Microservices: Conceitos e Práticas

O que são Microservices?

- Arquitetura baseada em serviços independentes e autônomos
- Comparação com sistemas monolíticos
- Benefícios: escalabilidade, flexibilidade e deploy independente

Características Principais

- Serviços independentes com deploy isolado
- Organização por domínios de negócio
- Comunicação leve: HTTP, gRPC, Mensageria
- Alta resiliência e observabilidade

Quando Usar Microservices?

- Projetos com múltiplas equipes e domínios
- Necessidade de escalabilidade modular
- Inovação rápida por domínio

Desafios Comuns

- Complexidade de comunicação entre serviços
- Dificuldade de testes integrados
- Gerenciamento de consistência de dados
- Necessidade de observabilidade robusta

Comunicação entre Serviços

- Síncrona: REST, gRPC
- Assíncrona: Kafka, RabbitMQ
- Exemplo: Pedido → Pagamento → Entrega

Padrões Comuns

- API Gateway
- Service Discovery
- Circuit Breaker (Resilience4j)
- Configuração Centralizada (Spring Cloud Config)

Deploy e Entrega Contínua

- Canary Release

Organização de Times

- Times pequenos e multifuncionais
- Alinhamento com Conway’s Law
- Autonomia e responsabilidade por serviço

Tecnologias Populares

- Kotlin com Spring Boot e Spring Cloud
- Docker & Kubernetes
- Kafka, RabbitMQ
- Jaeger, Prometheus, Grafana

Boas Práticas

- Isolamento de falhas
- Versionamento de APIs
- Documentação com OpenAPI
- Monitoramento distribuído
- Testes de contrato (ex: Pact)

Exemplo: Solicitação de Cartão

- Serviços: ms-card, ms-customer, ms-notification
- Comunicação REST + Kafka
- Processo assíncrono com antifraude e notificação

Dicas para Começar

- Iniciar com arquitetura modular
- Migrar gradualmente
- Investir em testes e observabilidade desde o início

Leituras e Recursos

- “Building Microservices” – Sam Newman
- “Microservice Patterns” – Chris Richardson
- Spring.io, DevDocs, Baeldung, etc.

Encerramento

- Microservices permitem escalabilidade e autonomia
- Requerem maturidade técnica e boas práticas
- Obrigado!