UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Implementación de un API Gateway para una arquitectura de microservicios

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Alejandro Carrión Sanmartín

Tutor: Patricio Letelier Torres

Curso 2020-2021

Resumen

????

Palabras clave: ?????, ???

Resum

????

Paraules clau: ????, ?????????

Abstract

????

Key words: ?????, ?????

Índice general

Índice de figuras

CAPÍTULO 1
Introducción

El uso de arquitecturas de microservicios es una práctica ya consolidada hoy en día en el mundo del desarrollo de *software*. Esta consiste en la construcción de servicios independientes, ejecutados en procesos diferentes, que se encargan de realizar funciones concretas y que trabajan de forma conjunta para lograr el objetivo u objetivos globales de la aplicación. Los beneficios que otorga este enfoque frente a la aproximación tradicional monolítica son muchos y muy variados. Algunos de ellos son:

- **Uso de diferentes tecnologías**. Cada microservicio puede estar construido con una tecnología diferente y puede utilizar distintos mecanismos de persistencia.
- **Maniobrabilidad en los despliegues**. Ante cualquier cambio no es necesario desplegar la aplicación entera, solamente los microservicios implicados.
- **Escalabilidad y mantenibilidad**. Los microservicios, y la separación funcional que otorgan, facilitan el escalado de las diferentes partes de la aplicación de manera independiente. Lo mismo sucede con el mantenimiento, pudiendo crear equipos especializados.

Se puede obtener más información acerca de los microservicios en el artículo de James Lewis y Martin Fowler titulado "*Microservices*" [1].

Por contraparte, este tipo de arquitecturas aumentan la complejidad del desarrollo en algunos aspectos concretos como pueden ser el versionado de los microservicios o la coordinación de las comunicaciones entre ellos. Es aquí donde entra este trabajo, pues está enfocado a paliar otra de sus desventajas: la exposición de múltiples puntos de entrada al *backend*. Para ello, se quiere desarrollar un nuevo componente que actúe de intermediario. La figura **??** muestra la comparativa de una arquitectura de microservicios basíca y otra que utiliza el componente mencionado.

1.1 Motivación

Este proyecto surge en el contexto de una práctica en empresa. El autor del mismo ha formado parte del equipo de *I+D+i* de la compañía *ADD Informática* durante el periodo de un año. *ADD* es una empresa española que se dedica a desarrollar y comercializar un *software* de gestión geriátrica llamado *ResiPlus* [2]. Actualmente cuenta con casi cien trabajadores, es líder del mercado nacional y se encuentra en proceso de expansión internacional.

La temática del trabajo es el desarrollo de un intermediario entre la interfaz de usuario de una aplicación y sus microservicios. Esta fue elegida debido a la estrecha relación que

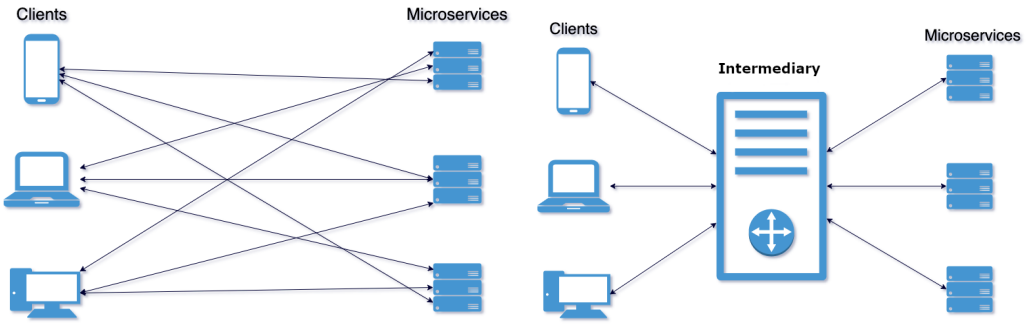

Figura 1.1: Arquitectura de microservicios básica frente a una con intermediario. Imágenes de https://www.adictosaltrabajo.com/2020/05/27/introduccion-al-api-gateway-pattern.

guarda con el desempeño del autor en las prácticas mencionadas en el párrafo anterior. Por otro lado, el desarrollo a llevar a cabo le va a otorgar una visión más global de la aplicación sobre la que se trabaja así como aumentar el nivel de conocimiento acerca de la misma con la motivación de seguir contribuyendo al proyecto por mucho tiempo más. Por último, la tecnología a utilizar, *.NET*, es de su interés y aspira así a crecer como desarrollador de ese *framework*.

En cuanto a la motivación de la empresa, esta quiere dar la posibilidad de aprovechar, en cierta medida, el trabajo realizado y a realizar por el autor del documento dentro del marco de trabajo de la entidad, viéndose esta beneficiada también.

1.2 Objetivos

Este proyecto pretende resolver un problema real. Este consiste en otorgar a una aplicación con arquitectura de microservicios la capacidad de ocultarlos. Por otro lado, también se quiere permitir la ejecución simultánea de más de una instancia de los mismos. Cabe destacar que la aplicación en cuestión se encuentra todavía en fase de desarrollo y que el componente intermediario que se va a realizar como solución es un punto clave en su arquitectura. A raíz de lo comentado se obtienen los siguientes objetivos sobre la aplicación en construcción:

- **Ocultar los microservicios** de forma que la interfaz de usuario no acceda directamente a estos por motivos de seguridad.
- Permitir la **instanciación múltiple** de los microservicios que, a su vez, tiene por objetivos:
  - Conseguir que la aplicación sea **tolerante a fallos**, gracias a la posibilidad de tener un mismo microservicio desplegado en máquinas diferentes.
  - **Aumentar la eficiencia**, al poder crear o parar instancias dinámicamente según el tráfico que reciba la aplicación.

