Trabajo de investigacion 1

Jhonnatan Mendez

September 3, 2023

1 Elaborar un mapa conceptual que represente los fundamentos de los sistemas distribuidos. Debe incluir conceptos clave como nodos, comunicación, recursos compartidos, latencia, escalabilidad, redundancia, tolerancia a fallos, entre otros. Cada concepto debe estar enlazado con una breve descripción y ejemplos de aplicación en situaciones reales.

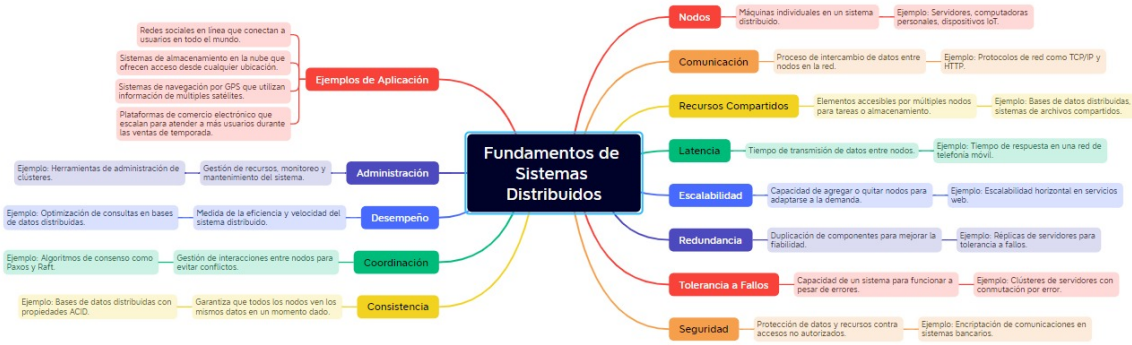

Figure 1: Mapa mental

2 Teniendo en cuenta la arquitectura de sistemas distribuidos microservicios, explicar qué son, por qué son relevantes en sistemas distribuidos, cómo se diferencian de otras arquitecturas, cómo se implementa, sus ventajas y desventajas, cómo aborda la escalabilidad y la tolerancia a fallos, y ejemplos de empresas o proyectos que la han adoptado.

2.1 Que son los microservicios?

Los microservicios son una arquitectura de software en la que una aplicación monolítica se descompone en componentes independientes y pequeños, llamados microservicios, que son responsables de funciones específicas del sistema.

2.2 Relevancia de los microservicios

Los microservicios son relevantes en sistemas distribuidos porque permiten dividir una aplicación en partes más pequeñas y manejables que se pueden implementar, escalar y actualizar de forma independiente. Esto hace que sea más fácil gestionar sistemas complejos y adaptarse a las demandas cambiantes de tráfico y funcionalidad.

2.3 Diferencias contra otras arquitecturas

- Monolítica: A diferencia de las aplicaciones monolíticas, en las que todo el código se ejecuta en un solo proceso, los microservicios dividen la aplicación en componentes separados que se ejecutan en procesos o contenedores independientes.
- Arquitectura orientada a servicios (SOA): Aunque comparten similitudes, los microservicios suelen ser más pequeños y ligeros que los servicios en una arquitectura SOA. Además, SOA se centra más en la reutilización de servicios, mientras que los microservicios se enfocan en la independencia y la agilidad.
- Arquitectura de funciones como servicio (FaaS): Los microservicios son generalmente más grandes que las funciones individuales en FaaS, que se ejecutan en respuesta a eventos específicos.

2.4 Como se implementa?

Los microservicios se implementan como unidades de código independientes y se ejecutan en contenedores (como Docker) o en máquinas virtuales. Cada microservicio tiene su propia base de datos, si es necesario, y se comunica con otros microservicios a través de API RESTful o mensajes.

2.5 Ventajas

- Escalabilidad: Los microservicios permiten escalar componentes individuales de una aplicación en función de las necesidades de rendimiento.
- Flexibilidad: Facilitan la adopción de nuevas tecnologías y el uso de diferentes lenguajes de programación para cada microservicio.
- Despliegue continuo: Los microservicios facilitan la implementación continua, ya que cada microservicio puede actualizarse y desplegarse de forma independiente.
- Facilita el trabajo en equipo: Diferentes equipos pueden desarrollar, implementar y mantener microservicios específicos, lo que mejora la eficiencia y la colaboración.

2.6 Desventajas

- Complejidad: Gestionar múltiples microservicios puede ser complicado y requerir herramientas de orquestación y monitoreo.
- Comunicación entre microservicios: La comunicación a través de la red puede generar latencia y problemas de rendimiento si no se gestiona adecuadamente.
- Consistencia de datos: Mantener la consistencia de datos en sistemas distribuidos puede ser un desafío.
- Mayor sobrecarga inicial: La creación de una infraestructura de microservicios puede requerir una inversión inicial significativa.

