el rendimiento limitado al acceder a la aplicación web del comercio electrónico. Por otro lado la arquitectura de microservicios es una nueva tendencia que crece rápidamente en el mundo empresarial, sin embargo existe poca literatura de composición de microservicios. El objetivo de esta investigación ha sido proponer un modelo de composición de microservicios para la implementación de una aplicación Web de comercio electrónico utilizando la tecnología Kubernetes, para lo cual se utilizó el modelo y notación el proceso de negocio de comercio electrónico, el diseño arquitectónico de la composición de los microservicios, la implementación de los microservicios en forma independiente la evaluación de éstos se ha realizado a través de atributos de calidad, y la validación del modelo propuesto usando pruebas de carga. Como resultado se obtuvo que la aplicación web funciona con una mejora significativa en un 104 %, en los indicadores de rendimiento, disponibilidad y tiempo de respuesta, en comparación con una aplicación web basado en el modelo monolítico. Por lo que el modelo de composición de microservicios presentado tiene un funcionamiento significativo.

Palabras clave: Aplicación web, comercio electrónico, disponibilidad, modelo de composición de microservicios, rendimiento, tiempo de respuesta.

ABSTRACT

Electronic commerce has emerged more in recent years and as a result it has received more interest from companies, and they are investing on web application implementations; as it has been increased demand and concurrent users at certain times, resulting in a higher level of demand; which leads to issues such as limited availability, response times exceeded, and lower performance while accessing to the e-commerce web application relative to demand. Meanwhile, the microservices architecture is a new trend that is growing rapidly in the business world, but there is little literature related to microservices composition. This research's goal was to propose a microservice composition model aimed towards implementing an e-commerce web application using Kubernetes technology, for this purpose, the model and notation was used for the e-commerce business process, the architectural design of the microservices composition, independent microservices implementation, the evaluation has been done through quality attributes, and validation of this proposed model has been done using load tests. As a result, it was obtained a web application that improved significantly by 104%, referring to performance indicators, availability and response time, all that in comparison to a traditional web application based on the monolithic model. Therefore, this proposed microservice composition model has a significant performance.

Keywords: Availability, electronic commerce, microservice composition model, performance, response time, web application.

[*]Autor para Correspondencia: donializ7@gmail.com

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

INTRODUCCIÓN

El comercio electrónico en el Perú ha crecido, en un 198% en los últimos años y con ello las aplicaciones web (CAPECE, 2016), permitiendo realizar comercio electrónico que consiste en la compra y venta de productos o servicios a través de Internet, estas aplicaciones facilitan a los clientes la manera de realizar sus compras y a las empresas incrementar sus ventas; ocasionando una demanda de usuarios concurrentes en un tiempo determinado, que trae consigo problemas en el rendimiento, la disponibilidad y tiempo de respuesta cuando los usuarios intentan acceder a las funcionalidades de una aplicación web; es por ello que las diferentes empresas van adoptando el cambio de paradigma en el desarrollo de software, con la finalidad de enfatizar más en el proceso de negocio a través de arquitecturas agiles y flexibles (Cockcroft *et al*., 2016), adaptables a los cambios continuos que ocurre en los modelos de negocios de estas organizaciones

En el Perú, de la totalidad de las empresas, el 13.6% desarrolla su propio software, debido a que presentan un modelo de negocio dinámico que tiene que ser reflejado en el software que utilizan (INEI, 2016); sin embargo, actualmente la construcción de software con arquitecturas monolíticas es común entre las diferentes empresas, pero cuando existe los desafíos de la escalabilidad, la flexibilidad, el desarrollo rápido, el corto tiempo de lanzamiento al mercado y la colaboración del equipo de desarrollo, que son cada vez más amplios y variados; la construcción de software, en específico, la elección de la arquitectura es un punto crítico y complejo para lograr la competitividad empresarial. Para llevar a cabo la tarea de desarrollar software que pueda soportar y escalar este tipo de situaciones de crecimiento continuo y circunstancias cambiantes, se han venido utilizando diferentes estilos arquitectónicos de software, entre ellos, uno de los más usados ha sido el estilo arquitectónico por capas, conocido como el modelo monolítico, el cual ha funcionado muy bien para la mayoría de problemas de pequeñas y medianas empresas (Mazlami *et al.*, 2017), pero con limitaciones para las situaciones que existen en la actualidad, que se cuenta con una gran concurrencia de usuarios en instantes de tiempo.

