uc3m | Universidad Carlos III de Madrid

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

RESUMEN

Palabras clave:

DEDICATORIA

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Planteamiento del problema

Tenemos la necesidad almacenar una serie de datos en una base da datos para posteriormente consultarlos, actualizarlos o crear nuevos registros. Por lo que necesitamos crear un servicio que cuando llamemos nos devuelva los datos almacenados con un tratamiento especifico 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 el servicio y el cliente. El servicio web sera un microservicio desarrollado con Spring Boot. En cuanto a la base de datos utilizaremos una base de datos PostgreSQL.

1.3. Justificacion de la solucion

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

Cada microservico es totalmente independiente de desarrollar frente al conjunto, lo cual nos viene genial ya que nuestra idea es realizar una pequeña aplicación que aceda 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 maquina 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.

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 seria GET, POST, PUT y DELETE.

En cuanto a base de datos hemos elegido PostgreSQL ya que es open source y total-

mente compatible con muchos lenguajes de programación, no solo con Java que es el caso de nuestra aplicación, sino que si en un futuro queremos creamos otro microservicio con python y reutilizar la base de datos está nos servirá. Ademas también es capaz de responder a muchas peticiones al mismo tiempo y no bloquearse y esto es totalmente esencial ya que como ya hemos dicho necesitamos alta disponibilidad.

1.4. Estado del arte

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

Primero surgió la arquitectura orientada objetos mas adelante la orienta a componentes. Pero no fue hasta 1996 cuando se desarrollo por primera vez SOA, arquitectura orientada a servicios. En ella se desarrollaban todos los servicios que tu necesitabas conjuntamente y se empaquetaban en un war el cual se desplegaba en un servidor de aplicaciones(tomcat) dentro de una maquina, esto lo podemos ver en la figura 1.1.

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

Todo esto antes era mas 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 linea que cambies tocara redesplegar, 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 codigo.

La necesidad de resolver todos estos problemas desencadeno en la arquitectura de microservicios. La primera vez que se menciono 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.

1.5. ¿Que es un Microservicio?

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

Cada microservico es una aplicación pequeña totalmente independiente de desarrollar

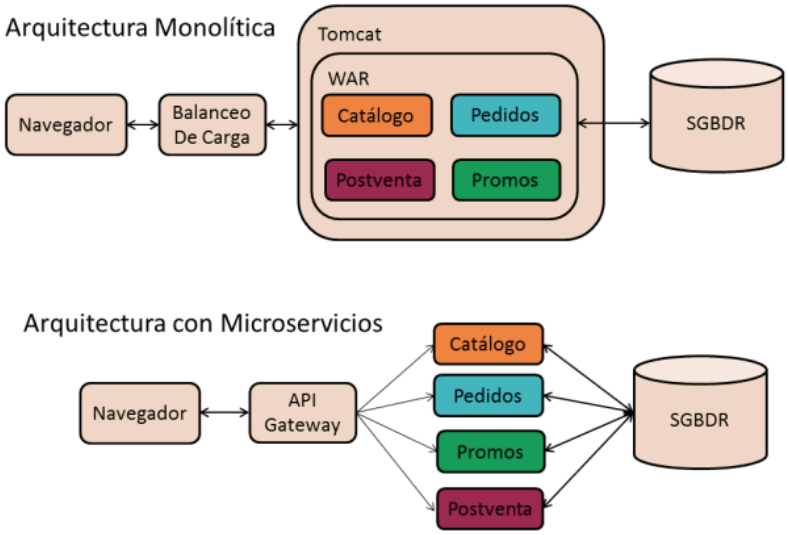

Fig. 1.1. Monolitica vs Microservicios

frente al conjunto. Este servicio tiene responsabilidad única, es decir, un único requisito funcional. Ya que este es pequeño, si falla algo es fácil encontrar el error y subsanarlo.

Ademas al tener un servidor embebido es autoejecutable en cualquier contenedor de aplicaciones, por ejemplo Kubernetes, lo que facilita enormemente su despliegue.

Cada microservicio puede desarrollarse en un lenguaje de programación diferente, gracias a que las peticiones a estos servicios suele ser mediante protocolos ligeros, como HTTP, que son universales a todos los lenguajes de programación. También es fácilmente escalable, se le asignan mas recursos al microservicio que los necesite.

Los expertos en la materia no se ponen de acuerdo en el tamaño que debe tener un microservicio. Sam Newman hace referencia en su libro "Building Microservices"(cita) al experto en microservicios Jon Eaves que el tiempo máximo para desarrollar un microservicio una persona serian dos semanas. Aunque esto es muy relativo, ya que en una empresa grande la funcionalidad que debe realizar cada microservicio no tiene porque ser pequeña, por lo que llevaría mas tiempo.

