9 DE ABRIL DE 2021

MICROSERVCIOS

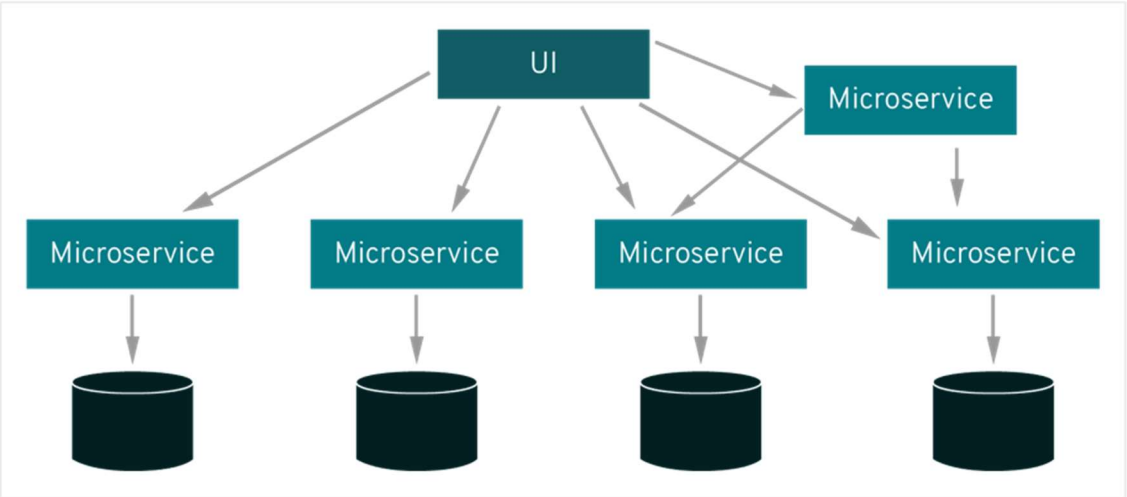
ARQUITECTURA DEL SOFTWARE

ALBERTO FERNÁNDEZ GUTIÉRREZ Y CHRISTIAN DÍAZ GONZÁLEZ

Contenido

¿Qué son los microservicios?

Los microservicios están de moda. Prometen manejar cantidades enormes de peticiones sin caerse, responder muy rápido y poder desplegar rápidamente nuevas versiones, pero ¿qué son los microservicios?



Los servicios son débilmente acoplados y su modelo de datos independiente para cada uno. Proporcionan una funcionalidad completa y se autogestionan ellos mismos. Esto es así hasta el punto de que cada microservicio puede estar implementado con un lenguaje diferente. Podríamos encontrarnos un servicio escrito en Java junto a otro escrito en Node.js o PHP, todo depende de lo que sea mejor para ese servicio en concreto. Igualmente ocurrirá con la base de datos: podríamos tener Oracle o MySQL para unos servicios o MongoDB e incluso Redis para otros, según sea más eficiente para el microservicio que se esté implementando.

Ventajas

La principal ventaja que proporciona la arquitectura de microservicios es la facilidad para escalar horizontalmente, su elasticidad. Unido a algún gestor de contenedores, es relativamente sencillo lanzar nuevas instancias de nuestros microservicios para atender grandes picos de demanda o eliminarlas para ahorrar costes cuando la demanda sea baja.

A esto se añade la capacidad para recuperarse de situaciones difíciles, su resiliencia. Al existir varias instancias de los microservicios, la caída de algunas de ellas puede superarse desplegando rápidamente nuevas instancias, incluso de forma automática.

Otra gran ventaja viene dada por el despliegue automático y el bajo nivel de acoplamiento. Permite a las aplicaciones ser más independientes y que, por tanto, puedan ser actualizadas afectando en menor medida al resto del sistema, posibilitando ciclos de entrega más cortos.

