

Grado Ingeniería de Sistemas Audiovisuales
2018-2019

Trabajo Fin de Grado

“Diseño e implementación de un microservicio con Spring”

Jesús Rienda Iáñez

Tutor/es

Carmen Pelaez Moreno

Leganés, 2019



[Incluir en el caso del interés en su publicación en el archivo abierto]

Esta obra se encuentra sujeta a la licencia Creative Commons **Reconocimiento - No Comercial - Sin Obra Derivada**

RESUMEN

Palabras clave:

DEDICATORIA

ÍNDICE GENERAL

1. INTRODUCCIÓN.	1
1.1. Planteamiento del problema.	1
1.2. Solución propuesta.	1
1.3. Justificación de la solución	1
1.4. Estado del arte	2
BIBLIOGRAFÍA	5

ÍNDICE DE FIGURAS

1.1	Monolítica frente Microservicios	2
1.2	Eficiencia microservicios frente SOA	3

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Planteamiento del problema

Tenemos la necesidad almacenar información en una base de datos para posteriormente consultarla, actualizarla o crear nuevos registros. Por lo que necesitamos crear un servicio que al invocar nos devuelva los datos almacenados con un tratamiento específico y un formato definido. Necesitamos que sea sencillo y simple para el cliente que va a consumir dicho servicio. Este servicio tendrá que tener una alta disponibilidad y escalarse cuando sea necesario para siempre tener unos tiempos de respuesta bajos.

1.2. Solución propuesta

Para poder resolver nuestro problema usaremos un protocolo REST para la transferencia de datos entre servicio y cliente. El servicio web será un microservicio desarrollado con Spring Boot. En cuanto a la base de datos utilizaremos PostgreSQL.

1.3. Justificación de la solución

La arquitectura de microservicios pretende dividir una aplicación compleja en pequeños servicios que solo realicen una función específica y se comuniquen entre ellos para formar la aplicación final.

Cada microservicio es totalmente independiente de desarrollar frente al conjunto, lo cual nos viene genial ya que nuestra idea es realizar una pequeña aplicación que acceda a una base de datos y en un futuro ampliar a varias aplicaciones o incluso un frontal. Los microservicios nos permiten desarrollar cada una en un lenguaje de programación diferente.

Al no disponer de una máquina donde desplegar la aplicación, es ideal que los microservicios lleven un servidor de despliegue (tomcat) embebido y así desplegar en la nube dentro de un contenedor de aplicaciones (Kubernetes).

Una vez desplegado en la nube decidimos que la comunicación sería mediante REST ya que es un protocolo simple y muy eficaz para realizar las distintas operaciones (verbos) en base de datos: añadir, recuperar, actualizar y eliminar, esto en REST sería GET, POST, PUT y DELETE.

En cuanto a base de datos hemos elegido PostgreSQL ya que es open source y totalmente compatible con muchos lenguajes de programación, no solo con Java que es el caso de nuestra aplicación, sino que si en un futuro queremos crear otro microservicio

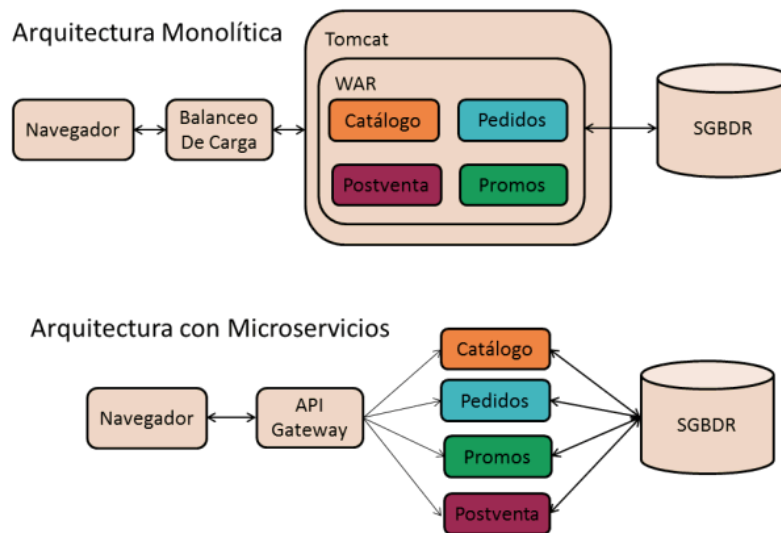


Fig. 1.1. Monolítica frente Microservicios

con python sería posible reutilizar la base de datos. Además también es capaz de responder a gran cantidad de peticiones en un mismo instante y no bloquearse, esto es totalmente esencial ya que necesitamos alta disponibilidad.

1.4. Estado del arte

En la década de los 60 surgió lo que a día de hoy conocemos como arquitectura de software, esta fue tomando cada vez más interés hasta que en la década de 1980 se integró totalmente el diseño en el desarrollo de software.

Primero surgió la arquitectura orientada objetos más adelante la orientada a componentes. Pero no fue hasta 1996 cuando se desarrolló por primera vez SOA, arquitectura orientada a servicios. En ella se desarrollaban todos los servicios que tu necesitas conjuntamente y se empaquetan en un war el cual se despliega en un servidor de aplicaciones (tomcat) dentro de una máquina, esto lo podemos ver en la figura 1.1.

