ISSN 2518-4725.

DISEÑO E IMPLEMENTACIÓN DE UNA ARQUITECTURA ESCALABLE BASADA EN MICROSERVICIOS PARA UN SISTEMA DE GESTIÓN DE APRENDIZAJE

José Manuel De Paz Estrada
Mtro. en Tecnologías
de la Información y la Comunicación
josedepaz9@gmail.com

María Elizabeth Aldana Díaz
Asesora
Mtra. en Ingeniería Informática, IPTV,
IPTV Móvil, Redes de Entrega de Contenido (REC) y Sistemas Distribuidos
maria.aldana.usac@gmail.com

Resumen

El paradigma de arquitectura monolítica utilizado por los principales sistemas de gestión de aprendizaje como Moodle y Blackboard, ha provocado que sus servicios tengan problemas de escalabilidad y disponibilidad durante fallos o nuevos despliegues por actualización.

Por otro lado, tomando en cuenta como antecedentes el éxito de las arquitecturas propuestas por empresas como Netflix y Amazon, se desarrolla un prototipo de sistema de gestión de aprendizaje sobre una arquitectura basada en microservicios que provee escalabilidad y resiliencia.

Para ello se hace uso de los diferentes patrones que componen este paradigma arquitectónico, como lo son: el patrón de instancia por contenedor de software utilizando Docker, el registro de servicios utilizando Consul y Registrator, los servicios REST para la comunicación interna entre servicios utilizando JSON, permite el uso de tecnologías heterogéneas y un API Gateway desarrollado en Node.js con la librería Hapi.js, que provee el manejo y acceso a los servicios internos por parte de clientes externos, exponiéndolos en forma de un proxy.

Palabras clave

Arquitectura, Software, Microservicio, LMS, Escalabilidad

Abstract

The paradigm of monolithic architecture used by leading learning management systems such as Moodle and Blackboard has caused problems of scalability and availability to its services during failures.

On the other hand, taking into account the success of architectures proposed by companies like Netflix and Amazon, the present work was developed a prototype of a learning management system on a microservices architecture that provides scalability and resilience.

For this purpose we developed the instance by software container pattern using Docker, services registry using Consul and Registrator, REST services for internal communication between services using JSON and an API Gateway developed in Nodejs with Hapi.js library.

To verify system performance, the testing was performed by request load simulations with Apache JMeter, resulting in the complete elimination of downtime of a service during the fault of another.

Keywords

Architecture, Software, Microservice, LMS, Scalability.

Introducción

El mundo de la tecnología evoluciona constantemente, debido a las necesidades que surgen día a día, esto ha provocado que vayan apareciendo nuevos retos para las personas relacionadas tanto con la infraestructura tecnológica como con los desarrolladores de nuevo software. Uno de estos retos se ha generado por la cantidad de personas que cada día se unen a la red de información, provocando la necesidad de diseñar nuevas arquitecturas que permitan escalar de forma óptima, y que además permitan realizar cambios muchas veces drásticos asegurando que el o los sistemas implementados en ellas no queden fuera de línea, o si en caso se queden inactivos, que sea la menor cantidad de tiempo posible.

El área académica es una de las que más se han visto beneficiadas con la llegada de nuevos sistemas de información y desde 1993, con Blackboard implementando el primer sistema de aprendizaje en línea, hasta el año 2012 en que se popularizaron los MOOC (cursos en línea masivos y abiertos) han aparecido muchos otros sistemas como Moodle o Dockeos, pero manteniendo la misma filosofía de ese primer sistema de 1993, sistemas limitados muy parecidos a los foros en línea, que permiten ver anuncios de los cursos y descargar material de los mismos. Estos sistemas tienen limitantes en escalabilidad, resiliencia y uso heterogéneo de tecnologías, ya que fueron implementándose sobre arquitecturas monolíticas. (Subramanian, Zainuddin, Alatawi, Javabdeh, y Hussin, 2014)

Una nueva tendencia para atacar estas limitantes, son las arquitecturas basadas en microservicios. Estas arquitecturas comienzan a definirse y a mostrar sus beneficios por el año 2014, siendo utilizadas y apoyadas por empresas grandes del internet como lo son Amazon, Google, Netflix y muchas otras más. (Richardson, 2014)

El presente estudio diseña a una arquitectura basada en microservicios que proporciona alta disponibilidad, escalabilidad y facilidad de despliegue en un entorno tecnológico heterogéneo para implementar el prototipo de un sistema de gestión de aprendizaje.