La arquitectura del software describe aspectos estructurales de un software en particular, es decir los componentes físicos y lógicos con sus propiedades y las relaciones entre ellos (Bass *et al*., 2012), enfocándose en los problemas estructurales, como los protocolos para la comunicación, la sincronización, el acceso a datos, la distribución física, la composición de elementos de diseño, la escala y el rendimiento (Garlan y Shaw, 1993). Estos componentes, en la arquitectura tradicional o monolítica, están estrechamente vinculados entre sí, y su desarrollo, implementación y administración se realiza como una sola entidad que ese ejecuta en un solo proceso del sistema operativo y se escala mediante la replicación de todas las funciones en servidores múltiples (Fowler y Lewis, 2014). Por otro lado, la arquitectura del microservicio es un estilo que ha ido ganando popularidad cada vez más en los últimos años, tanto en la literatura de la arquitectura del software (Pahl y Jamshidi, 2016), evidenciado en los congresos de ingeniería de software; y en el ámbito empresarial, en el que las empresas reconocidas tales como Netflix, Twitter y entre otras, han adoptado la arquitectura de microservicios (Newman, 2015). El uso de esta arquitectura y su comparación con la arquitectura monolítica conlleva a ventajas de la primera sobre la segunda (Richardson y Smith, 2016), en el que los servicios individuales son más fáciles de entender y pueden desarrollarse e implementarse independientemente de otros servicios.

La composición de servicios es un principio de las arquitecturas orientadas al servicio, que consiste en reutilizar y combinar servicios existentes con el fin de lograr una funcionalidad nueva

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

o superior (Haupt *et al*., 2014), permitiendo automatizarlas para impulsar la automatización de servicios en la web (Fernández *et al.*, 2010).

En la línea de arquitectura de software, en el tema de composición de servicios, se han realizado investigaciones en técnicas de composición de servicios (Garriga *et al*., 2016), enfocadas a los servicios tradicionales (Bellido, 2015), enfatizando la composición en la calidad del servicio (Pejman *et al*.,2012), sin embargo la composición de servicios y la composición de microservicios difiere en su integración (Eberhard, 2016), esta solución de integración en una arquitectura basada en microservicios no posee inteligencia, sin embargo (Xiao *et al.*, 2017) se debe aprovechar lo mejor de cada composición para poseer la capacidad de responder a las necesidades del mundo digital.

Actualmente existe una limitada literatura de composición de microservicios (Dragoni *et al*., 2017), incluso se cuenta con pocas experiencias en el desarrollo de sistemas basados en microservicios (Vivar, 2015), por lo que no se tiene una referencia clara del proceso de construcción de aplicaciones basados en microservicios (Kharbuja, 2016), que refleja una falta de investigación empírica repetible sobre el diseño, desarrollo y evaluación de aplicaciones de microservicios (Aderaldo *et al* , 2017). Sin embargo existe algunas experiencias de desarrollo de aplicaciones utilizando Docker como tecnología de contenedores (Ueda *et al.*, 2016), tecnología que ayuda eficazmente en el aprovechamiento de la arquitectura de microservicio (Jaramillo *et al.*, 2016), ya que éstas están intrínsecamente relacionadas con los contenedores basados en la nube para la implementación de microservicios (Savchenko *et al.*, 2015).