Todos los servicios tenían que estar desarrollados con el mismo lenguaje y no podías asignar más recursos a uno de ellos sino que se lo asignabas a todo el conjunto, escalando el war en varias máquinas o réplicas en la misma. Para ello necesitabas un balanceador de carga antes que determine qué máquina va a atender tu petición.

Todo esto antes era más que suficiente para las empresas, pero a día de hoy cuando una aplicación monolítica (SOA) crece mucho es difícil mantener y es complicado añadir nuevas funcionalidades, ya que cada línea modificada implica re-desplegar toda la aplicación, lo que en una empresa grande puede llevar bastante tiempo, ya que en los despliegues normalmente están involucrados varios departamentos de la empresa como seguridad, operaciones, arquitectura y desarrollo, que impide al equipo seguir desarrollando. También es complicado encontrar el origen de algún error en el código.

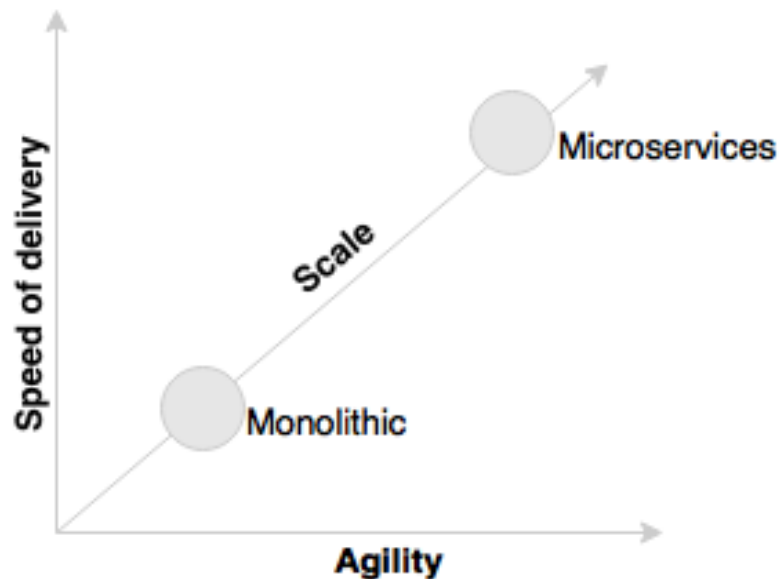


Fig. 1.2. Eficiencia microservicios frente SOA

La necesidad de resolver todos estos problemas desencadenó en la arquitectura de microservicios. La primera vez que se mencionó la palabra "microservicios" fue en 2011 en una conferencia sobre computación en la nube donde el Dr. Peter Rogers[1] se refirió a ello para describir la arquitectura que estaban usando grandes empresas como Netflix, Facebook, Amazon o PayPal.

Los microservicios gestionan la complejidad granulando funcionalmente en un conjunto de servicios pequeños e independientes. Con esto se consigue que el equipo de desarrollo sea capaz de desarrollar varias funcionalidades a la vez sin tocar código de otra funcionalidad y desplegar cada módulo por separado tal y como se ve en la figura 1.1 donde cada microservicio está separado del resto, y puede o no tener una base de datos común.

El cambio más notable respecto a SOA es que los equipos de desarrollo tienen una mayor responsabilidad, lo que se traduce en una gran facilidad, ya que ellos manejan todo el proceso de desarrollo, despliegues en distintos entornos, gestión de contenedores como Kubernetes, etc. Todo esto antes tenían que realizarlo otros departamentos de la empresa con el aumento de tiempos que suponía.

Aunque la arquitectura de microservicios resuelve todos los problemas que presenta SOA y cada vez es más popular, aún está en su base de inicio como se menciona en el artículo[2] y aún le queda mucho por mejorar y evolucionar.

Rajest RV en su libro "Spring Microservices"[3] nos muestra un gráfico 1.2 en el que se ve cómo es mucho más rápido y ágil el desarrollo de aplicaciones con microservicios frente a tradicionales. "Los microservicios prometen más agilidad, velocidad de entrega y escala. En comparación con las aplicaciones monolíticas tradicionales."

En un futuro se deberían solucionar problemas debidos a estar poco restringidos, por

ejemplo si cada microservicio lo desarrollas con un lenguaje diferente no esta del todo claro que los protocolos que uses en cada uno de comunicación sean totalmente compatibles.

En otro tema que tienen que mejorar es en la seguridad, ya que cuando tu descompones una aplicación en cientos de microservicios creas dificultad en la depuración, monitoreo, auditoría y análisis forense de toda la aplicación. Los atacantes podrían aprovechar esta complejidad para atacar.

Lo que si es seguro es que han surgido para quedarse y que cada vez mas gente se esta pasando a ellos.

BIBLIOGRAFÍA

- [1] L. Mauersberger, “A brief history of microservices”, *blog.leanix.net*, 2017. [En línea]. Disponible en: <https://blog.leanix.net/en/a-brief-history-of-microservices>.
- [2] N. Dragoni et al., “Microservices: Yesterday, Today, and Tomorrow”, en *Present and Ulterior Software Engineering*, M. Mazzara y B. Meyer, eds. Cham: Springer International Publishing, 2017, pp. 195-216. doi: 10.1007/978-3-319-67425-4_12. [En línea]. Disponible en: https://doi.org/10.1007/978-3-319-67425-4_12.
- [3] R. RV, *Spring Microservices*. Packt Publishing, 2016. [En línea]. Disponible en: <https://books.google.es/books?id=pwNwDQAAQBAJ>.