También se nos permite poder seleccionar la tecnología más conveniente para cada servicio, al permitir integrar de forma sencilla distintos lenguajes y otras piezas software, como la base de datos. Además, gracias a esta independencia, los cambios de tecnología pueden realizarse con mayor rapidez, mejorando la renovación de las herramientas.

Inconvenientes

Sin embargo, todo tiene un coste. Con la gran cantidad de microservicios e instancias a desplegar, aumenta considerablemente la complejidad del sistema y su gestión puede dar algún dolor de cabeza. Requerirá un esfuerzo extra en monitorización y gestión de errores.

A diferencia de un sistema monolítico o simplemente más acoplado, la solución no permite ser tan eficiente como en estos sistemas, lo que lleva a necesitar mayor cantidad de recursos hardware para procesar la misma carga de trabajo. Este problema puede verse compensado, en parte, por la facilidad para escalar el sistema, especialmente en entornos de pago por uso de recursos.

Una arquitectura de microservicios tiene implicaciones más allá del propio software. Para aprovechar realmente sus capacidades, requiere la adopción de tecnologías de integración y despliegue continuos, así como la especialización de los equipos de trabajo. También implicará realizar un esfuerzo para la organización del sistema en base a su funcionalidad para la división en microservicios y un extra de configuración y en la parte de sistemas. Es en definitiva un cambio de filosofía que debe calar en los equipos para ser efectivo.

A menudo se tiende a pensar que es una nueva arquitectura a la que hay que adaptarse sí o sí, pero nada más lejos de la realidad. Como en todo, antes de decidir adoptar una arquitectura de microservicios, será necesario evaluar si las ventajas superan a los inconvenientes para nuestro caso concreto y si el esfuerzo merecerá la pena. Si, por ejemplo, la escalabilidad no es un factor demasiado importante, la adopción de una arquitectura MSA podría no ser lo más indicado.

Mejor diseño

La realidad es: la estructura de tiempo de ejecución elegida no impide de ninguna manera la estructura de su código fuente. Las unidades estructurales en tiempo de ejecución normalmente son al menos tan grandes como las unidades estructurales del código fuente, generalmente más grandes. Nuestras herramientas de compilación suelen permitir asignaciones arbitrarias desde el código fuente a las unidades de tiempo de ejecución. Esto significa que puede organizar y estructurar su código fuente de forma completamente independiente de la estructura de tiempo de ejecución elegida.

Esto significa: Un monolito de despliegue se puede estructurar perfectamente.

No necesita microservicios para lograr una estructura de código fuente aceptable (es decir, mantenible).

Sin embargo, los monolitos rara vez están bien estructurados. Por lo general, encontramos diseños deteriorados llenos de dependencias, violaciones de diseño, hacks rápidos y cosas peores.

Pero, y este es el punto clave aquí, no se debe a la estructura de tiempo de ejecución elegida "monolito". Se debe a la falta de conocimiento, experiencia y disciplina de los ingenieros de software involucrados.

Cambio de tecnología más fácil

Es cierto que las unidades de tiempo de ejecución más pequeñas facilitan la reescritura de una sola unidad en una nueva tecnología. Aún así, el Principio de uniformidad del esfuerzo de migración de tecnología que discutí en esta publicación es cierto:

Se necesita el mismo esfuerzo para migrar un sistema existente a una nueva tecnología que para desarrollar el sistema hasta su estado actual.

Si bien puede ser bastante fácil migrar un solo microservicio a una nueva tecnología, el esfuerzo de migrar una aplicación completa que se compone de muchos microservicios será tan costoso como migrar un monolito correspondiente, que en ambos casos suele ser mucho más costoso que esperado.

La ventaja es que puede estirar mejor la migración, ya que puede migrar la aplicación servicio por servicio. De hecho, esta es una gran ventaja, ya que no tiene un tiempo de inactividad prolongado al migrar toda la aplicación a la vez. Puede equilibrar mucho mejor la migración y la evolución funcional continua requerida de la aplicación. De esta manera, la descomposición de la aplicación en servicios apoya la gestión de riesgos en este contexto muy bien.

