ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS
INGENIERÍA EN SOFTWARE

**Aplicaciones Web Avanzadas**

**Nombre:** Ricardo Falcón

**Fecha:** 07/03/2022

## Arquitectura Orientada a Servicios vs Microservicios

Una encuesta de 2018 [1] encontró que el 63 % de las empresas estaban adoptando arquitecturas de microservicios. Esta adopción generalizada está impulsada por la promesa de mejoras en resiliencia, escalabilidad, tiempo de comercialización y mantenimiento, entre otras razones.

En una aplicación orientada a servicios (SOA), todos los objetos de datos y acciones son manejados por una única base de código muy unida. Los datos generalmente se almacenan en una sola base de datos o sistema de archivos. Se desarrollan funciones y métodos para acceder a los datos directamente desde este mecanismo de almacenamiento, y toda la lógica comercial está contenida dentro del código base del servidor y la aplicación del cliente.

SOA surgió como una alternativa a las aplicaciones monolíticas a principios de la década de 2000. Esencialmente, esta arquitectura de software también implica la implementación de servicios: una aplicación que maneja un determinado proceso y brinda ese servicio a otros componentes a través de un protocolo de comunicación [2]. Las aplicaciones orientadas a servicios suelen incluir una estricta jerarquía de servicios. En primer lugar, los servicios de nivel superior que controlan el flujo general de actividades e invocan uno o más servicios comerciales, según la acción de los usuarios. El segundo nivel consta de servicios que cumplen una tarea comercial específica, por ejemplo, validar la dirección de correo electrónico de un usuario. El tercer nivel incluye servicios de acceso a datos; cada uno de ellos maneja tareas relacionadas con la lectura/escritura de datos en áreas de almacenamiento de datos. Aunque tanto la arquitectura SOA como la de microservicios asumen el uso de servicios débilmente acoplados, existen algunas diferencias importantes en términos de cómo funcionan estas arquitecturas y qué objetivos logran.

Es posible migrar varias aplicaciones y/o plataformas monolíticas, cada una con sus propios mecanismos de almacenamiento de datos, interfaces de usuario y esquema de datos, a un conjunto unificado de microservicios que realizan las mismas funciones que las aplicaciones originales en una única interfaz de usuario. Migrar estas aplicaciones a microservicios ofrece las siguientes ventajas:

- Eliminar la duplicación de esfuerzos para la entrada manual.
- Reducir los riesgos de desarrollo programático.
- Proporcionar una vista única y unificada de los datos.
- Mejorar el control y la sincronización de estos sistemas

Algunas aplicaciones actuales se implementan como una arquitecta orientada a servicios, en los que todo el almacenamiento y el procesamiento de datos están controlados por el monolito, y todas las funciones para todos los objetos de datos se procesan utilizando la misma base de código de back-end. Aunque este método de desarrollo es históricamente típico, puede conducir a problemas de mantenibilidad:

- La actualización del código para un conjunto de objetos de datos puede romper las dependencias en otras áreas.
- Escalar el sistema puede ser difícil debido a las relaciones de datos entrelazados.
- Proporcionar múltiples formas de ubicar, actualizar y acceder a los datos mediante programación puede ser un desafío.

El patrón de desarrollo de software de microservicios proporciona un método para construir un único servicio unificado a partir de una colección de servicios más pequeños. Cada componente de microservicio se centra en un conjunto de acciones poco acopladas en un conjunto de datos pequeño y bien definido. Cada microservicio incluye un sistema de almacenamiento independiente para que todos sus datos estén ubicados en una única ubicación [3]. Al reducir la cantidad de operaciones de bloqueo en el back-end de almacenamiento de datos, esta arquitectura permite que el microservicio se escale horizontalmente, aumentando la capacidad al agregar más nodos al sistema en lugar de simplemente aumentar los recursos (RAM, CPU, almacenamiento) en un solo nodo.

Los microservicios se pueden escribir en cualquier lenguaje de programación y pueden usar cualquier entorno de base de datos, hardware y software que tenga más sentido para la organización. Una interfaz de programación de aplicaciones (API) proporciona el único medio para que los usuarios y otros servicios accedan a los datos del microservicio [4].

La API no necesita estar en ningún formato en particular, pero la transferencia de estado representacional (REST) es popular, en parte porque su legibilidad humana y su naturaleza sin estado la hacen útil para las interfaces web.

Para ejemplificar lo dicho anteriormente, se toma un caso de estudio de la implementación de microservicios. A2B.Direct, una innovadora plataforma de transporte de carga B2B, la cual optó por una arquitectura de microservicios para lograr una mayor flexibilidad para el próximo aumento en el uso. Al combinar este tipo de arquitectura con el uso extensivo de tecnologías de código abierto y la incorporación de las mejores prácticas actuales de DevOps, el equipo de Infopulse redujo el tiempo de comercialización y los costos 3 veces más de lo que se proyectó originalmente.