En la investigación se tuvo como objetivo proponer un modelo de composición de microservicios para la implementación de una aplicación web de comercio electrónico utilizando Kubernetes como tecnología de contenedores.

MATERIALES Y MÉTODOS

La investigación tomo como referencia una organización que brinda servicio de comercio electrónico, esta organización es globalmente competitiva y cuenta con una gran cantidad de usuarios con mayor exigencia que acceden en forma concurrente en un tiempo determinado a la petición de los servicios que brinda la aplicación de comercio electrónico, estas peticiones de consumo, fueron simuladas por la herramienta tecnológica Locust, donde se tuvo limitaciones en el ancho de banda, por lo que consideró 20 peticiones por segundo como muestra accesible representativa.

La investigación es de tipo experimental tecnológico que manipula directamente la variable independiente, el modelo de composición de microservicios; para medir los efectos en la variable dependiente, el funcionamiento de la aplicación web de comercio electrónico, en los indicadores de rendimiento, disponibilidad y tiempo de respuesta.

En el desarrollo de la aplicación web de comercio electrónico se utilizó la metodología de desarrollo de software XP, conocida como la metodología extrema, que es una metodología ágil centrada en la programación de la solución software. En el análisis de los requerimientos para la implementación de la aplicación web de comercio electrónico, se utilizó las técnicas de historias de usuario y tareas de ingeniería (Letelier y Penadés, 2006) para la captura de requerimientos. En el diseño, se realizó énfasis en la arquitectura física y lógica a través de diagramas arquitectónicos. En la implementación del modelo de composición de microservicios se utilizó la programación

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

orientada a objetos y como infraestructura tecnológica a Kubernetes, que es un sistema de orquestación de contenedores. Finalmente en la evaluación del modelo de composición de microservicios propuesto, se empleó la técnica de pruebas de carga (Jiang, 2015).

Modelo composición de microservicios

El modelo propuesto se presenta a través del álgebra de composición de servicios, notación propuesta por Hamadi y Benatallah (2003), y el diseño de la arquitectura de composición de microservicios.

Se muestra los microservicios identificados a partir del diagrama del proceso de negocio de ventas en línea (Tabla 1).

Tabla 1. Microservicios y tareas seleccionadas para la composición.

Microservicios	Tareas	Código	Tipo de petición	Rutas finales
Usuarios	Inicio de sesión	AS1	POST	/inicioSesion
Productos	Ver lista productos	CL1	GET	/productos
	Ver detalle de producto	CP2	GET	/productos/detalle/{{n}}
	Buscar productos	CB3	POST	/productos/búsqueda
Ventas	Agregar a carrito de compra	VA1	POST	/ventas/carrito
	Comprar productos	VC2	POST	/ventas/checkout
	Cantidad de ítems en carrito	VR3	GET	/ventas/carrito

El álgebra de composición de microservicios de la aplicación web de comercio electrónico está dado por:

$$C_{co} = \{[(CB3 \diamond CL1) \odot CP2]^n \odot (AS1 \odot VA1)^m \odot VC2\} /\!/ VR3$$

$Donde$:

$/\!/ : Invocación\ paralela.$

$\diamond : Invocación\ alternativa.$

$\odot : Invocación\ secuencial.$

$n, m: iteraciones.$

$m <= n.$

En la fórmula de composición del servicio de compras en línea, se identifica peticiones a las funcionalidades de la aplicación web en forma paralela, alternativa, secuencial y por iteraciones, que sirvieron de base para las pruebas de la composición de microservicios de manera cuantificable. Así, la funcionalidad de encontrar un producto con el buscador del catálogo y la de seleccionar producto de la lista de productos por categoría, son actividades alternativas. Seguidamente el usuario visualiza el detalle del producto seleccionado, ésta tarea lo puede realizar en forma cíclica hasta que el usuario decide agregar al carrito de compra, pero éste requiere ser autenticado por el sistema, en forma paralela se puede observar el resumen de carrito de compra;

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

esta tarea se puede realizar también de forma cíclica, lo que significa que el usuario puede agregar varios productos a su carrito de compra, finalmente se concretiza el servicio al realizar la compra.

