Resenha do Artigo de Martin Fowler sobre Microsserviços

Introdução

Martin Fowler, em seu artigo sobre microsserviços, apresenta uma visão detalhada dessa abordagem arquitetural, destacando seus benefícios em comparação com sistemas monolíticos. Ele explora as características dos microsserviços, como a divisão da aplicação em pequenos serviços independentes, e discute os desafios e as melhores práticas para sua implementação.

Exemplo de Aplicação:

Em um sistema bancário digital, serviços como gestão de contas, pagamentos, análise de crédito e notificações podem ser implementados como microsserviços independentes. Isso permite que cada parte do sistema evolua separadamente, reduzindo riscos e facilitando a inovação.

O que são Microsserviços?

A arquitetura de microsserviços divide uma aplicação em pequenos serviços independentes, cada um executando seu próprio processo e se comunicando por meio de mecanismos leves, como APIs HTTP. Cada serviço é focado em uma capacidade de negócio específica e pode ser desenvolvido, implantado e escalado separadamente.

Contraste com Sistemas Monolíticos:

Em um sistema monolítico, toda a aplicação é construída como um único bloco, o que dificulta a escalabilidade e a manutenção. Qualquer pequena alteração no código exige a recompilação e redistribuição de toda a aplicação.

Exemplo de Aplicação:

Um sistema de e-commerce monolítico pode se tornar difícil de manter à medida que novas funcionalidades são adicionadas. A migração para microsserviços permite que serviços como estoque, processamento de pedidos e atendimento ao cliente sejam desenvolvidos e escalados independentemente.

Componentização via Serviços

A arquitetura de microsserviços promove a modularização por meio de serviços independentes, que se comunicam via chamadas remotas, como APIs REST ou mensagens assíncronas. Isso reduz o acoplamento entre os serviços e facilita a evolução do sistema.

Desafios:

Chamadas remotas são mais custosas que chamadas locais, exigindo um design cuidadoso para evitar problemas de desempenho.

A interface entre serviços deve ser bem definida para garantir a interoperabilidade.

Exemplo de Aplicação:

Em um sistema de gestão de pedidos, o serviço de pagamento pode ser separado do serviço de estoque, permitindo que cada um seja atualizado sem afetar o outro.

Organizados em torno de Capacidades de Negócio

A abordagem de microsserviços sugere que as equipes sejam organizadas em torno de capacidades de negócio, ou seja, cada equipe deve ser responsável por um conjunto de funcionalidades completas. Isso inclui interface do usuário, lógica de negócio e banco de dados.

Exemplo de Aplicação:

A empresa Compare the Market estruturou suas equipes para que cada uma fosse responsável por um produto específico, composto por múltiplos serviços que se comunicam via um barramento de mensagens.

Benefícios:

- Maior autonomia e agilidade no desenvolvimento.
- Redução de dependências entre equipes.

Produtos e não Projetos

No modelo tradicional de desenvolvimento, os projetos têm um início e um fim bem definidos. Após a conclusão, o código é entregue a uma equipe de manutenção, que pode não estar familiarizada com o desenvolvimento inicial.

Na abordagem de microsserviços, cada equipe é responsável pelo ciclo de vida completo do serviço, garantindo que ele funcione bem em produção. Isso cria uma relação mais próxima entre os desenvolvedores e os usuários.

Exemplo de Aplicação:

A Amazon adota o princípio "você constrói, você opera", onde as equipes que desenvolvem um serviço também são responsáveis por sua operação e suporte. Isso melhora a qualidade e a estabilidade da aplicação.

Endpoints Inteligentes e Pipelines Simples

A comunicação entre microsserviços deve ser simples e eficiente. A comunidade de microsserviços prefere endpoints inteligentes (serviços com lógica de negócio) e pipelines simples (mecanismos de comunicação leves).

Métodos de Comunicação:

- HTTP e APIs REST: Permitem requisições diretas entre serviços.
- Mensageria assíncrona: Utiliza filas de mensagens como RabbitMQ ou Kafka para comunicação desacoplada.

