UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Desarrollo de software basado en microservicios: un caso de estudio para evaluar sus ventajas e inconvenientes

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Víctor Alberto Iranzo Jiménez

Tutor: Patricio Orlando Letelier Torres

Curso 2017-2018

Resum

????

Paraules clau: Microservices, Arquitecturas de software, ??????????????

Resumen

????

Palabras clave: Microservicios, Arquitecturas de software, ??????????????

Abstract

????

Key words: Microservices, Software Architecture, ?????????????

Índice general

Índice de figuras

Índice de tablas

CAPÍTULO 1
Introducción

1.1 Motivación

En la actualidad, no es necesario un alto grado de conocimientos en ingeniería del software para desarrollar una aplicación o sistema. Personas que no tienen estudios relacionados con la informática pueden producir código que, sin ser limpio y elegante, funciona. Desarrollar sistemas de calidad requiere de grandes conocimientos, pero minimiza los costes y aumenta la productividad de una organización. Se debe poner el foco en emplear una arquitectura de software que se adapte a nuestras necesidades. De lo contrario, el futuro mantenimiento será más costoso y repercutirá en la confianza de los clientes y en la moral del equipo.[3]

Las arquitecturas basadas en microservicios son una tendencia actual que emerge asociada a conceptos clave como la integración continua, el desarrollo centrado en el dominio del problema o el despliegue en contenedores. En estas arquitecturas diferentes funcionalidades se encapsulan en servicios pequeños y autónomos que cooperan entre ellos. En términos de diseño, principios como el de Responsabilidad Única son más fáciles de conseguir y los desafíos de organización del código pueden abordarse de formas más diversas por la baja granularidad de la arquitectura.

1.2 Objetivos

El objetivo de este proyecto es validar con un caso de estudio las ventajas e inconvenientes de una arquitectura basada en microservicios frente a una arquitectura monolítica. Concretamente,los objetivos específicos son:

- Desarrollar una misma aplicación para la venta de productos y la gestión de pedidos siguiendo dos arquitecturas diferentes: una basada en microservicios y otra monolítica.
- Comparar el proceso de desarrollo de ambos sistemas a lo largo del ciclo de vida del software.
- Evaluar cómo se pueden llevar a cabo diferentes modificaciones durante el mantenimiento de ambas aplicaciones una vez se ha finalizado su implementación.
- Verificar que una arquitectura basada en microservicios facilita alcanzar los requisitos no funcionales de escalabilidad y tolerancia a fallos frente a una arquitectura monolítica.

1.3 Estructura de la memoria

CAPÍTULO 2
Estado del arte

2.1 ¿Qué son los microservicios?

Según Newman, los microservicios son servicios pequeños y autónomos que trabajan conjuntamente.[4] Podemos desglosar esta definición así:

- Un **servicio** es un conjunto de funcionalidades que se expone a los clientes para que las empleen con diferentes propósitos.[5] La programación orientada a servicios es un paradigma que se aplica en las arquitecturas orientadas a servicios (SOA). El objetivo principal de SOA es desarrollar servicios que aporten valor al negocio y se adapten a los cambios en las necesidades de los clientes, de forma ágil y con el menor coste posible. Las arquitecturas orientadas a servicios no están asociadas a ninguna tecnología específica. En líneas generales, dividen un sistema en componentes que cambian por el mismo motivo y promueven la flexibilidad y los servicios compartidos frente a implementaciones específicas y óptimas. Son muchos los beneficios de estas arquitecturas, sin embargo existe una falta de consenso sobre cómo debe llevarse a cabo este tipo de arquitecturas en aspectos como los protocolos de comunicación a emplear o la granularidad de los servicios.[1] Los microservicios pueden entenderse como una aproximación específica de las arquitecturas SOA.

- Diseñar microservicios con el menor **tamaño** posible no debe ser el foco principal. En todo momento han de cumplirse los principios de cohesión y coherencia: el código relacionado debe agruparse conjuntamente porque será modificado por el mismo motivo. Una regla que puede ser aplicada según el autor Jon Eaves es establecer un tamaño para un microservicio tal que pueda ser completamente reescrito en 2 semanas.