1.6. Estado del arte

Actualmente la arquitectura basada en microservicios es una realidad y las grandes empresas tales como Netflix, Facebook, Amazon, PayPal se han replanteado la arquitectura que anteriormente usaban para decantarse por los microservicios debido a sus ventajas.

Rajest RV en su libro "Spring Microservices"[2] nos muestra un gráfico en el que se ve como avanza el desarrollo de aplicaciones tradicionales frente a microservicios. "Los microservicios prometen más agilidad, velocidad de entrega y escala. En comparación con las aplicaciones monolíticas tradicionales."

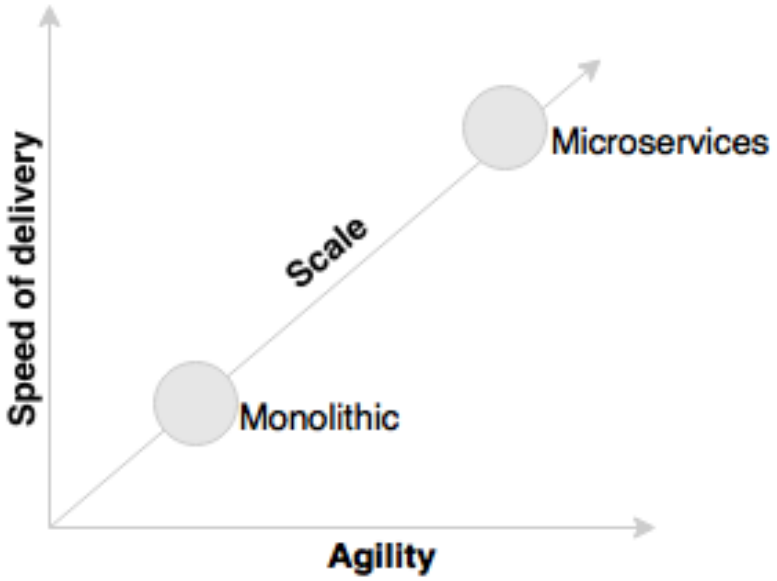

Fig. 1.2

Los microservicios han evolucionado tan rápido gracias al desarrollo de nuevas tecnologías. Por ejemplo la creación de bases de datos NOSQL no relacionales, la aparición de docker contenedores de aplicaciones ligeros y portables que facilitan enormemente los despliegues y el comienzo del desarrollo de aplicaciones en la nube. Todo ello unido ha desembocado en la creación de microservicios.

Existen muchas razones por las cuales hacer el cambio o directamente empezar a desarrollar con esta arquitectura. - Los microservicios son mas rápidos y baratos de desarrollar que las antiguas aplicaciones monolíticas. - Es más fácil remplazar una parte de la aplicación global que en los sistemas antiguos. - No necesitas un servidor tipo tomcat donde desplegar la aplicación, ya que cada microserivcio cuenta con uno embebido. - Se suelen desplegar en la nube dentro de contenedores dockers por lo que no es necesaria una maquina física(servidor) 24 horas funcionando. - Puedes desarrollar cada microservicio en un lenguaje de programación diferente, en función de las necesidades.

Una metáfora que suele usarse para explicar los microservicios es un panal de abeja. Las abejas construyen el panal rellenado pequeñas celdas hexagonales, cada una independiente pero también integrada con las otras. Esto genera una estructura muy fuerte. Si una celda se daña no afecta a las demás, las abejas reconstruyen solo esa celda.

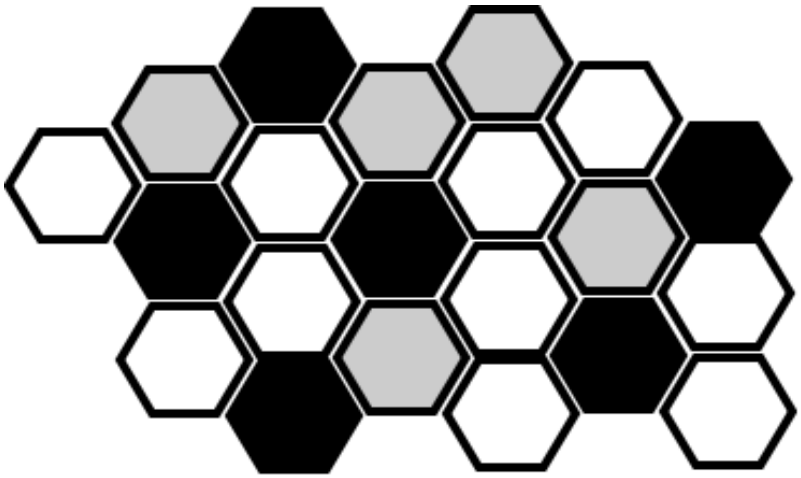

Fig. 1.3

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] R. RV, *Spring Microservices*. Packt Publishing, 2016. [En línea]. Disponible en: https://books.google.es/books?id=pwNwDQAAQBAJ.