Microservices

Víctor Iranzo

6 de julio de 2018

1. Definición

Los microservicios son una aproximación para desarrollar una aplicación compuesta por pequeños servicios, cada uno ejecutándose en su propio proceso y comunicando a través de mecanismos ligeros. Estos servicios se construyen alrededor de las capacidades de negocio y se despliegan de forma independiente.

Las aplicaciones monolíticas se construyen como una única unidad que normalmente se divide en 3 partes: la interfaz de usuario, la parte servidora de la aplicación y la base de datos. La escalabilidad en estas aplicaciones se consigue replicando de forma horizontal toda la parte servidora y situando las replicas tras un balanceador de carga. En lugar de escalar solo aquellos módulos de la aplicación que requieren de más recursos, se escala la aplicación en su totalidad. En contraposición, una arquitectura basada en microservicios establece unos límites claros entre los módulos de la aplicación, haciendo que se puedan desarrollar, desplegar, escalar y administrar de forma independiente unos de otros.

2. Características de una arquitectura basada en microservicios

2.1. Componentes como servicios

Un componente es una unidad de software que se puede reemplazar y actualizar de forma independiente. Las librerías son componentes que están ligados a un programa y son invocadas a través de llamadas a funciones. En cambio, los servicios son componentes que se ejecutan como procesos

externos y con los que se puede comunicar a través de mecanismos como llamadas a procedimientos remotos (RPC) o peticiones HTTP.

Uno de los motivos por los que se recomienda emplear servicios frente a librerías es que los servicios se pueden desplegar de forma independiente, lo que significa que un cambio en el servicio solo requiere que el servicio se vuelva a desplegar. Solo algunos cambios en la interfaz o contrato del servicio requerirán un cambio en los consumidores. Emplear contratos para consumir un servicio hace que la interfaz del componente sea más explícita.

Sin embargo, las llamadas remotas son más costosas que las invocaciones dentro del mismo proceso, por lo que la interfaz del servicio debe definirse de tal forma que sus consumidores no tengan que comunicarse con él continuamente (chatty conversations).

2.2. Organización alrededor de las capacidades del negocio

En las aplicaciones monolíticas, normalmente los equipos de trabajo que se forman están asociados la interfaz de usuario, la base de datos y la parte servidora. Cualquier cambio simple involucra la coordinación entre diferentes equipos. De acuerdo a la ley de Conway, una organización que diseña un sistema producirá un diseño para la aplicación que será una copia de su estructura organizacional.

La organización en los microservicios se realizada alrededor de las capacidades de negocio. Los equipos que se encargan de cada servicio son multifuncionales y se pueden autogestionar. En cada equipo están presentes, entre la totalidad de sus miembros, todas las habilidades necesarias para el desarrollo completo del servicio.

2.3. Productos en lugar de proyectos

En muchas organizaciones, el desarrollo de sistemas se hace a través de proyectos: piezas de software que una vez desarrolladas se consideran completadas.

En una aproximación basada en microservicios, cada equipo es responsable del ciclo de vida completo de un producto. Esto sigue la filosofía de Amazon: ɔou build, you run it", que aproxima a desarralloradores y clientes.

2.4. Servicios inteligentes y protocolos de comunicación tontos

La lógica asociada a la capacidad de negocio se debe encontrar en lo máximo posible dentro del servicio. Existen mecanismos de comunicación como el .Enterprise Service Bus (EBS)"que permite implementar mucha lógica dentro de la infraestructura de comunicación. Los servicios han de estar desacoplados y para ello se emplean protocolos simples como REST a través de patrones de tipo coreografía, donde cada servicio conoce cómo debe responder a cada petición, o estándares asíncronos de mensajería AMQP.

2.5. Gobierno descentralizado

Emplear una única tecnología específica para el desarrollo completo de una aplicación puede ser restrictivo: no todos los problemas se pueden solucionar de la misma forma. Los microservicios que compongan una aplicación pueden ser desarrollados con diferentes tecnologías, pero que se pueda hacer no significa que siempre se deba hacer. Netflix, por ejemplo, construye herramientas que puedan emplear otros desarrolladores que se enfrentan a problemas similares, aunque empleen otra pila tecnológica.

2.6. Gobierno descentralizado en los datos

Mientras que las aplicaciones monolíticas prefieren una única base de datos para persistir sus datos, los microservicios apuestan por que cada servicio tenga su propia base de datos, que puede implementarse en tecnologías de bases de datos distintas (lo que se conoce como el patrón políglota).

Sin embargo, la gestión descentralizada supone varios inconvientes. En primer lugar, no se pueden realizar transacciones como se realiza en la base de datos monolítica porque puede haber más de una base de datos implicada en la actualización. Esto se traduce a que la consistencia pasa a ser eventual y se tienen que implementar mecanismos de compensación para que, en caso de que parte de la antigua transacción falle en uno de los microservicios, se restauren los valores modificados en el resto de servicios. En segundo lugar, los servicios se basan en las capacidades de negocio, pero una misma entidad puede estar modelada en más de un microservicio desde perspectivas distintas. La integridad referencial no se puede realizar a través de claves ajenas porque existe más de una base de datos implicada, lo que significa que se tienen que implementar mecanismos para asegurar la integridad de los datos.

2.7. Automatización de la infraestructura

Muchos sistemas compuestos por microservicios emplean las prácticas de entrega continua e integración continua. Con estas prácticas se garantiza que los cambios a los clientes llegan antes. Como el despliegue está automatizado, para garantizar que el cambio que se va a desplegar es correcto se ejecutan una suite de pruebas automatizadas. También gracias a la automatización, la puesta en producción del sistema es más familiar y no es una fase que suponga riesgos dentro del proceso de desarrollo.

2.8. Diseñados para el fallo

El uso de servicios implica que cualquier invocación puede fallar, ya que se realiza a través de la red. Netflix ha desarrollado una librería para inducir fallos en servicios y centros de datos en producción para probar la resilencia y monitorización de sus servicios. Es importante detectar los fallos rápido y automatizar el proceso de restauración del servicio.

2.9. Diseño evolutivo

El diseño modular de los servicios hace que puedan ser fácilmente reemplazados y actualizados. Las cambios que se deben realizar son muchas veces localizados en un único servicio, por lo que solo se debe desplegar este y no toda la aplicación.

En el momento en que nos demos cuenta que dos servicios cambian conjuntamente de forma repetida estaremos frente a un signo de que puede ser conveniente unirlos. En caso de cambios que puedan romper a los consumidores, se puede lidiar con este problema a través del versionado de servicios, siendo tolerantes con los consumidores para que tengan disponible la vieja interfaz hasta que migren sus invocaciones.

Cabe tener en cuenta que no siempre se obtienen los límites de un servicio a la primera. El código debe ser sencillo de refactorizar y el uso de microservicios hace esta tarea más complicada que si emplearamos librerías.

3. ¿Son los microservicios el futuro?

Muchas organizaciones (Amazon, Netflix, Travis CI...) están empleando microservicios como estilo arquitectónico, aunque a veces lo envuelvan

dentro del concepto de las arquitecturas SOA.

Los microservicios son la dirección futura de las arquitecturas software. Sin embargo, las consecuencias de elegir un estilo arquitectónico no son evidente hasta años más tarde de haber sido tomadas, por lo que hasta que no se vean sistemas maduros que empleen microservicios no podremos afirmar a ciencia cierta que son el futuro.