En la ***Figura 1***, la interfaz se comunica con la puerta de enlace API. Que redirige las solicitudes a los servicios necesarios. Cada servicio que requiere almacenamiento persistente tiene una base de datos separada. Los servicios se comunican entre sí mediante un bus de mensajes. Dentro del sistema, se implementa el registro detallado y el seguimiento de consultas. Estos datos se analizan en busca de anomalías, lo que permite responder a las fallas de manera oportuna. Esto se diseñó con el fin de obtener una mayor estabilidad, una de las ventajas de utilizar microservicios.

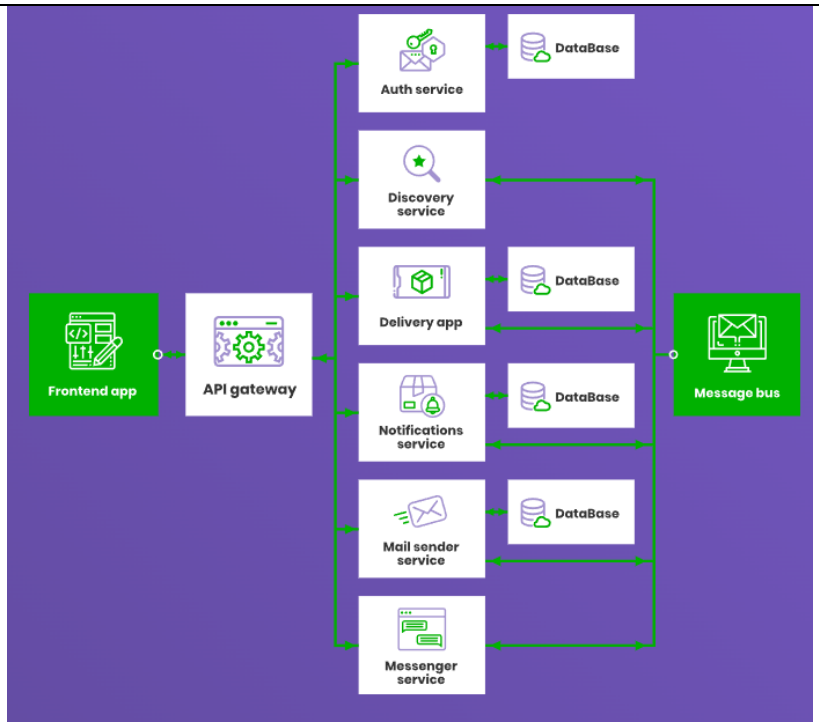


**Figura 1.** Arquitectura de microservicios diseñada por A2B Direct

A2B Direct mejoró su escalabilidad, puesto que cada servicio se ejecuta como un componente independiente, lo que significa que puede escalar una sola función a pedido, sin escalar toda la aplicación. Su producto se vuelve más ágil y puede manejar los patrones de uso cambiantes mientras le ofrece un costo total de propiedad reducido, especialmente si está utilizando una arquitectura de microservicios en la nube. Del mismo modo, puede optimizar el uso y el rendimiento del servidor local mediante la implementación de servicios críticos para el negocio en varios servidores dedicados para delimitar el uso de recursos de diferentes componentes de software.

A2B Direct facilitó la entrega e implementación continua puesto que, al utilizar microservicios, donde el desarrollo y las pruebas se fusionan en un proceso unificado. Con este enfoque, crea un nuevo entorno en el que el código se desarrolla, depura e implementa continuamente en un entorno similar al de producción mejorando en:

- 21% de aumento en el nuevo software y servicios entregados;
- 22% mejoró la calidad de las aplicaciones implementadas;
- aumento del 19% en los ingresos;
- 50% menos fallas.

## Referencias

[1] T. Smith, “Microservices Zone,” *New Research Shows 63% of Enterprises Are Adopting Microservices Architectures*, 2018. .

[2] T. Salah, M. J. Zemerly, C. Y. Yeun, M. Al-Qutayri, and Y. Al-Hammadi, “The evolution of distributed systems towards microservices architecture,” *2016 11th Int. Conf. Internet Technol. Secur. Trans. ICITST 2016*, pp. 318–325, 2017, doi: 10.1109/ICITST.2016.7856721.

[3] C. Pahl and P. Jamshidi, “CLOSER 2020 - Proceedings of the 10th International Conference on Cloud Computing and Services Science,” *CLOSER 2020 - Proc. 10th Int. Conf. Cloud Comput. Serv. Sci.*, vol. 1, no. Closer, pp. 137–146, 2020.

[4] C. M. Aderaldo, N. C. Mendonça, C. Pahl, and P. Jamshidi, “Benchmark Requirements for Microservices Architecture Research,” *Proc. - 2017 IEEE/ACM 1st Int. Work. Establ. Community-Wide Infrastruct. Archit. Softw. Eng. ECASE 2017*, pp. 8–13, 2017, doi: 10.1109/ECASE.2017.4.