Exemplo de Aplicação:

Em um sistema de notificações, o serviço de envio de e-mails pode ser acionado por mensagens assíncronas, garantindo que o sistema principal não seja bloqueado enquanto o e-mail é enviado.

Governança Descentralizada

A governança descentralizada permite que cada equipe escolha as melhores ferramentas e tecnologias para seus serviços. Isso contrasta com a abordagem monolítica, onde a padronização em uma única plataforma tecnológica é comum.

Exemplo de Aplicação:

A Netflix utiliza diferentes tecnologias para seus serviços, como Node.js para interfaces simples e C++ para processamento em tempo real. Essa flexibilidade permite que cada serviço seja otimizado para sua função específica.

Benefícios:

- Maior adaptabilidade às necessidades de cada serviço.
- Incentivo à inovação e experimentação.

Gerenciamento Descentralizado de Dados

Na arquitetura de microsserviços, cada serviço gerencia seu próprio banco de dados, um conceito conhecido como Polyglot Persistence. Isso reduz a necessidade de transações distribuídas e prioriza a consistência eventual.

Exemplo de Aplicação:

Em um sistema de comércio eletrônico, o serviço de pedidos pode usar um banco de dados relacional, enquanto o serviço de recomendação de produtos pode usar um banco de dados NoSQL.

Benefícios:

- Maior independência entre serviços.
- Escalabilidade e flexibilidade no armazenamento de dados.

Automatização de Infraestrutura

A automação de infraestrutura é essencial para a implantação e operação de microsserviços. Ferramentas como Docker e Kubernetes facilitam a implantação e o gerenciamento de serviços em ambientes de nuvem.

Exemplo de Aplicação:

A Netflix utiliza automação de infraestrutura para implantar e escalar seus serviços rapidamente, garantindo alta disponibilidade e resiliência.

Benefícios:

- Redução de erros humanos.
- Implantação rápida e segura de novas versões.

Projeto para Falhas

A arquitetura de microsserviços exige que as aplicações tolerem falhas de componentes individuais. Cada chamada de serviço pode falhar, e os clientes precisam responder de forma grácil.

Exemplo de Aplicação:

A Netflix implementa a Simian Army, que causa falhas propositalmente para testar a resiliência do sistema. Isso garante que o sistema continue funcionando mesmo em condições adversas.

Benefícios:

- Maior confiabilidade e disponibilidade do sistema.
- Melhoria na experiência do usuário.

Design Evolutivo

Os microsserviços permitem um design evolutivo, onde os serviços podem ser substituídos ou atualizados independentemente. Isso facilita a adaptação do sistema às mudanças de negócio.

Exemplo de Aplicação:

O site do The Guardian migrou de uma arquitetura monolítica para microsserviços, permitindo a adição rápida de funcionalidades temporárias, como páginas especiais para eventos esportivos.

Benefícios:

- Flexibilidade para adicionar ou remover funcionalidades.
- Redução do risco em mudanças de negócio.

Conclusão

A arquitetura de microsserviços oferece diversas vantagens, como maior escalabilidade, facilidade de manutenção e independência no desenvolvimento. No entanto, também apresenta desafios, como a complexidade na comunicação entre serviços e a necessidade de um bom gerenciamento de APIs e dados distribuídos.

Exemplo de Aplicação:

Empresas como Amazon, Netflix e The Guardian adotaram microsserviços com sucesso, transformando a forma como seus sistemas são desenvolvidos e mantidos.

Quando bem aplicada, essa abordagem torna os sistemas mais flexíveis e preparados para crescer junto com o negócio.

Considerações Finais:

Embora os microsserviços sejam promissores, é importante avaliar cuidadosamente se essa abordagem é adequada para cada projeto. Sistemas pequenos e de baixa complexidade podem se beneficiar mais de uma arquitetura monolítica. A escolha da arquitetura certa depende das necessidades do negócio e da capacidade da equipe de desenvolvimento.