En la Figura 1, se muestra, la relación de los componentes físicos y virtuales de la arquitectura del modelo propuesto. La comunicación de los microservicios entre sí se realiza a través de interfaces de programación de aplicaciones (API), éstas APIs fueron expuestas como puntos finales de tipo REST.

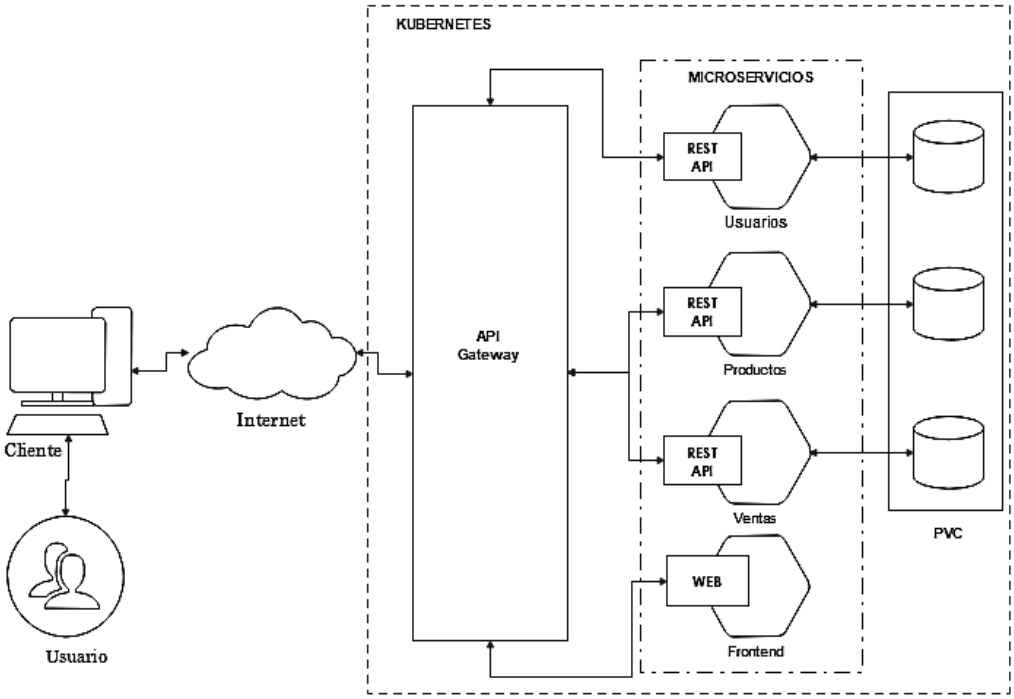

Figura 1. Arquitectura física del modelo de composición de microservicios.

La arquitectura física del modelo propuesto presenta los componentes físicos: el cliente que puede ser un navegador Web, el clúster compuesto por 4 nodos de 3.75 GB de memoria RAM para cada microservicio, un componente PVC (Persistent Volume Claim) para la persistencia de la base de datos, un API Gateway, que utiliza el servidor proxy NGINX, encargado del descubrimiento de los microservicios por DNS o HTTP.

RESULTADOS Y DISCUSIÓN

En la validación del modelo propuesto se realizó las pruebas de carga a la aplicación web de comercio electrónico, compuesta por microservicios. Las peticiones para las pruebas de carga fueron simuladas como utilizando la herramienta tecnológica Locust, considerándose 500 usuarios, con una tasa de eclosión de 10 usuarios, que realizan peticiones por segundo a cada funcionalidad de la aplicación web, que es lograda por la composición de microservicios (Haupt, *et al*., 2014) al igual que en las investigaciones de Dragoni *et al.,* (2017), Kharbuja (2016) y Vivar (2015).