Moverse rápido

Esa es probablemente la razón principal por la que los hiperescaladores adoptaron microservicios. Viven en mercados muy rápidos en los que el ganador se lo lleva todo. Esto significa que necesitan moverse rápido para mantenerse en la cima, es decir, adaptar de manera continua y rápida su oferta de servicios (negocios basados en TI) a las necesidades y demandas en constante cambio de sus usuarios.

Para poder moverse con rapidez, se organizaron en equipos de capacidad orientados al mercado. Esto permitió a los equipos moverse a su propia velocidad porque, si se dividen correctamente, las capacidades de cara al mercado son (en su mayoría) independientes entre sí. Como resultado, se minimiza la coordinación entre equipos, que normalmente ralentiza mucho a las empresas.

Si todos los equipos necesitan moverse al unísono debido a necesidades de coordinación, en un momento dado el equipo más lento define la velocidad. Sin la necesidad de coordinación cruzada, cada equipo puede moverse a su propia velocidad óptima.

Los hiperescaladores también sabían (o aprendieron rápidamente) que los microservicios imponen muchos desafíos con respecto a todos los aspectos de la TI, que los microservicios son un estilo arquitectónico muy exigente, probablemente uno de los estilos más exigentes que existen. Pero a medida que los ayudaron a resolver sus singulares desafíos comerciales, estuvieron dispuestos a pagar el precio.

Si transfiere la historia del hiperescalador a empresas normales, significa que los microservicios pueden ayudarlo a moverse realmente rápido a nivel empresarial en un mercado altamente competitivo. Aún así, también requiere que configure su organización y procesos de una manera que permita una actuación rápida e independiente de los equipos.

Y significa que debe implementar prácticas muy rigurosas con respecto al diseño, desarrollo, pruebas, implementación y operaciones que generalmente no necesita con otros estilos arquitectónicos, al menos en ese grado. Y finalmente debes estar dispuesto a invertir mucho en la formación de tu gente (dentro y fuera de TI).

Si no está dispuesto a hacer todo esto, los microservicios no lo ayudarán a ser más rápido. Simplemente harán las cosas más complicadas y probablemente resultarán en un panorama de TI más frágil que antes.

Requisitos no funcionales muy dispares

La segunda razón, mucho más rara, para optar por microservicios son los NFR muy dispares. Suponga que parte de su aplicación es utilizada por muchos miles de usuarios al mismo tiempo que esperan una alta disponibilidad, incluidos tiempos de respuesta muy cortos, incluso en las horas pico (o especialmente en ese momento).

Otra parte de su aplicación es utilizada solo por un puñado de usuarios, pero debido a la sensibilidad de sus actividades, esa parte tiene que cumplir con demandas de privacidad extremadamente altas: que nadie, excepto un puñado de personas, pueda acceder a los datos como puede o es. capaz de desencadenar sus acciones.

Sería lamentable que tuviera que implementar estas demandas no funcionales tan dispares para todas las partes de la aplicación por igual. Pero eso es lo que básicamente necesitas hacer si implementas todo en una sola aplicación. Básicamente, debe crear la unión de todos los NFR y asegurarse de que la aplicación resultante los cumpla todos en tiempo de ejecución, sin importar si solo un pequeño subconjunto de la aplicación los necesita.

Volviendo al pequeño ejemplo:

• La parte sensible no tiene demandas de escalabilidad y demandas de tiempo de respuesta muy moderadas.

• La parte pública tiene demandas de seguridad muy moderadas.

Sin embargo, enviar ambas partes en una sola aplicación significaría que la aplicación debe ser capaz de cumplir con la unión de las demandas. Esto hace que la aplicación sea mucho más complicada de lo que debería ser si las partes con NFR dispares pudieran aislarse.

Eso es lo que los microservicios pueden brindarle: la capacidad de dividir su aplicación en múltiples partes, cada una de las cuales implementa NFR muy diferentes.

