# ARQUITECTURA DE MICROSERVICIOS

*Atilio Rúveda, Lucas Confalonieri, Nicolas Costantini*

*Universidad de la Cuenca del Plata*

*Paradigmas y Lenguajes de Programación III*

*Profesor Jose A. Fernandez*

# INTRODUCCIÓN

La arquitectura de software es un concepto que surge hace muchos años y se enfoca en la planificación basada en modelos y patrones de programación, así como en abstracciones teóricas, que busca llevar a cabo una pieza o la totalidad de un software de cierta complejidad.

A lo largo de este informe estudiaremos uno de los tipos más relevantes de arquitectura de software, llamada arquitectura basada en microservicios, donde expondremos qué es, cómo surge y sus ventajas y desventajas para el estudio de dicho estilo de arquitectura.

# ¿CÓMO NACEN LOS MICROSERVICIOS?

Tradicionalmente, la arquitectura y el desarrollo de software estaban orientados en un modelo llamado monolítico, el cual se basaba en tener un solo código referido a una aplicación en el cual todos los equipos de desarrolladores trabajan en paralelo.

Para aplicaciones sencillas este modelo de desarrollo no presentaba ningún problema mayor, pero a la hora de trabajar sobre un código muy grande se presentaban unas cuantas dificultades. Algunas de ellas son la gran dificultad que presenta esta arquitectura para escalar y evolucionar a lo largo del tiempo, debido a que obliga a todas las partes involucradas en el desarrollo a ponerse de acuerdo para realizar cualquier cambio a la estructura del código. Ya sea para leer, modificar o siquiera probar cualquier parte del código, la eficiencia se ve extremadamente reducida al tratarse de un único bloque de desarrollo.

En vista de los problemas que esta arquitectura tenía y teniendo en cuenta la rápida evolución del mercado global y sus necesidades, en los años 90 surge la arquitectura SOA(arquitectura orientada a servicios), la cual es en primera instancia una aproximación a los microservicios. En SOA, los componentes de una aplicación se encuentran relativamente separados entre sí, y se comunican y sirven a otros componentes mediante un protocolo de comunicaciones en red.

Si bien SOA significó un gran avance y evolución para el desarrollo de aplicaciones, todavía arrastraba ciertos problemas de eficiencia de su predecesor, tales como que los servicios o componentes no eran totalmente independientes uno del otro, por lo cual los equipos de desarrollo seguían en la obligación de coordinar cada acción que se hacía sobre un determinado componente.

Finalmente, en 2014, Martin Fowler definió el concepto de MSA (arquitectura basada en microservicios), la cual llegaba para darle una vuelta distinta a SOA y darle un enfoque técnico para desarrollar aplicaciones como un conjunto de servicios pequeños, los cuales trabajarían de manera totalmente independiente y que se ejecutan de manera autónoma en su propio proceso, comunicándose además con otros servicios a través de APIs.

## ¿QUÉ ES LA ARQUITECTURA BASADA EN MICROSERVICIOS?

Ahora que hemos descubierto cómo la tecnología y los paradigmas avanzaron según la necesidad y la demanda global de las empresas, estamos listos para definir y empezar a hablar de características específicas acerca de los microservicios.

La arquitectura basada en microservicios es un método de desarrollo de aplicaciones que toma cada componente de un código de programación como pequeños y distintos servicios que se ejecutan de manera independiente y autónoma. Cada uno de los microservicios presentes en el proyecto pueden estar en un lenguaje de programación distinto y desempeñan una función específica que aporta a toda la funcionalidad de la aplicación. Dichos servicios se comunican entre sí a través de APIs e incluso tienen sistemas de almacenamiento propios lo que evita problemas como la sobrecarga o la caída de la aplicación.

- Los componentes principales de una Arquitectura de Microservicios son:
- Servidor de configuración central: se encarga de centralizar y proveer remotamente la configuración a cada microservicio.
- Servicio de registro / descubrimiento: es el encargado de proveer los endpoints de los servicios para su consumo.
- Balanceo de carga (Load balancer) permite el balanceo entre distintas instancias de forma transparente a la hora de consumir un servicio.
- Tolerancia a fallos (Circuit breaker): permite que cuando se produzca un fallo este no se propague en cascada por todo el pipe de llamadas, y poder gestionar el error de forma controlada a nivel local del servicio donde se produjo.
- Servidor perimetral / exposición de servicios (Edge server): gateway en el que se expondrán los servicios a consumir.

- Centralización de logs: Se hace necesario un mecanismo para centralizar la gestión de logs. Pues sería inviable la consulta de cada log individual de cada uno de los microservicios. Adicionalmente, también son interesantes los dos siguientes componentes.
- Servidor de Autorización: Para implementar la capa de seguridad (recomendable en la capa de servicios API).
- Monitorización: mecanismos y algún dashboard para monitorizar aspectos de los nodos como, salud, carga de trabajo.