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

La prueba de carga se desarrolló durante un tiempo considerable para alcanzar las 20 peticiones por segundo. Los resultados de esta prueba se orientaron en los indicadores de tiempo de respuesta, disponibilidad y rendimiento, que se presentan a continuación:

Tiempo de respuesta

El resumen indicador tiempo de respuesta, referido al tiempo empleado por la solicitud de servicio del cliente y la respuesta final del servidor (Tabla 2).

Se muestra que cuando se alcanza a las 20 peticiones por segundo, el modelo propuesto mejora en un 75 % el tiempo de respuesta del modelo monolítico.

Tabla 2. Tiempo de respuesta del modelo de composición de microservicios propuesto y la arquitectura monolítica.

	Tiempo de respuesta(ms)		% de mejora
peticiones/segundo	Monolítico	Modelo propuesto	
2	498	190	62
4	598	225	62
6	1700	299	82
8	1692	376	78
10	2212	393	82
12	2125	451	79
14	2403	444	82
16	2512	512	80
18	2630	681	74
20	3021	750	75

Así mismo se observa que las peticiones que son realizadas a la aplicación web de comercio electrónico, implementada con el modelo de composición de microservicios, son respondidas en un tiempo menor, lo que significa que el tiempo empleado por la solicitud de inicio del cliente y la respuesta final del servidor es menor, debido a que cada microservicio efectúa una funcionalidad específica (Fernández *et al*., 2010) y está almacenado dentro un contenedor independiente que es parte de un clúster, teniendo una ventaja la utilización de la tecnología de contenedores que Jaramillo, *et al.* (2016) menciona, sin embargo se delimita a 20 peticiones por minuto debido a las limitaciones del ancho de banda que se otorga al lado cliente. Por otro lado en el modelo monolítico, los servicios de la aplicación web de comercio electrónico, implementadas en forma monolítica, están almacenadas dentro de un mismo servidor y esto conlleva a mayor tiempo en dar respuesta a las peticiones concurrentes y conlleva a tener mayor número de errores en dar respuesta a cada petición que menciona Mazlami *et al.*, (2017); sin embargo los errores que surgen en el modelo propuesto, es debido a que no se realizó una composición utilizando algoritmos inteligentes como Bellido (2015), en la composición de servicios tradicionales, sino que se realizó la composición en función a la integración como

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

menciona Eberhard (2016) donde la infraestructura tecnológica, en este caso los contenedores que fueron orquestados haciendo uso de la tecnología de Kubernetes tuvo un rol primordial.

Disponibilidad

Se muestra el resumen indicador disponibilidad, referida a la capacidad de la aplicación web para brindar un servicio 24/7, que consiste en brindar un servicio las 24 horas del día durante los siete días de la semana en diferentes zonas horarias, a los usuarios que la accedan. Este indicador fue calculado a través del número de peticiones y el número de errores que surgieron durante las peticiones a la aplicación web (Tabla 3).

Tabla 3. Disponibilidad del modelo de composición de microservicios propuesto y la arquitectura monolítica.

	Disponibilidad (%)		**% de mejor.a**
peticiones/segundo	**Monolítico**	**Modelo propuesto**	
2	100	100	0
4	100	100	0
6	95	100	5
8	79	100	21
10	68	96	28
12	70	93	23
14	57	86	29
16	52	87	35
18	43	86	43
20	37	81	44

Se muestra, que ambos modelos responden a 100% de disponibilidad inicialmente, cuando se realiza de 6 peticiones por segundo a más, se va observando como baja la disponibilidad del modelo monolítico, el cual llega a perder el 63% de la disponibilidad al realizar 20 peticiones por segundo, también se observa que en el modelo propuesto solo llega a perder 19% en disponibilidad al realizar 20 peticiones por segundo, (Tabla 3).

