PUC Minas

Pontifícia Universidade Católica de Minas Gerais

Instituto de **C**iências **E**xatas e **I**nformática - **ICEI**

Engenharia de Software

Projeto de Software

Professor: Dr. João Paulo Aramuni

06 out. 2024

Explorando Microsserviços: Uma Resenha Crítica do Artigo de Fowler e Lewis

João Pedro Silva Braga

FOWLER, Martin. ***Microservices***. martinfowler.com, 25 mar. 2014. Disponível em: https://www.martinfowler.com/articles/microservices.html. Acesso em: 06 out. 2024.

O artigo “*Microservices*” foi escrito por Martin Fowler e James Lewis, publicado em 2014 no site de Fowler. Este texto se tornou referência no campo da engenharia de software ao discutir o padrão de microsserviços, uma arquitetura moderna que busca e resolve problemas associados à construção e manutenção de sistemas monolíticos. A pergunta do trabalho é explorar o conteúdo do conceito de microsserviço, respondendo à seguinte pergunta: “como o conceito de microsserviço pode ser relevante para criar sistemas mais escaláveis, flexíveis e gerenciáveis?”.

O texto está estruturado em três partes principais: introdução, desenvolvimento e conclusão. A introdução define o contexto sobre o qual a arquitetura de microsserviços será discutida. O desenvolvimento, sob o título "Características de uma Arquitetura de Microsserviços", explora nove aspectos cruciais: Componentização via Serviços, Organização em torno de Capacidades Empresariais, Foco em Produtos ao invés de Projetos, Pontos de Extremidade Inteligentes e *Pipes* Burros, Governança Descentralizada, Gestão de Dados Descentralizada, Automação de Infraestrutura, Design para Falhas e Design Evolucionário. Por fim, a conclusão levanta a questão crucial: "Os Microsserviços são o Futuro?".

A princípio, o artigo trata a arquitetura de microsserviços como um estilo arquitetônico que estrutura um aplicativo como uma coleção de pequenos serviços independentes, em oposição a um único aplicativo monolítico. Dessa forma, cada serviço é executado em seu próprio processo, comunicando-se com outros serviços por meio de interfaces, normalmente APIs HTTP. Os serviços são organizados em torno de capacidades de negócios, implantados de forma independente usando processos automatizados e podem utilizar diferentes linguagens de programação e tecnologias de armazenamento de dados.

Embora o texto deixe claro que não haja uma definição precisa, a abordagem utilizada sobre arquitetura de microsserviços enfatiza a modularidade, a implantação independente e o controle descentralizado. Isso contrasta com as aplicações monolíticas tradicionais, que podem se tornar cada vez mais difíceis de manter, dimensionar e implantar ao longo do tempo. Sendo assim, ao decompor um sistema em serviços menores e desacoplados, a arquitetura de microsserviços visa melhorar a modularidade, acelerar os ciclos de lançamento, facilitar a manutenção e aumentar a escalabilidade.

Posteriormente, as características da arquitetura de microsserviços são abordadas, trazendo o que, na visão do autor, são características comuns para arquiteturas que se encaixam no rótulo de microsserviços. A seção “Componentização via Serviços” explora a componentização dentro de arquiteturas de microsserviços, argumentando que os serviços, ao invés de bibliotecas, oferecem uma forma superior de construir componentes, definido pelo autor como uma unidade de software atualizável e substituível de forma independente.

Ainda que ambas sejam formas de componentização, os serviços se comunicam por meio de chamadas de processo externo, enquanto as bibliotecas usam chamadas de função dentro do processo. Essa diferença torna os serviços implantáveis de forma independente, permitindo atualizações e implantações isoladas sem reimplantar o aplicativo inteiro. A desvantagem dessa abordagem inclui o aumento da sobrecarga de chamadas remotas e interfaces potencialmente mais complexas em comparação com bibliotecas. Apesar dessas desvantagens, o autor sugere que os benefícios da modularidade e do desacoplamento oferecidos pelos serviços superam as desvantagens na construção de uma arquitetura de microsserviços.

A seção “Organizado em torno de capacidades empresariais” defende a organização de microsserviços em torno de capacidades de negócios em vez de camadas tecnológicas. O autor argumenta que estruturar equipes em torno de capacidades de negócios com cada equipe administrando um serviço de pilha completa, leva a melhor alinhamento e menos sobrecarga em comparação com equipes divididas por função técnica (por exemplo, interface do usuário, banco de dados).

Citando a Lei de *Conway*, o autor afirma que essa organização baseada em serviços leva a uma correspondência mais estreita entre a arquitetura do sistema e a estrutura organizacional, promovendo autonomia e agilidade. Essa abordagem é contrastada com aplicações monolíticas, que geralmente enfrentam desafios na manutenção de limites modulares claros quando alinhadas às capacidades de negócios.

