**Microservicios: teoría y aplicaciones**

**Architecure**

Todo lo que escribimos como software es malo ya que toma más tiempo mantenerlo de lo que beneficio que nos da. Al usar un monolito también se genera código espagueti el cual nos brinda el servicio deseado, pero a la hora de actualizarlo con nuevas funcionalidades esto demorará y será más difícil. Para eso vienen la arquitectura de microservicios que separan el código, estandariza como se escribirá el código para que no surjan estos problemas.

Para aplicar microservicios se necesitan principios:

* **Encapsulación**
* Automatización
* **Domain Centric:**
* Descentralizad
* **Independent**
* **Fail-safe**
* Observable

Con la arquitectura de microservicios el punto más importante es la escalabilidad se comienza por el business domain. Los microservicios fueron creados bajo el concepto de que separemos las funciones en partes bien específicas y que se entiendan y si vamos a hacer algo hay que hacerlo bien.

EL día de hoy se esta pasando a mantenimiento, ya que el costo de personas esta aumentando y el hardware bajando. Así que se necesita bajar costo de poder mantener el sistema. Para eso se usa los microservicios así se optimiza y reduce costos. Usando principios SOLID en el desarrollo se mejora la calidad del código y es mas fácil de ser mantenido.

Usan varios patrones los microservicios como el Command Query Responsibility Segregation(CQRS), Event sourcing, proxy, orcheasted api. Nos ayuda en la mantenibilidad y la integridad.

**Approach to success**

Para poder tener éxito se debemos entender bien el negocio, la Estructura de la organización, el tamaño y como implementar la tecnología a las metas de la empresa

El éxito tiene muchas definiciones según el problema que tengas puede ser que no tenga defectos que sea seguro, escalable, robusto, fácil de manejar y cambiar, dentro del Budget y que este a tiempo.

Para entender bien el negocio se debe desarmar los problemas en partes manejables y poner los procesos en casos de uso, escritos. También la estructura de la organización debe estar organizadas se usan patrones como SOLID o DRY.

El tamaño del equipo es importante, no debe ser muy grande y debe haber una comunicación buena. Depende más de la organización y como trabajan.

Todas las tecnologías se pueden usar dependerá de la elección que se haga según el problema y si se siguen los patrones y se aíslan los problemas.

Se debe de implementar una estrategia para particionar, el mejor es Single responsability principle.

Usar microservicios nos trae muchos beneficios como la mejorar de performance y reduce el tiempo de desarrollo, es más escalable y robusto. Ubiquitous Language solo se debe usar solo un lenguaje para la arquitectura. Con esto nos da un desarrollo más rápido, más fácil de testear, más barato hacer que el sistema escale y es a prueba de fallos ya que están aislados.

Tiene sus desventajas como son más complejos los sistemas distribuidos, debe haber un sistema de testeo puede ser un reto porque están separados. La organización y cultura de la organización. El grado de madurez.

En conclusión, se debe de hacer una trancision de los servicios monolíticos a los microservicios ya que en la actualidad las cosas cambian rápidamente y necesitan cambios y abaratar costos. Pero para implementar los microservicios es necesario tener claro las buenas prácticas como los principios solid y el uso de Single Responsability para que sea más fácil su transición.