mostrando una mayor disponibilidad que el modelo monolítico, mejorando de ésta forma en un 44%, es decir que las funcionalidades de la aplicación de comercio electrónico brinda un servicio 24/7, debido a que se aprovechó la ventaja de la arquitectura de microservicios en descomponer las funcionalidad de la aplicación web en servicios pequeños reduciendo la complejidad Xiao *et al.*(2017) y que son más fáciles de entender y pueden desarrollarse e implementarse independientemente de otros microservicios (Richardson, 2014), y la comunicación de estos microservicios se realizaron de forma liviana utilizando REST como protocolo de comunicación (Garriga *et al*., 2016) haciendo uso de la tecnología de contenedores (Savchenko *et al.*, 2015); con estos resultados se pudo realizar los cambios a las funcionalidades del servicio en forma continua (Cockcroft *et al.*, 2016).

Rendimiento

Se muestra, que durante el tiempo estimado correspondiente a un minuto, se obtuvo un rendimiento óptimo en ambos modelos, pero a partir de seis peticiones por segundo en adelante, el modelo monolítico disminuye su rendimiento debido a su arquitectura monolítica; en cuanto el

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

modelo de composición de microservicios se observa que mantiene un rendimiento óptimo hasta las 16 peticiones por segundo, debido a que se diseñó una arquitectura escalable y distribuida (Pahl y Jamshidi, 2016) (Tabla 4).

Tabla 4. Rendimiento del modelo de composición de microservicios propuesto y la arquitectura monolítica.

Rendimiento(peticiones/minuto)			% de mejora
peticiones/segundo	Monolítico	Modelo propuesto	
2	120	120	0
4	240	240	0
6	350	360	3
8	408	480	18
10	510	600	18
12	503	698	39
14	495	814	64
16	422	820	94
18	400	743	86
20	252	740	194

A partir de las 18 peticiones por segundo, baja su rendimiento debido a las limitaciones del ancho de banda de la red. Por tanto el modelo propuesto presenta una mejora en el rendimiento en un 194% del modelo monolítico lo que apoya a Ueda *et al.*, (2016) y Jaramillo *et al*., (2016) de que la tecnología de contenedores como infraestructura tecnológica para los microservicios mejora el rendimiento y ayuda eficazmente en el aprovechamiento de la arquitectura microservicios. Así mismo comprueba que el modelo propuesto brinda mejor rendimiento, que la arquitectura debe soportar. (Garlan y Shaw, 1993).

Se muestra el funcionamiento de la aplicación web de comercio electrónico, observándose que tiene una mejora del 104% del modelo monolítico en los indicadores de tiempo de respuesta, disponibilidad y rendimiento, debido a que esta aplicación fue desarrollada como composición de un conjunto de microservicios a través de la orquestación (Newman, 2015), cada uno ejecutándose en su propio servidor y comunicándose en forma liviana a través de una API (Fowler y Lewis, 2014) (Tabla 5).

Tabla 5. Funcionamiento de la aplicación web de comercio electrónico basado en el modelo de composición de microservicios propuesto.

Indicador	% De Mejora por Indicador	% De Mejora Total
Tiempo de respuesta	75	
Disponibilidad	44	104
Rendimiento	194	

En la obtención de requerimientos para el desarrollo de la aplicación web se enfatizó en los requerimientos no funcionales expresadas a través de los atributos de calidad (Pejman *et al*.,2012), como la eficiencia, rendimiento, disponibilidad, escalabilidad y seguridad, recomendadas por Bass *et al.*, (2012), y de ésta forma se aporta en la fase de análisis, en específico en la recolección

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

de datos del proceso de desarrollo de software basado en la arquitectura de microservicios Aderaldo *et al.*, (2017).