## VENTAJAS Y DESVENTAJAS DE LA ARQUITECTURA DE MICROSERVICIOS

**VENTAJAS:**

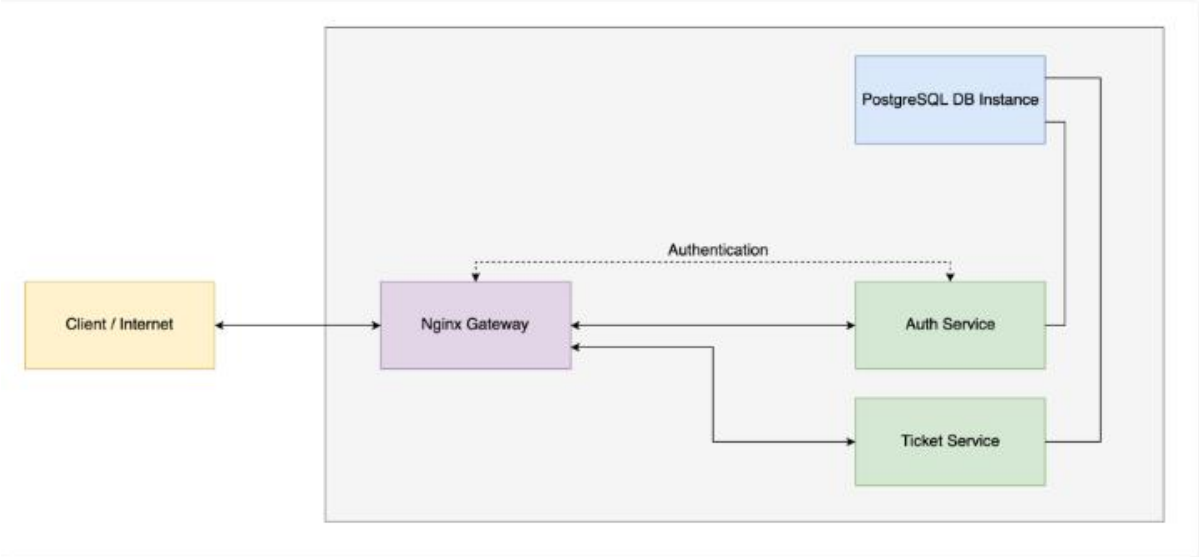
- Versátil — los microservicios permiten el uso de diferentes tecnologías y lenguajes.
- Fácil de integrar y escalar con aplicaciones de terceros.
- Los microservicios pueden desplegarse según sea necesario, por lo que funcionan bien dentro de metodologías ágiles.
- Las soluciones desarrolladas con arquitectura de microservicio permiten la mejora rápida y continua de cada funcionalidad.
- El mantenimiento es más simple y barato — con los microservicios se puede hacer mejoras de un módulo a la vez, dejando que el resto funcione normalmente.

**DESVENTAJAS:**

- Debido a que los componentes están distribuidos, las pruebas globales son más complicadas.
- Es necesario controlar el número de microservicios que se gestionan, ya que cuantos más microservicios existan en una solución, más difícil será gestionarlos e integrarlos.
- Los microservicios requieren desarrolladores experimentados con un nivel muy alto de experiencia.
- Se requiere un control exhaustivo de la versión.

# EJEMPLO DE MICROSERVICIOS CON NODE.JS

Para ilustrarlo, creamos una aplicación de gestión de tickets, aquí el diagrama para explicarlo:




Escenario

En el diagrama anterior, Nginx tiene dos funciones. El primero es administrar todo el tráfico y el segundo verificar la autenticación del usuario.

También contamos con servicios de autenticación y tickets, y solo tienen un modelo. El servicio de autenticación gestiona el registro y la verificación de autenticación, mientras que el servicio de entradas gestiona las entradas que pertenecen a los usuarios.

Tenemos dos puntos finales expuestos sin ninguna autenticación para registrarse e iniciar sesión. Mientras que un middleware gestiona la autenticación del servicio de autenticación, Nginx gestiona la autenticación del resto de servicios. Si llega alguna solicitud al servicio de entradas, Nginx pedirá validez del token al servicio de autenticación y, según el resultado, las rechazará o aceptará. Si se acepta la solicitud, Nginx también pasará la identificación del usuario activo al servicio de tickets con el encabezado.

Estructura de carpetas de un microservicio

```
- tests
- src
  - db
    - config
    - models
  - enums
  - exceptions
  - middlewares
  - services
  - utils
  - index.js
  - routes.js
  - server.js
```

Repositorio: https://github.com/lucasconfalonieri/Microservicios-TP2

# CONCLUSIÓN

Como conclusión, podemos decir que la invención de la arquitectura basada en microservicios como concepto fue un importantísimo avance en la programación a nivel global, que permitió cumplir e incluso elevar la demanda de las empresas y/o cualquier entidad que requiriera un software escalable, pero de alta complejidad y calidad.

Nos resulta incluso difícil entender cómo previamente a los microservicios se trabajaba a una escala tan grande siendo que existían tantas complicaciones a la hora de llevar a cabo un programa utilizando una arquitectura monolítica.

Como punto final, consideramos que el objetivo ha sido cumplido con creces y el proceso de aprendizaje aportó mucho a la visión que tenemos sobre el mundo de la programación y sobre cómo este trabajo se lleva a cabo hoy en día. De la mano de esto, también creemos que mirar al pasado y ver qué aspectos de la programación han sido ampliamente mejorados nos aporta un mayor entendimiento de cómo hemos llegado a los avances con los que contamos hoy en día y la gran relevancia que tienen los mismos.

# BIBLIOGRAFÍA

*Arquitectura de microservicios: Qué es, ventajas y desventajas*. (2022, 20 abril). Decide. Recuperado 29 de septiembre de 2022, de https://decidesoluciones.es/arquitectura-de-microservicios/

*Estilo de arquitectura de microservicios - Azure Architecture Center*. (s. f.). Microsoft Learn. Recuperado 29 de octubre de 2022, de https://learn.microsoft.com/es-es/azure/architecture/guide/architecture-styles/microservices

Martín, M. J. (2018, 9 febrero). *¿Qué son los microservicios?* Profile. Recuperado 30 de septiembre de 2022, de https://profile.es/blog/que-son-los-microservicios/

Atlassian. (s. f.). *Arquitectura de microservicios*. Recuperado 30 de septiembre de 2022, de https://www.atlassian.com/es/microservices/microservices-architecture