Cuando usar microservicios

Si necesita tiempos de ciclo muy cortos, los microservicios pueden ser una opción sensata para usted. Aún así, es una cuestión de tamaño de la organización y escalamiento anticipado. Si eres un solo equipo, los microservicios no te dan ninguna ventaja.

Si necesita ir rápido y tener varios equipos (o al menos espera crecer mucho en el futuro), los microservicios pueden ser una opción sensata para usted. Aún así, debe estar dispuesto a pagar el precio por ellos.

Si no está dispuesto o no puede hacer todas las demás cosas (gobierno diferente, organización diferente, diseño diferente, disciplina de alto desarrollo, automatización rigurosa, implementabilidad totalmente independiente, ...), prefiere no optar por microservicios porque enfrentaría los desafíos de microservicios sin poder aprovechar sus beneficios. En un entorno así, es mejor encontrar un estilo arquitectónico más simple que se adapte a sus necesidades.

Finalmente, si usted es uno de los pocos hiperescaladores o tiene algún caso de uso específico con varios millones de solicitudes simultáneas por segundo y requisitos de disponibilidad muy altos, los microservicios pueden ser su estilo arquitectónico. Nuevamente, tenga en cuenta el precio que debe pagar, ya que solo podrá aprovechar los beneficios de los microservicios si paga el precio.

Alternativas a los microservicios

Los dos estilos son:

• Moduliths

• Microlitos

Moduliths:

Si usted

• no es necesario moverse rápido y / o no tener varios equipos (y la escala exponencial no está a la vista)

• y no tienen requisitos especiales de tiempo de ejecución en términos de NFR muy dispares,

un monolito de implementación es probablemente la arquitectura en tiempo de ejecución más simple posible.

ACTUALIZACIONES DE TECNOLOGÍA CON MÓDULITOS

Todavía tenemos el tema de las actualizaciones tecnológicas. Incluso si no se necesitan con frecuencia, pueden volverse difíciles para un monolito de implementación realmente grande, pero solo si el diseño funcional es malo.

Si su diseño interno está impulsado por la reutilización, normalmente terminará con muchos módulos (re) utilizándose entre sí. Si lo hace bien, tiene una estructura de módulo en capas internas donde cada módulo de una capa superior se basa en las funcionalidades proporcionadas por la capa de abajo. 2

Si su nueva tecnología puede llamar a la vieja tecnología en un entorno de este tipo, puede reemplazar la aplicación de arriba hacia abajo. Si la vieja tecnología puede llamar a la nueva tecnología, puede reemplazarla de abajo hacia arriba. Si ambas direcciones funcionan, usted es libre de decidir.

El punto clave es que las actualizaciones de tecnología no se vuelven difíciles debido a la estructura de tiempo de ejecución monolítica, sino a las deficiencias del diseño funcional interno y la implementación.

Hablando visualmente, en lugar de capas tienes pilares. En tal diseño, puede reemplazar el sistema módulo por módulo porque los módulos son independientes entre sí. 3

COORDINACIÓN DE EQUIPO

Otra posible preocupación con respecto a los módulos es la coordinación del equipo. ¿Un módulo no requiere mucha más coordinación de equipo que los microservicios?

Lo mismo es básicamente cierto para los moduliths. Sin una configuración organizativa adecuada y una arquitectura de módulo funcional, todas las expectativas de autonomía son vanas.

MICROLITHS:

Si no necesita moverse rápido y / o no tiene varios equipos pero, a diferencia de la configuración de modulith, tiene requisitos de tiempo de ejecución especiales en términos de NFR muy dispares, los microlitos pueden ser una opción sensata.

También pueden ser una opción sensata si se cumplen las condiciones previas para usar microservicios, pero no puede o no está dispuesto a pagar el precio de los microservicios para crear un panorama de servicios completamente distribuido.