CONCLUSIONES

La implementación de una aplicación web de comercio electrónico, basado en el modelo de composición de microservicios, funciona significativamente, en comparación con una aplicación web basado en el modelo monolítico, en un 104% con respecto a los indicadores de rendimiento, disponibilidad y tiempo de respuesta. La aplicación web de comercio electrónico fue desarrollado bajo una arquitectura de microservicios, lo cual permitió la separación de funcionalidades del proceso de negocio, al dividirlo en pequeñas partes manejables a través de casos de uso, enfocadas a funcionalidades específicas para ser implementadas en forma independiente, así mismo se tomó en cuenta los requerimientos en función a los requerimientos arquitectónicos de la aplicación web, como son la disponibilidad, eficiencia, rendimiento, escalabilidad y seguridad. A nivel de infraestructura se utilizó las tecnologías de contenedores como Docker y la gestión de éstas con como Kubernetes, los cuales difieren del soporte necesario para una aplicación basado en la arquitectura monolítica.

LITERATURA CITADA

Aderaldo, M., Mendonça, C., Pahl, C., y Jamshidi, P. (2017). Benchmark Requirements for Microservices Architecture Research. *IEEE/ACM 1st International Workshop on Establishing the Community-Wide Infrastructure for Architecture-Based Software Engineering (ECASE), At Buenos Aires, Argentina*, (March), Microservices have recently emerged as a new ar-c. https://doi.org/10.1109/ECASE.2017..4

Bass, L.; Clements, P.; y Kazman, R. (2012). Software Architecture in Practice. *Vasa*, *2nd*, 1–426. https://doi.org/10.1024/0301-1526.32.1.54

Bellido, J. (2015). Dynamic Composition of REST services, 1–23.

CAPECE. (2016). Cámara peruana de comercio electrónico. Retrieved January 1, 2016, from http://www.cacepe.com

Cockcroft, A., Fellow, T., & October, B. V. (2016). Microservices Workshop : Why , what , and how to get there, (October).

Dragoni, N., Giallorenzo, S., Lafuente, L., Mazzara, M., Montesi, F., Mustafin, R., y Safina, L. (2017). Microservices : yesterday , today , and tomorrow.

Eberhard, W. (2016). *Microservices: Flexible Software Architecture* (file:///C:).

Fernández-Villamor, J. I., Iglesias, C. a., & Garijo, M. (2010). Microservices: lightweight service descriptions for REST architectural style. *Proc. of Second International Conference on Agents and Artificial Intelligence. Valencia, Spain*. Retrieved from http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:MICROSERVICES+:+LIGHTWEIGHT+SERVICE+DESCRIPTIONS+FOR+REST+ARCHITECTURAL+STYLE#0

Fowler, M., y Lewis, J. (2014). Microservice Architecture. *Martinfowler.Com*. https://doi.org/10.1007/978-1-4842-1275-2_3

Garlan, D., y Shaw, M. (1993). An introduction to software architecture. *Advances in Software Engineering and Knowledge Engineering*, *1*(January), 1–39. https://doi.org/10.1142/9789812798039_0001

Garriga, M., Mateos, C., Flores, A., Cechich, A., y Zunino, A. (2016). RESTful service composition at a glance: A survey. *Journal of Network and Computer Applications*. https://doi.org/10.1016/j.jnca.2015.11.020

Hamadi, R., y Benatallah, B. (2003). A Petri net-based model for web service composition. *Proceedings of the 14th Australasian Database Conference-Volume 17*, *12*(4), 191–200. https://doi.org/10.1007/s11741-008-0409-2

Haupt, F., Fischer, M., Karastoyanova, D., Leymann, F., & Vukojevic-Haupt, K. (2014). Service Composition for REST. In *Proceedings . IEEE 18th international Enterprise Distributed object computing conference* (Vol. 2014–Decem, pp. 110–119). https://doi.org/10.1109/EDOC.2014.24

INEI. (2016). Perú : Tecnología de Información y. *Encuesta Económica Anual 2015*.