Desarrollo del estudio

El desarrollo del prototipo de sistema de gestión de aprendizaje sobre una arquitectura basada en microservicios, se puede observar en la Figura 1.

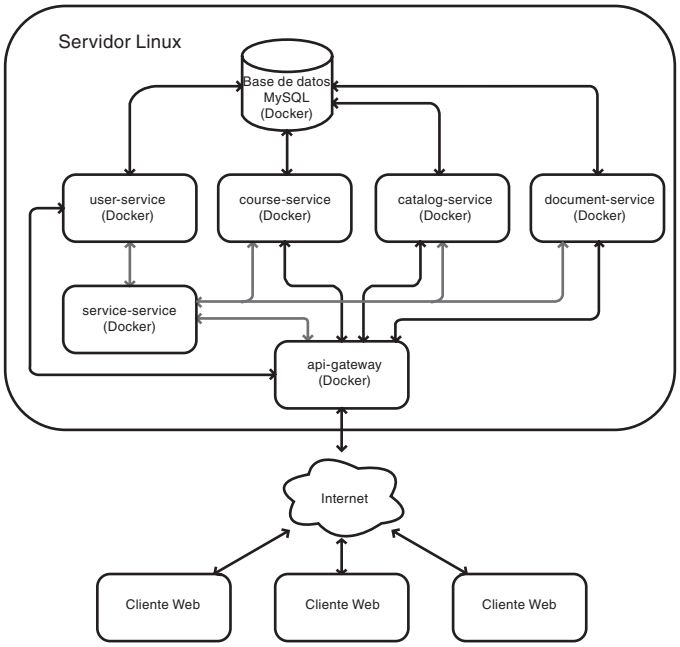

Figura 1. Arquitectura de sistema de gestión de aprendizaje basada en microservicios.

Por otro lado, se utilizan cuatro patrones arquitectónicos importantes, el primero de ellos es la implementación de instancias por contenedor de software, para ellos se utilizó Docker.

Otro patrón utilizado es el registro y descubrimiento de servicios, el cual se realiza de forma automática utilizando Consul y Registrator, esto permite el descubrimiento de servicios por medio de HTTP.

Para la comunicación interna entre los servicios, se utilizan servicios REST, puesto que al utilizar solicitudes JSON, estas son ligeras y gracias a la cantidad de librerías disponibles de JSON para la mayoría de lenguajes de programación, se permite que el sistema sea implementado utilizando tecnologías heterogéneas y además por ser servicios

REST, no se necesita la instalación de nuevos componentes.

Por último, se utiliza el patrón arquitectónico de API Gateway, implementado sobre Node.js, utilizando las librerías Hapi.js y h2o2, se crean servicios REST que pueden ser controlados por medio de proxy y acceder a los servicios internos del sistema. Hapi.js que permite la autenticación y manejo de permisos.

Resultados obtenidos

Para verificar los resultados de la implementación del sistema de gestión de aprendizaje sobre una arquitectura basada en microservicios, se realiza una comparación con el mismo sistema, pero sobre una arquitectura monolítica. Para ello se realiza una prueba de rendimiento utilizando JMeter, sobre las dos arquitecturas.

La prueba de rendimiento consiste en colocar en funcionamiento los servicios de usuarios y cursos, luego de un intervalo de tiempo se detiene el servicio de cursos, simulando una falla o un despliegue de servicio, y se verifica lo ocurrido con el servicio de usuarios, pues al ser dos servicios distintos, este último no debe presentar ningún cambio en su funcionamiento.

La arquitectura monolítica, como era de esperarse, demuestra un buen redimiendo en cuanto el sistema se pone en ejecución, pero demuestra que cuando se detiene, el servicio de usuarios se ve afectado, incluso se queda inactivo durante el mismo tiempo de inactividad del servicio de cursos. El tiempo de respuesta de las solicitudes al servicio de usuarios, durante la inactividad del servicio de cursos es igual a cero.

Este tiempo de respuesta se puede tomar como bueno, pero cuando se observa la Figura 4, se verifica que el tiempo de respuesta fue igual a cero, porque en realidad no hubo respuesta por parte del servicio de usuario. Este tiempo equivale a un 20 % del tiempo en que fue realizada la prueba.

Por otro lado, cuando se realiza la misma prueba de rendimiento en el sistema sobre una arquitectura basada en microservicios, el tiempo de respuesta del servicio de usuarios permanece en un promedio de 7.5 milisegundos, y no cambia su comportamiento durante los 4 segundos que se mantuvo inactivo el servicio de cursos.