Un microlito es básicamente un servicio que se diseña utilizando los principios de diseño de módulos independientes que evita llamadas entre módulos / servicios mientras se procesa una solicitud externa. Por lo general, esto requiere algún mecanismo de conciliación de estado y datos entre servicios para propagar cambios que afectan a múltiples microlitos. El mecanismo de conciliación debe estar temporalmente desacoplado del procesamiento de solicitudes externas. 1

Habiendo separado el procesamiento de la solicitud externa y la comunicación entre microlitos, puede dividir su aplicación monolítica en varias partes más pequeñas sin verse afectado por las imponderabilidades de los sistemas distribuidos con un golpe.

La compensación es que debe pagar el precio de la reconciliación temporalmente desacoplada entre los microlitos. Por otro lado, no necesita lidiar con los problemas que surgen si falla una llamada a otro servicio mientras se procesa una solicitud externa.

Esto también requiere un diseño de independencia mutua, ya que no se permite que los microlitos se llamen entre sí mientras procesan una solicitud externa que puede resultar desconocida para los ingenieros que están acostumbrados a pensar en capas, reutilización y técnicas de división y conquista basadas en pila de llamadas:

• Toda la funcionalidad necesaria para un caso de uso (o al menos una interacción del usuario) debe implementarse dentro de un solo microlito.

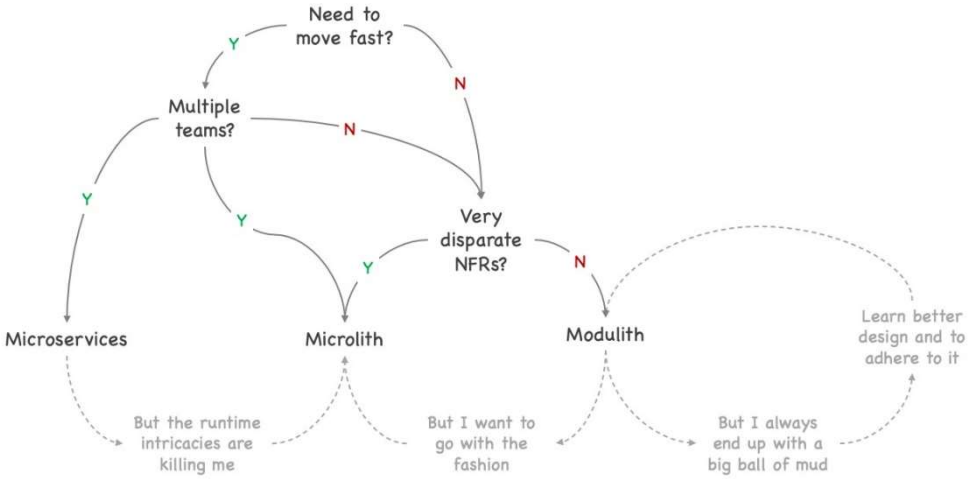
• Todos los datos necesarios para atender una solicitud externa deben estar en su base de datos, si es necesario, replicados desde la base de datos de otro microlito a través del mecanismo de conciliación de datos.

• La funcionalidad reutilizable debe proporcionarse a través de bibliotecas o alguna otra implementación o mecanismo de modularización del tiempo de construcción, no a través de otros servicios. 2

Si bien podría argumentar que los microlitos son microservicios con solo algunas restricciones adicionales aplicadas, estas restricciones son exactamente las que marcan la diferencia.

Entonces, las restricciones son cruciales en el diseño arquitectónico. Diferentes restricciones significan un estilo arquitectónico diferente con diferentes propiedades.

Los microlitos además facilitan los cambios tecnológicos que los modulitos porque son más pequeños.


Need to move fast?
Y
N
Multiple teams?
N
Y
Y
Y
Very disparate NFRs?
N
Microservices
Microlith
Modulith
Learn better design and to adhere to it
But the runtime intricacies are killing me
But I want to go with the fashion
But I always end up with a big ball of mud