Jaramillo, D., Nguyen, D. V., & Smart, R. (2016). Leveraging microservices architecture by using Docker technology. In *Conference Proceedings - IEEE SOUTHEASTCON* (Vol. 2016–July). https://doi.org/10.1109/SECON.2016.7506647

Jiang, J. (2015). Load Testing Large-Scale Software Systems. In *Proceedings - International Conference on Software Engineering* (Vol. 2, pp. 955–956). https://doi.org/10.1109/ICSE.2015.304

Kharbuja, R. (2016). Desingning a Business Platform using Microservices. Retrieved from http://mediatum.ub.tum.de/doc/1285460/1285460.pdf

Letelier, P., & Penadés, M. C. (2006). Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Técnica Administrativa*, *5*(26), 17. https://doi.org/1666-1680

UNIVERSIDAD NACIONAL DEL ALTIPLANO
REVISTA DE INVESTIGACIONES DE LA ESCUELA DE POSGRADO
Versión electrónica ISSN 2077 – 8686
Versión Impresa ISSN 1997-4035
ARTÍCULO ORIGINAL

Vol. 7 No.3, pp. 728-737, Julio/Setiembre 2018
Recibido 22/01/2018
Aceptado 23/01/2018
Publicado 31/07/2018
DOI: http://dx.doi.org/10.26788/riepg.2018.3.95

Mazlami, G., Cito, J., y Leitner, P. (2017). Extraction of Microservices from Monolithic Software Architectures. In *Proceedings - 2017 IEEE 24th International Conference on Web Services, ICWS 2017* (pp. 524–531). https://doi.org/10.1109/ICWS.2017.61

Newman, S. (2015). *Building Microservices - Chapter 1, 4 and 11*. *Building Microservices*.

Pahl, C., & Jamshidi, P. (2016b). Microservices: A Systematic Mapping Study. *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, (October), 137–146. https://doi.org/10.5220/0005785501370146

Pahl, C., & Jamshidi, P. (2016a). Microservices: A Systematic Mapping Study. *Proceedings of the 6th International Conference on Cloud Computing and Services Science*, (May), 137–146. https://doi.org/10.5220/0005785501370146

Pejman, E. ., Rastegari, Y. ., Majlesi Esfahani, P. ., & Salajegheh, A. . (2012). Web service composition methods: A survey. In *Lecture Notes in Engineering and Computer Science* (Vol. 2195, pp. 603–607). Retrieved from https://www.scopus.com/inward/record.uri?eid=2-s2.0-84867459385&partnerID=40&md5=5f92f96ca5bc81904b0af6a9a18e8a08

Richardson, C. (2014). Microservices: Decomposing Applications for Deployability and Scalability. Retrieved from http://www.infoq.com/articles/microservices-intro

Richardson, C., y Smith, F. (2016). Microservices - From Design to Deployment. *Nginx*.

Savchenko, D. I., Radchenko, G. I., & Taipale, O. (2015). Microservices validation: Mjolnirr platform case study. In *2015 38th International Convention on Information and Communication Technology, Electronics and Microelectronics, MIPRO 2015 - Proceedings* (pp. 235–240). https://doi.org/10.1109/MIPRO.2015.7160271

Ueda, T., Nakaike, T., y Ohara, M. (2016). Workload characterization for microservices. In *Proceedings of the 2016 IEEE International Symposium on Workload Characterization, IISWC 2016* (pp. 85–94). https://doi.org/10.1109/IISWC.2016.7581269

Vivar, O. (2015). *Plataforma para la Anotación Semántica Automática de Servicios Web RESTful sobre un Bus de Servicios*. Cuenca-Ecuador.

Xiao, Z., Wijegunaratne, I., y Qiang, X. (2017). Reflections on SOA and Microservices. In *Proceedings - 4th International Conference on Enterprise Systems: Advances in Enterprise Systems, ES 2016* (pp. 60–67). https://doi.org/10.1109/ES.2016.14