A seção “Produtos não Projetos” destaca uma mudança de mentalidade prevalente em arquiteturas de microsserviços: passar de um modelo orientado a projetos para uma abordagem centrada no produto. Em vez de entregar o software para uma equipe de manutenção separada após a conclusão, as equipes de microsserviços geralmente assumem a propriedade total de um produto durante todo o seu ciclo de vida – um conceito frequentemente referido como “você cria, você executa”.

Essa propriedade incentiva um relacionamento mais estreito entre os desenvolvedores e suas operações de software e usuários finais, resultando em um melhor entendimento das necessidades do usuário e um tempo de resposta mais rápido aos problemas. Apesar dessa mentalidade centrada no produto não seja exclusiva dos microsserviços, o autor argumenta que a natureza granular dos serviços facilita o estabelecimento de conexões diretas entre os desenvolvedores e os usuários dos serviços,

promovendo assim a responsabilidade e uma perspectiva de longo prazo na evolução do produto.

Segundo os autores, em vez de concentrar a inteligência na camada de comunicação, como observado em abordagens como ESB, a arquitetura de microsserviços defende o conceito de "pontos de extremidade inteligentes e *pipes* burros". Isso significa que a lógica de negócio e a complexidade residem nos próprios serviços (*endpoints*), que se comunicam entre si através de protocolos leves e simples como HTTP/REST e mensageria.

Essa abordagem, inspirada na web e em sistemas Unix, simplifica a comunicação, facilita a implementação de cache e mantém o acoplamento entre os serviços baixo. A migração de um monólito para microsserviços exige, portanto, repensar o padrão de comunicação, migrando de um modelo de chamadas frequentes entre classes e métodos para uma abordagem mais desacoplada, evitando sobrecarga e "conversas" desnecessárias entre os componentes.

Em contraste com a governança centralizada, que muitas vezes limita a escolha tecnológica e a autonomia das equipes, a comunidade de microsserviços prefere uma abordagem descentralizada. Isso significa dar liberdade para cada serviço utilizar a linguagem de programação e tecnologias mais adequadas, promovendo a "ferramenta certa para o trabalho certo", segundo o autor na seção “Governança Descentralizada”.

Além do mais, o texto trata que a padronização, em vez de ser imposta por documentos, emerge da criação e compartilhamento de ferramentas internas que resolvem problemas comuns relacionados a bancos de dados, comunicação entre processos e automação de infraestrutura. A ênfase está em contratos de serviço flexíveis, como os "*Tolerant Reader*" e "*Consumer-Driven Contracts*", que permitem a evolução independente dos serviços e podem ser integrados ao processo de desenvolvimento. Essa abordagem, juntamente com práticas como "*build it / run it*", popularizada pela Amazon e Netflix, aumenta a autonomia e responsabilidade das equipes por todo o ciclo de vida do software.

A seção “Gestão de Dados Descentralizada” destaca a descentralização do gerenciamento de dados como um diferencial importante dos microsserviços. Isso significa que cada serviço tem autonomia para definir seu próprio modelo conceitual e escolher a tecnologia de armazenamento de dados mais adequada às suas necessidades, adotando o conceito de "persistência poliglota". Essa abordagem contrasta com a centralização em um único banco de dados geralmente vista em aplicações monolíticas, proporcionando maior flexibilidade e adaptabilidade.

A descentralização, porém, traz o desafio da consistência de dados entre os serviços. Em vez de depender de transações distribuídas, complexas e propensas a erros, os microsserviços favorecem a "consistência eventual" e a coordenação transacional. Problemas de inconsistência são tratados com operações de compensação, espelhando a realidade de muitos negócios que lidam com certo grau de inconsistência para responder rapidamente às demandas do mercado. Essa mudança exige adaptação das equipes de desenvolvimento, mas pode trazer benefícios significativos em termos de autonomia, flexibilidade e velocidade de resposta.

O autor aborda, também, a “Automação de Infraestrutura” e como ela se tornou essencial para o desenvolvimento e entrega de software, especialmente no contexto de microsserviços. A evolução da nuvem e de práticas como Entrega Contínua (CD) simplificou

a construção, implantação e operação de sistemas complexos. Através de pipelines de build automatizados, testes são realizados a cada etapa e a promoção de software funcional para os diferentes ambientes ocorre de forma automática e confiável.

A ênfase na automação impulsionou a criação de ferramentas que facilitam o dia a dia de desenvolvedores e equipes de operações, como as ferramentas *open source* da Netflix e o Dropwizard. Tais ferramentas auxiliam na gestão de código, criação de artefatos, configuração de serviços e monitoramento. Embora a automação beneficie qualquer tipo de arquitetura, ela se torna ainda mais crucial em um cenário de microsserviços, onde a gestão da operação se torna desafiadora devido à quantidade de serviços independentes.