2.7 Escalabilidad y fallos

Los microservicios abordan la escalabilidad al permitir que los servicios se escalen de forma independiente según la demanda. En cuanto a la tolerancia a fallos, al dividir una aplicación en microservicios, un fallo en un servicio no afecta necesariamente a toda la aplicación, lo que mejora la resiliencia del sistema.

2.8 Ejemplos de proyectos

- Netflix: Netflix utiliza una arquitectura de microservicios para su plataforma de streaming, lo que le permite escalar y actualizar de manera eficiente.
- Amazon: Amazon ha adoptado microservicios en muchas de sus aplicaciones y servicios, como AWS Lambda.
- Uber: Uber utiliza una arquitectura basada en microservicios para su plataforma de transporte compartido..

3 Investigar un caso de una empresa que haya adoptado la arquitectura de microservicios y analizar los siguientes aspectos

La arquitectura de microservicios en Uber está estructurada de manera modular, donde cada microservicio se encarga de una función específica y se comunica con otros a través de interfaces bien definidas.

3.1 Cómo está estructurada la arquitectura de microservicios en la empresa.

1. Sistema Central de Coordinación: Uber cuenta con un sistema central que actúa como coordinador para enrutar solicitudes de viajes, administrar la autenticación de usuarios y realizar funciones de alto nivel para mantener la cohesión de la plataforma.
2. Gestión de Viajes: Este es uno de los componentes clave de Uber, donde se gestionan las solicitudes de viaje, la asignación de conductores y el seguimiento de la ubicación de los vehículos. Cada una de estas funciones se implementa como microservicios independientes.
3. Gestión de Pagos: Uber procesa una gran cantidad de transacciones financieras, por lo que la gestión de pagos es fundamental. Este componente se encarga de la facturación, la gestión de tarjetas de crédito y otros aspectos relacionados con el pago.
4. Gestión de Usuarios: La autenticación, la creación de perfiles de usuarios y la administración de cuentas se implementan como microservicios separados.
5. Geolocalización: La funcionalidad de geolocalización es esencial para el funcionamiento de Uber. Utilizan microservicios para rastrear la ubicación en tiempo real de conductores y pasajeros, así como para calcular las rutas óptimas.
6. Comunicaciones y Notificaciones: Los microservicios se encargan de enviar notificaciones en tiempo real a conductores y pasajeros, proporcionando información importante sobre el estado del viaje y las actualizaciones relevantes.

3.2 Comunicación entre microservicios

La comunicación entre los microservicios en Uber generalmente se realiza a través de API RESTful y protocolos de comunicación como HTTP o gRPC. Cada microservicio expone interfaces bien definidas que permiten a otros servicios interactuar con ellos de manera segura y eficiente.

3.3 Ventajas de la arquitectura de microservicios en el contexto de la empresa.

- Escalabilidad: Los microservicios permiten escalar componentes específicos de la plataforma de acuerdo con la demanda, lo que mejora la eficiencia y el rendimiento.
- Despliegue continuo: Los equipos pueden implementar actualizaciones de manera independiente, lo que acelera el tiempo de entrega de nuevas características y correcciones de errores.

- Mejora la independencia tecnológica: Cada microservicio puede utilizar las tecnologías más adecuadas para su función, lo que fomenta la innovación y la adopción de nuevas tecnologías.
- Resiliencia y tolerancia a fallos: Un fallo en un microservicio no afecta necesariamente a toda la plataforma, lo que mejora la resiliencia del sistema.

3.4 Desafíos y desventajas que han enfrentado al adoptar esta arquitectura.

- Complejidad operativa: Gestionar una gran cantidad de microservicios puede ser complejo y requiere herramientas de monitoreo y orquestación sofisticadas.
- Latencia de comunicación: La comunicación entre microservicios a través de la red puede generar latencia, lo que debe ser gestionado adecuadamente.
- Consistencia de datos: Mantener la consistencia de datos en sistemas distribuidos puede ser un desafío, y Uber ha desarrollado soluciones específicas para abordar este problema.

3.5 Cómo la arquitectura aborda la escalabilidad y la tolerancia a fallos.

La arquitectura de microservicios en Uber aborda la escalabilidad al permitir que los equipos escalen componentes individuales en función de la demanda. En cuanto a la tolerancia a fallos, al dividir la plataforma en microservicios, un fallo en uno de ellos no afecta necesariamente a los demás, lo que reduce la superficie de fallos y mejora la resiliencia del sistema.