

# Capítulo 3: Modelar microservicios

Víctor Iranzo

19 de mayo de 2018

## 1. Alta cohesión y bajo acoplamiento

Estos dos términos son claves para implementar servicios de buena calidad:

- Alta cohesión: funcionalidades relacionadas deben agruparse juntas y separadas de otras no relacionadas. De esta manera, cuando se realice un cambio sólo se deberá modificar y desplegar un microservicio, haciendo el despliegue más seguro y coherente. Además, agrupar funcionalidades similares reducirá el número de llamadas potenciales a otros servicios.
- Bajo acoplamiento: en los microservicios poco acoplados un cambio en uno de ellos no requiere cambiar ningún otro. Esta propiedad es clave ya que de otra manera no se puede desplegar de manera independiente un servicio sin cambiar otros que lo consuman. Un servicio debe conocer lo mínimo posible de otros con los que colabore. Un síntoma para detectar que dos servicios están acoplados es que estén en continua comunicación (el término en inglés es *chatty communication*).

## 2. Contexto de un microservicio

En el libro 'Domain-Driven Design', Eric Evans explica que el dominio de una solución está compuesta de múltiples contextos bien limitados. Cada contexto está formado por modelos que no necesitan ser compartidos con otros a menos que se defina explícitamente una interfaz que los empleen. La interfaz es el punto de entrada para que otros contextos puedan comunicar con el nuestro, empleando los términos y entidades que en nuestros modelos se definan.

Esta perspectiva puede trasladarse fácilmente al modelado de microservicios. Los contextos limitados de Evans que analicemos en nuestro sistema son firmes candidatos a transformarse en servicios. Así, los límites de un servicio quedan bien limitados porque todas las entidades que pueda requerir se encuentran dentro de sus fronteras, garantizándose su alta cohesión y bajo acoplamiento.

No todos los modelos de un servicio son públicos. Por ejemplo, podemos definir una entidad en el modelo con muchos atributos pero solo hacer públicos al resto de servicios parte de ellos. También puede ocurrir que una misma entidad aparezca en modelos de distintos servicios. En este caso, los atributos de la entidad en cada uno de ellos podrían variar porque a cada uno les interesa unas propiedades u otras.

### **3. Descomposición en microservicios según las capacidades del negocio**

Cuando se razona sobre los límites de un servicio no se debe pensar en los datos que este almacena sino en las funcionalidades que ofrece. Pensar en los datos nos conduce a desarrollar únicamente servicios CRUD (en inglés, aquellos que nos permiten las operaciones de crear, leer, actualizar y eliminar datos) que ofrecen unas operaciones muy limitadas. Un servicio ofrece ciertas funcionalidades o capacidades que aportan valor al negocio.

Una descomposición temprana de un sistema en microservicios puede conllevar ciertos riesgos. Si el equipo a cargo del desarrollo tiene pocos conocimientos del dominio del problema a resolver, puede ser buena idea comenzar la implementación como si de un sistema monolítico se tratara. Una vez sean conocidos los límites de cada servicio, se puede refactorizar el código implementado. En este momento, se puede comenzar dividiendo la solución en grandes servicios que poco a poco se vayan dividiendo en más pequeños conforme se estudien las ventajas de hacer cada nueva extracción.

Cuando sea necesario realizar un cambio por nuevos requisitos del negocio, estos se localizarán en un contexto bien delimitado porque existirá una correspondencia entre la estructura de la organización y los microservicios del sistema. Como consecuencia, el tiempo medio para realizar un cambio se verá reducido porque solo hará falta volver a desplegar una porción del sistema.

Además, la comunicación entre microservicios se asemejará a la existente entre las entidades del negocio. Los objetos empleados para dicha comuni-

cación serán los detallados en los modelos compartidos, que son usados en la interfaz de uno de los interlocutores.

## **4. División vertical y horizontal de un servicio**

Cuando un servicio crece demasiado por incluir más funcionalidades, es el momento de dividirlo en otros más pequeños. Las funcionalidades que ofrecía el servicio se reparten entre los nuevos atendiendo a las capacidades del negocio. La división que se debe realizar es en vertical ya que todas las capas del servicio quedan partidas. Cada uno de los nuevos servicios tendrá todas las capas del original pero solo con el código necesario para ofrecer sus funcionalidades.

El otro tipo de división de un servicio es la horizontal. Cuando un servicio es muy consumido puede sufrir problemas de rendimiento. Una posible solución es dividir el sistema de forma horizontal. La división horizontal consiste en añadir una capa intermedia entre dos para reducir el número de accesos que la primera capa hace sobre la segunda. La capa añadida implementa una caché para atender las peticiones de la capa sobre la que está sin recurrir en todo momento a la capa sobre la que está. Otra posible solución es replicar el servicio de forma geográfica u organizacional atendiendo sus consumidores.

En resumen, a la hora de diseñar microservicios es la primera división la que se debe seguir y únicamente se debe aplicar una división en horizontal por motivos de rendimiento.