**Microservices: Theory and Application**

Introducción a la arquitectura

El expositor es Lucas Krause, quien ha estado trabajando en este rubro por 10 años. El primer tema que se tocó es sobre la Arquitectura, las primeras partes que debe tener son la comunicación y patrones; es decir, patrones de diseño. Luego son la estandarización y las metas. El principal objetivo de estas dos es que estén de acuerdo con lo que estás haciendo y el diseño de este; poder llegar a la meta y ver si está bien o mal lo que uno hizo.

Seguido, se habló de Monolítico, el cual es un largo sistema o aplicación que hace muchas cosas, pero es difícil mantenerlas. Además, en las compañías ya conocen estos tipos de sistemas; por lo que, se les considera algo malo porque mientras crecen se vuelven más indefinibles. Su costo de mantenimiento deja atrás su beneficio, por eso es mejor no tenerlos en las empresas.

Entonces los microservicios son como si uno rompiera su empresa en dos, así que se trata de hacer un contexto delimitado por servicio. Uno puede ver que varias personas de la empresa migran de un monolítico a un microservicio, ya que es amistoso a grandes migraciones. Además, cuenta con tecnología agnóstica porque se pueden usar varias tecnologías y no una en particular.

Luego se explican algunos principios para que la implementación de un microservicio sea correcto. Estos son Encapsulación, Automatización, Dominio Centralizado, Descentralizado, Independiente, Fail-Safe y Observable. Si uno aplica estos principios podrá recoger los beneficios de la arquitectura de microservicios. La idea de la escalabilidad se basa en descomponer las funciones, separar los procesos en tamaños pequeños, con respecto a las similitudes que tienen. Por lo tanto, encontramos 3 dimensiones: x, duplicación horizontal; y, descomposición funcional; y z, particionamiento de datos. Primero se debe escalar por el eje y después por cualquiera de los otros dos, es lo que se recomienda.

El cambio al mantenimiento significa escribir de manera detalla el dominio centralizado de los microservicios, por lo que, se debe enfocar más en el problema de las empresas. Este cambio está envuelto en las industrias, el mantenimiento es el factor clave en los sistemas de software. También el aportan valor el uso de los principios SOLID y DRY.

Se nombra también sucesos que cambian actualmente, por ejemplo, Time is ripe, Automatización, Containers, Dev-ops, altos niveles de abstracciones. Seguidamente de eso, se nombre patrones y aproximaciones que usan los microservicios, estos son Command Query Responsibility Segregation (CQRS), Event Sourcing, API Gateway/Proxy y Orchestrated API.

Aproximaciones al éxito

Las definiciones del éxito son las siguientes:

* Ningún defecto (hacer lo que debe hacer)
* Cumplir las necesidades de los usuarios.
* Seguro.
* Escalable, el software es diseñado para soportar cantidades grandes de usuarios.
* Robusto, tolerante a fallos o cambios impredecibles.
* Fácil de manejar y lanzar.
* Dentro del presupuesto.
* Dentro del tiempo.

Se debe entender el negocio, el ciclo de vida de los procesos de software, historias, casos de uso, escenarios; la usabilidad e interacción de la empresa; descomponer los problemas dentro de partes manejables y fáciles de solucionar; y por último, el API economy.

La estructura organizativa; es decir, los servicios pertenecen a un equipo y no a varios de ellos. La simplicidad es la llave del éxito, el uso de patrones DRY y SOLID son de gran ayuda.

Beneficios

Performance: en ejecución, en desarrollo, y reducir el tiempo de mercado.

Cumplir las expectativas altas de los usuarios a través de la arquitectura, robusto y escalable.

Lenguajes ubicuos.

Retos

Más complejidad en sistemas distribuidos.

Sistemas de pruebas.

Transacciones distribuidas (consistencia eventual).

Manejo de sistemas.

Mayor memoria.

Organización y cultura.

Madurez

Por último, se pueden aplicar varias tecnologías a los microservicios. Uno de estos es CQRS (Command Query Responsibility Segregation). También se encuentra Domain Driven Design donde se almacena eventos. DDD/Bounded, acá se puede usar diferentes lenguajes en manera conjunta; y API Gateway donde existe la comunicación entre plataformas.