1.3 Estructura del documento

?????

CAPÍTULO 2
Contexto tecnológico

En la actualidad existen en el mercado muchas aplicaciones y servicios que resuelven problemas parecidos al planteado. Estas herramientas se pueden dividir en dos tipos: los *API Gateway* y los *proxy* inversos. Ambos patrones comparten algunos casos de uso, por lo que sus diferencias causan confusión y no suelen quedar claras. Generalmente, se entiende que un *API Gateway* es una especialización de un *proxy* inverso, proporcionando así funcionalidades extra. Las más aceptadas e importantes son:

- Interpretan los mensajes que reciben y pueden hacer transformaciones sobre ellos; los *proxy* inversos solo los redirigen donde corresponda.
- Suelen ofrecer agregaciones de peticiones, esto es, aunar dos o más llamadas al *backend* y exponer esta composición a través de un único *endpoint*.
- Realizan tareas transversales a todos los microservicios como autenticación, autorización o monitorización.

Algunos de estos productos *software* son de pago, otros gratuitos e incluso algunos de código abierto. Se comentan los más relevantes a continuación.

2.1 *Kong*

Tal y como dice su página web, *Kong* [3] es el *API Gateway* más conocido a nivel mundial. Cuenta con las siguientes características:

- Está optimizado para arquitecturas de microservicios.
- Está concebido para ser altamente escalable.
- Ha sido creado sobre el *proxy* inverso *NGINX*, lo que hace que su eficiencia destaque.
- Es gratuito, aunque ofrece la versión *Enterprise*, que facilita la gestión del API Gateway, y la *SaaS*, llamada *Kong Cloud*.
- Posee un gran número de *plugins* desarrollados por la propia compañía y por su comunidad, que es bastante grande al ser una aplicación tan popular.
- Actualmente es utilizado por grandes empresas como *Intel*, *Ferrari* o *Yahoo!*.

2.2 *Tyk*

Tyk [4] es un *API Gateway* de código abierto.

- Permite alojar sus servicios tanto en la nube como en una infraestructura propia. También ofrece un modelo híbrido.
- Dispone de extensiones para integrarlo con diferentes mecanismos de autenticación como *OAuth* o *LDPA*.
- Necesita una base de datos *Redis*.
- Puede escalar horizontalmente y verticalmente.

2.3 *Ocelot*

Ocelot [5] es una librería para *.NET Core* que permite a una aplicación de ese *framework* actuar como API Gateway. Posee las características siguientes:

- Está pensada para arquitecturas orientadas a servicios o a microservicios.
- Al tratarse de una librería, se utiliza de forma sencilla añadiéndola como un paquete *NuGet* más.
- Su configuración es muy básica, teniendo que especificarla en un fichero *json*.
- No permite cambiar su configuración de manera dinámica.

2.4 *NGINX*

Originariamente *NGINX* [6] fue construido para ser un servidor web, como por ejemplo *Apache*, pero más tarde ofreció la posibilidad de actuar como *proxy* inverso, balanceador de carga o *proxy* para protocolos de correo electrónico. Desde la vertiente que interesa a este trabajo:

- Se trata de un *proxy* inverso ligero y de alto rendimiento
- Ofrece una versión gratuita y otra de pago, *NGINX Plus*, la cual ofrece funcionalidades extra.
- Se configura a través de un fichero el cual puede ser recargado durante su ejecución, es decir, es configurable dinámicamente.

2.5 *YARP*

YARP [7] es otra librería para *.NET* hecha por el propio *Microsoft*. En este caso, facilita la creación de *proxies* inversos. Sus aspectos más destacables son:

- Se encuentra todavía en desarrollo y la única versión lanzada hasta la fecha es una *preview*. Se prevé que vayan saliendo a la luz más versiones con más funcionalidades basadas en la experiencia de la propia empresa pero también en las opiniones de la gente.

- Fue diseñada para construir *proxies* inversos dentro de la propia infraestructura de *Microsoft* ya que en muchas áreas de la compañía necesitaban algo así.
- A raíz de la heterogeneidad de casos de uso que debe cubrir derivado del punto anterior, está pensada para ser muy personalizable y flexible.
- Permite cambiar su configuración de forma dinámica.

CAPÍTULO 3
Análisis del problema

?????

3.1 Especificación de requisitos

?????

3.2 Soluciones posibles

?????

3.3 Solución propuesta

?????

3.4 Plan de trabajo y metodología

?????

CAPÍTULO 4
Diseño de la solución

?????

4.1 Arquitectura del sistema

?????

4.2 Diseño detallado

?????

4.3 Tecnología utilizada

?????

4.4 Mantenimiento y gestión de versiones

?????

CAPÍTULO 5
Desarrollo de la solución

?????

5.1 Prototipo microservicio básico

?????

5.2 Prototipo microservicio autogenerado

?????

5.3 Comparativa de tiempos

?????

CAPÍTULO 6
Pruebas

?????

CAPÍTULO 7
Conclusiones

?????

Bibliografía

[1] Artículo de J. Lewis y M. Fowler acerca de los microservicios. Disponible en: https://martinfowler.com/articles/microservices.html

[2] Página web de *ADD Informática* y su producto *ResiPlus*: https://addinformatica.com

[3] Documentación oficial de *Kong*: https://konghq.com/kong

[4] Documentación oficial de *Tyk*: https://tyk.io

[5] Documentación oficial de *Ocelot*: https://threemammals.com/ocelot

[6] Documentación oficial de *NGINX*: https://www.nginx.com/

[7] Documentación oficial de *YARP*: https://microsoft.github.io/reverse-proxy