Antipatrones y trampas en microservicios

Víctor Iranzo

6 de julio de 2018

1. Antipatrón: migración dirigida por los datos

Cada microservicio es dueño de sus propios datos. Este antipatrón ocurre cuando se migra de una aplicación monolítica a una basada en microservicios y parece una buena idea migrar conjuntamente la funcionalidad del servicio y sus datos.

Es complicado acertar la granularidad de los servicios a la primera, por lo que es mejor comenzar con una división de grano grueso. Realizar dos migraciones simultáneamente requiere un sobreesfuerzo y las migraciones de datos son una fuente de error mayor que las de funcionalidad.

Como solución es conveniente migrar primero solo la funcionalidad del servicio sin preocuparse en un primer momento por los límites del servicio. Una vez se verifique que el nivel de granularidad es correcto y se observe cómo el servicio es usado por sus consumidores, se deben migrar los datos del servicio a una base de datos independiente.

2. Antipatrón: timeouts

La disponibilidad del servicio es la habilidad de un consumidor para conectar al servicio y enviarle peticiones. La capacidad de respuesta del servicio es el tiempo que tarda el servicio en responder a una petición una vez se inicia la comunicación. Si el servicio no está disponible, el consumidor puede o devolver un error rápidamente o repetir la petición pasado un intervalo. En cambio, si el servicio no responde, el consumidor puede esperar indefinidamente o esparar hasta alcanzar un timeout.

El uso de timeotus puede ser una mala idea. No se puede poner un límite demasiado bajo porque trataría como errores respuestas que han tardado

más de lo debido. Tampoco puede ser muy alto, o se tardaría mucho en devolver un error, sobretodo cuando se encadenan diferentes llamadas.

Una solución pasa por usar un patrón de cortocircuitos. Los cortocircuitos monitorizan continuamente el estado de un servicio. Si detectan que un servicio no responde cierto número de peticiones, el cortocircuito pasa a un estado en el que rechaza cualquier petición al servicio para que estas fallen rápidamente. Cuando el servicio vuelve a estar operativo, el cortocircuito comienza a aceptar de nuevo peticiones. Una implementación de este patrón es la librería Hystrix.