## Os 12 Princípios dos Microserviços: Garantindo Escalabilidade e Flexibilidade para Aplicações Modernas

Os 12 fatores para o desenvolvimento de microserviços oferecem diretrizes claras para criar aplicações distribuídas de alta disponibilidade, escalabilidade e facilidade de manutenção. Cada princípio aborda uma prática fundamental para alcançar o máximo potencial dos microserviços, desde o controle de dependências até a gerência de logs e administração. Vamos revisar cada princípio e entender seu papel na construção de sistemas eficientes e robustos.

### 1. ****Base de Código: Gestão Centralizada e Versionada****

Cada aplicação deve ter sua própria base de código rastreada em sistemas de controle de versão, como Git ou Subversion. Em microserviços, isso significa que cada serviço tem seu próprio repositório de código, permitindo uma independência completa. Isso facilita o entendimento e o gerenciamento do projeto, além de deixar mais claro como os serviços se conectam.

Em contrapartida em modelos de arquitetura monolítica podem conter repositórios únicos em ambientes de versionamento, tendo mais chances de conflitos em alterações e compllexificando a leitura, manutenção e implementações do projeto a longo prazo.

### 7. ****Vínculo de Porta: Serviços Expostos por Portas****

Cada microserviço deve ser exportado por uma porta específica, permitindo sua comunicação com outros serviços de maneira controlada e independente. A visibilidade entre serviços é garantida por port binding, promovendo a escalabilidade e assegurando que cada serviço possa se comunicar de forma independente e segura.

Esse é um fator que em uma arquitetura monolítica pode se tornar um problema por centralizar os serviços em uma só porta que além de ser causador de conflitos, separar cada serviço pode injetar muita complexidade ao projeto.

### 9. ****Descartabilidade: Inicialização e Desligamento Rápidos****

A capacidade de iniciar e desligar um microserviço rapidamente é fundamental para uma arquitetura resiliente. Esse princípio garante uma inicialização ágil e um desligamento gracioso (graceful stop), onde o processo libera os recursos adequadamente e, se necessário, salva seu estado. Esse comportamento é essencial para operações automáticas e resposta rápida a falhas.

Enquanto na arquitetura monolítica, além do risco de queda do sistema inteiro se uma das partes falharem, é muito provável que inicialização e velocidade dos serviços sejam comprometidas, esse é um fator que pode prejudicar o projeto inteiro a longo prazo e um dos principais motivos pelo qual deve-se optar pela arquitertura de microserviço.

fazer uma ponte com a arquitetura monolitica e tentar identificar por que esses fatores se diferem entre uma arquitetura monoliticaa e uma arquitetura de microservice