Microservices: Theory and Application

Al hablar de arquitectura de software, cada empresa ve las cosas de una forma distinta. *The Timeless Way of Building* fue un libro que planteaba patrones arquitecturales para construir una arquitectura. La importancia de estos patrones radica en la comunicación de los elementos.

Todos están de acuerdo en lo que hacen y en su diseño con respecto a una meta común que permite mejor entendimiento entre las diversas partes del sistema

**Arquitectura monolítica**

Sistemas largos o aplicaciones que hacen muchas cosas y no pueden recibir mantenimiento (debido a que el costo y tiempo que demandan son muy grandes).

**Microservicios**

Se debe de tener entendimiento acerca de que es lo que se desea construir para dividir las funciones del negocio en contextos relacionados, cada uno atado a un servicio.

Utiliza técnicas de *refactoring* para adaptar la arquitectura monolítica con la que ya se cuenta. No deben de estar atados a una tecnología en específico. Deben de ser escalables, para generar independencia entre las diversas funciones del sistema. También deben de realizar una labor en específico, y hacerla de la mejor forma posible. Deben de ser enfocados al mantenimiento debido a la evolución de la industria, por lo que se deben de aplicar principios SOLID y DRY.

**Patrones y enfoques**

* CQRS: Separar las lecturas de escrituras para la optimización de *queries*
* *Event Sourcing*: Base para diseños de negocio que requieren confiabilidad y transparencia
* API Gateway / Proxy: Cambios dinámicos de los números y localidades de los servicios
* Orchestrated API

**Principios principales de los microservicios**

* Encapsulamiento
* Domain Centric
* Independientes
* A prueba de fallos

**Camino al éxito de los microservicios**

Entender el negocio y lo que se desea lograr, tener una estructura organizativa definida, definir el alcance de los servicios y cuestionar si es que las tecnologías a utilizar van acorde a lo que se desea lograr.

**Como entender el negocio**

* Partir los problemas en problemas más pequeños y sencillos
* Darle importancia a la usabilidad y las interacciones con el usuario
* Economía de API: Mostrar el valor de todas las áreas. Todas son interdependientes e importantes, todas buscan cumplir la meta general.

**Estructura organizacional**

Los equipos organizacionales no deben de ser muy grandes y debe de existir confianza entre los miembros para que exista buena comunicación. Estos grupos son dueños de un servicio. Lo simple y el uso de patrones contribuyen a llegar al éxito.

**Tecnologías**

Se prefieren lenguajes basados en eventos, bases de datos de grafos, índices de búsqueda optimizados y lenguajes como Java, .Net, NodeJs, Go entre otros. Sin embargo, uno puede elegir lo que mejor se adapta a los problemas del negocio.

**Beneficios de microservicios**

Rendimiento, cumplir con las expectativas de los usuarios a través una arquitectura robusta y escalable. Además, el costo de escalar la plataforma es reducido.

**Desafíos**

* Mayor complejidad en sistemas distribuidos
* Dificultades en realizar pruebas
* Mayor uso de memoria
* Manejo de sistemas más complicados