A seção “Projetar para o fracasso” aborda a importância crítica da tolerância a falhas em arquiteturas de microsserviços. Diferente de um monólito, a falha de um único serviço pode impactar todo o sistema. Por isso, microsserviços devem ser projetados para lidar com essas falhas de forma resiliente.

A implementação de padrões juntamente com testes automatizados em produção, garantem a detecção rápida de falhas e a recuperação automática sempre que possível. O monitoramento em tempo real, tanto de métricas técnicas quanto de negócio, é essencial para identificar comportamentos inesperados e garantir a saúde do sistema. Adicionalmente, o uso preferencial de comunicação assíncrona entre serviços, evitando chamadas síncronas em cascata, minimiza o impacto da indisponibilidade de um serviço. A ênfase em monitoramento, resiliência e comunicação assíncrona é fundamental para construir sistemas de microsserviços robustos e confiáveis.

Também, é apresentado o “Design Evolucionário” que defende que a decomposição em microsserviços se alinha ao design evolutivo, permitindo que times de desenvolvimento controlem e implementem mudanças de forma rápida e eficiente, sem comprometer a estabilidade do sistema como um todo. A granularidade dos serviços oferece a flexibilidade de modificar, adicionar ou remover funcionalidades com maior autonomia, sem a necessidade de reconstruir ou reimplantar toda a aplicação a cada mudança.

A modularidade, guiada pelo padrão de mudanças, garante que elementos com ciclos de vida similares permaneçam juntos, enquanto a granularidade dos releases permite que apenas os serviços modificados sejam reimplantados, acelerando o processo de entrega e reduzindo o impacto de possíveis falhas. A compatibilidade entre serviços é crucial nesse contexto, priorizando tolerância a mudanças em detrimento do versionamento, que deve ser usado com cautela.

Por fim, o autor apresenta uma conclusão, trazendo a pergunta “Os microsserviços são o futuro?”, em que reconhece o potencial da arquitetura de microsserviços, mas recomenda cautela em sua adoção. Apesar de casos de sucesso em empresas como Amazon e Netflix, a técnica ainda é considerada recente e seus efeitos a longo prazo precisam ser melhor compreendidos.

A modularidade e flexibilidade dos microsserviços são atrativas, mas trazem desafios como a complexidade na comunicação entre serviços, refatoração trabalhosa e a necessidade de times experientes e bem preparados. A decisão de adotar essa arquitetura deve considerar cuidadosamente os custos e os benefícios, optando por um monólito modular em caso de dúvida, especialmente com times menos experientes, e considerando a migração para microsserviços apenas quando realmente necessário.

Portanto, o artigo "*Microservices*" se destaca por sua clareza, oferecendo uma introdução bem fundamentada. Os autores argumentam de forma convincente que a modularidade oferecida pelos microsserviços torna os sistemas mais adaptáveis às mudanças e à escalabilidade. A originalidade está em como eles apresentam a comparação entre sistemas monolíticos e distribuídos, abordando a questão de forma prática e acessível tanto para engenheiros de software quanto para gestores.

No entanto, uma limitação do artigo reside na falta de exemplos mais profundos sobre como implementar algumas das práticas descritas. Além disso, o artigo poderia ser enriquecido com mais informações sobre as dificuldades práticas enfrentadas ao migrar de uma arquitetura monolítica para uma de microsserviços e poderia explorar melhor as implicações para equipes com pouca experiência em sistemas distribuídos, oferecendo orientações mais detalhadas sobre como superar as dificuldades técnicas.

Em suma, o artigo apresenta uma contribuição valiosa para a área de arquitetura de software, especialmente para equipes que buscam criar sistemas escaláveis e resilientes. Ele atinge com sucesso seus objetivos de esclarecer o conceito e os benefícios dos microsserviços, além de abrir caminhos para discussões mais profundas sobre a adoção dessa arquitetura. Os resultados têm implicações importantes para empresas que desejam adaptar seus sistemas ao crescimento rápido ou à necessidade de inovação constante.

Dessa maneira, ele oferece uma visão abrangente e acessível sobre os desafios e benefícios dessa arquitetura, destacando sua capacidade de transformar a maneira como os sistemas de software são construídos e mantidos. Embora o texto apresente alguns pontos fracos em relação à profundidade de exemplos práticos, ele ainda fornece uma excelente base para engenheiros de software e líderes tecnológicos. No final, a principal conclusão é que os microsserviços, embora não sejam uma solução universal, oferecem uma maneira eficaz de enfrentar a complexidade e a demanda por sistemas mais ágeis e escaláveis.