A continuación, se confirma que el servicio de usuarios retornó el código 200 (éxito) a todas las solicitudes realizadas durante la prueba de rendimiento.

Discusión de resultados

Con base ena las pruebas de rendimiento realizadas, se observa cuál es la diferencia en el comportamiento del prototipo del sistema de gestión de aprendizaje, cuando éste se implementa sobre una arquitectura monolítica y cuando se implementasobre una arquitectura basada en microservicios.

Para observar estas diferencias de comportamiento, primero se realiza la prueba en el sistema sobre una arquitectura monolítica, y como resultado se observa que existe un intervalo de tiempo, durante el cual la latencia para obtener una respuesta del servicio de usuarios es igual a 0, esto, se debe a los 4 segundos durante los cuales el servicio de cursos no se encuentra disponible, lo cual provoca que todo el sistema de gestión de aprendizaje, se detenga incluyendo al servicio de usuarios.

A diferencia de los resultados mostrados para la arquitectura monolítica, cuando la prueba de rendimiento se realiza sobre la arquitectura basada en microservicios, se observa cómo el servicio se comporta de una forma tolerante a fallos mostrando su capacidad de resiliencia, ya que retornó los datos de forma exitosa a todas las solicitudes realizadas, no importando si otro servicio se encuentra detenido por una actualización o por un fallo en el mismo momento.

Conclusiones

1. Se implementa el prototipo de un sistema de gestión de aprendizaje sobre una arquitectura basada en microservicios, que permite escalabilidad y una disponibilidad del 100 % durante inactividad o fallos de otros servicios.

2. Se implementa un patrón de instancias de servicio por contenedor utilizando Docker, que permite reducir el tiempo de inactividad de los servicios durante un fallo o nuevo despliegue del 20 % a 0 %, eliminando por completo el tiempo de inactividad.
3. Se implementa un patrón arquitectónico de registro de servicios por medio de Consul y Registrator, que permite dar de alta y de baja de forma automática, a las instancias de los servicios del sistema.
4. Se implementan servicios REST sobre el protocolo HTTP para la comunicación entre los diferentes servicios del sistema, los cuales permiten el uso de diferentes lenguajes de programación y, además, al utilizar JSON se logra que las solicitudes y los retornos de los servicios sean ligeros.
5. Se implementa una puerta de enlace API Gateway utilizando Node.js, Hapi.js y H2O2, para crear un proxy para manejar el acceso a los servicios del sistema por parte de clientes externos, que ahorra recursos por su modelo de no bloqueo de entrada y salida.

Recomendaciones

A futuros investigadores se le sugiere:

1. Realizar previamente al desarrollo, un análisis profundo para determinar si es necesario implementar el sistema sobre una arquitectura basada en microservicios, para sistemas muy pequeños, se puede convertir en algo muy complejo el manejo de la infraestructura.
2. Utilizar Docker Compose para el proceso de creación y ejecución de contenedores Docker cuando no es necesario utilizar un orquestador de contenedores.
3. Utilizar herramientas para el monitoreo de espacio en disco, uso de CPU y de memoria RAM de los contenedores, para determinar el límite que se le debe configurar a cada uno de ellos.
4. Analizar herramientas para orquestación de contenedores como Docker Swarm o Kubernetes de Google para automatizar el manejo de los contenedores, cuando se realiza el desarrollo de sistemas muy grandes.

Referencias bibliográficas

Raj, P., Chelladhurai, J. S., Singh, V. (2015). Learning Docker. Bangalore, India, Birmingham, Reino Unido. Pack Publishing.

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

Subramanian, P., Zainuddin, N., Alatawi, S., Javabdeh, T. y Hussin, A. R. C. (2014). A Study of Comparison between Moodle and Blackboard based on Case Studies for Better LMS. Journal Of Information Systems Research And Innovation. Recuperado de http://seminar.utmspace.edu.my/jisri/download/F_Vol6Feb2014_FullPaper/Pub4_ComparisonBetweenMoodleAndBlackboard.pdf

Información del Autor

Ingeniero en Ciencias y Sistemas, Facultad de Ingeniería, Universidad de San Carlos de Guatemala (USAC), 2012.

Maestro en Artes en Tecnologías de la Información y la Comunicación, Escuela de Estudios de Postgrado, Facultad de Ingeniería (USAC), 2016