- Los servicios han de ser lo menos acoplados posibles para garantizar la **autonomía** de cada uno. Cada microservicio es una entidad separada: cambian de forma independiente al resto y al hacerlo sus consumidores no necesiten ser modificados, a menos que se modifique el contrato entre ambas partes. Para lograrlo, lo más habitual es que cada servicio exponga una interfaz (API) y todas las comunicaciones se realicen mediante llamadas a través de la red.

  Otra definición interesante es la que aportan Lewis y Fowler. Según ellos, los microservicios son una aproximación para desarrollar una aplicación compuesta por pequeños servicios, cada uno ejecutándose en su propio proceso y comunicando a través de mecanismos ligeros. Estos servicios se construyen alrededor de las capacidades de negocio y se despliegan de forma independiente.[2]

2.2 Los microservicios en la fase de requisitos

La fase de requisitos del software es aquella en la que se elicitan, analizan, documentan, validan y mantienen los requisitos del sistema. Los requisitos del software expresan las necesidades y restricciones asociadas a un sistema. El artefacto principal que se produce en esta fase es el documento con la especificación de requisitos software (ERS). Una vez validado dicho documento por los stakeholders se puede iniciar la fase de diseño la solución.[**?**]

Los requisitos se pueden clasificar en funcionales y no funcionales. Los requisitos funcionales describen la funcionalidad que los usuarios esperan del sistema. Los requisitos no funcionales son restricciones impuestas sobre el sistema a desarrollar, estableciendo por ejemplo como de rápido o fiable ha de ser. Mientras que los primeros no incluyen ninguna mención relacionada con la tecnología que emplea el sistema, los segundos sí pueden establecer restricciones de este tipo. Por ejemplo, un requisito no funcional puede consistir en desarrollar una aplicación en un lenguaje de programación específico o hacer que esté disponible para diferentes sistemas operativos móviles. Por este motivo, los requisitos deben ser tenidos en cuenta a lo largo de todo el desarrollo del sistema.

Los requisitos funcionales y no funcionales son ortogonales en el sentido de que diferentes diseños de software pueden ofrecer la misma funcionalidad pero con distintos atributos de calidad. Los arquitectos software están más centrados en los requisitos no funcionales porque estos son los que conducen hacia la elección de una u otra arquitectura. Los requisitos no funcionales pueden influir en los patrones arquitectónicos a seguir, las futuras estrategias de implementación del sistema o la plataforma en la que se desplegará. [**?**]

2.3 Los microservicios en la fase de diseño

2.4 Los microservicios en la fase de implementación

2.5 Los microservicios en la fase de pruebas

2.6 Los microservicios en la fase de despliegue

2.7 Los microservicios en la fase de mantenimiento

2.8 Ventajas del desarrollo de software basado en microservicios

2.9 Inconvenientes del desarrollo de software basado en microservicios

CAPÍTULO 3
Conclusiones

????? ????????????? ????????????? ????????????? ????????????? ?????????????

Bibliografía

[1] Ali Arsanjani, Grady Booch, Toufic Boubez, Paul C. Brown, David Chappell, John DeVadoss, Thomas Erl, Nicolai Josuttis, Dirk Krafzig, Mark Little, Brian Loesgen, Anne Thomas Manes, Joe McKendrick, Steve Ross-Talbot, Stefan Tilkov, Clemens Utschig-Utschig, and Herbjörn Wilhelmsen. SOA Manifesto. *SOAManifesto*, 2009.

[2] James Lewis and Martin Fowler. Microservices, 2014.

[3] Robert C. Martin. *Clean Architecture: A Craftsman's Guide to Software Structure and Design*. Prentice Hall, 2017.

[4] Sam Newman. *Building Microservices*. O'Reilly, 2015.

[5] Wikipedia. Service (systems architecture).

APÉNDICE A

Configuración del sistema

????? ????????????? ????????????? ????????????? ????????????? ?????????????

A.1 Fase de inicialización

????? ????????????? ????????????? ????????????? ????????????? ?????????????