Resenha artigo “Microservices”- Ana Júlia Teixeira Cândido
O artigo de Martin Fowler sobre microserviços traz uma discussão extremamente atual e relevante, especialmente para alguém como eu, que está cursando uma área relacionada à tecnologia. A abordagem de microserviços, que tem ganhado cada vez mais espaço no desenvolvimento de software, é apresentada de maneira prática e estratégica, fazendo com que qualquer estudante ou profissional de tecnologia reflita sobre os desafios e oportunidades dessa arquitetura em comparação com o modelo monolítico tradicional.

Logo no início do artigo, Fowler aborda a tendência natural de ignorarmos novas terminologias até que estas mostrem seu valor. Ele reconhece que, apesar disso, os microserviços vêm se consolidando como uma solução eficaz em diversos projetos, especialmente no desenvolvimento de grandes aplicações corporativas. A forma como o autor compara a arquitetura de microserviços com a arquitetura monolítica é um ponto de destaque, já que a maioria dos estudantes de tecnologia, como eu, já se deparou com os desafios de um sistema monolítico. A ideia de dividir uma aplicação em pequenos serviços independentes que podem ser implantados e escalados separadamente é claramente uma solução para uma das maiores dores de cabeça de quem desenvolve sistemas grandes: a dificuldade de atualizar pequenas partes sem impactar o restante do sistema.

A comparação entre essas duas arquiteturas, microserviços e monolítica, ajuda a ilustrar como a complexidade aumenta à medida que uma aplicação cresce. No monolito, pequenas mudanças podem exigir recompilação e redistribuição de todo o sistema, algo que em um ambiente de produção é extremamente arriscado e demorado. Já nos microserviços, ao dividir a aplicação em componentes menores, é possível resolver esses problemas de modularidade e escalabilidade. Para mim, que ainda estou aprendendo as nuances dessas arquiteturas, essa é uma das maiores vantagens dessa abordagem.

Além disso, o artigo explica as características fundamentais dos microserviços, como a "componentização via serviços" e a organização em torno de "capacidades de negócios". Fiquei particularmente interessado na ideia de dividir uma aplicação com base nas funcionalidades de negócios. Isso não só torna o desenvolvimento mais eficiente, como também permite que as equipes se organizem de forma mais coesa, facilitando a colaboração e o foco em objetivos específicos. Fowler menciona a Lei de Conway, que sugere que a estrutura organizacional de uma empresa acaba refletida na arquitetura do software. Essa parte me fez refletir bastante sobre a importância de alinhar a estrutura de desenvolvimento com os objetivos de negócio.

Um ponto levantado no texto que me intrigou foi a falta de uma definição clara sobre o que constitui um microserviço. Cada empresa parece adotar suas próprias práticas e, com isso, surge o debate sobre o tamanho ideal de um serviço. A referência ao "time de duas pizzas" da Amazon, que sugere que o tamanho da

equipe de microserviços deve ser pequeno o suficiente para ser alimentado por duas pizzas, é uma forma leve de exemplificar a importância de manter a simplicidade, evitando o inchaço das equipes e dos próprios serviços.

Ao falar sobre a distinção entre microserviços e SOA (Arquitetura Orientada a Serviços), Fowler esclarece que, embora ambos compartilhem semelhanças, o foco prático de implementação é bem diferente. As implementações de SOA muitas vezes falharam ao criar sistemas complexos e difíceis de gerenciar, enquanto os microserviços, com sua abordagem mais simples, focada em "endpoints inteligentes e pipes simples", têm sido mais eficazes. Para quem já teve contato com SOA, essa distinção é útil para entender o porquê de os microserviços serem vistos como uma evolução, não apenas uma alternativa.

Outro aspecto interessante abordado no artigo é a "governança descentralizada". A possibilidade de escolher as melhores ferramentas e linguagens para cada serviço sem se prender a padrões rígidos é um sonho para qualquer desenvolvedor. No entanto, essa liberdade vem com uma responsabilidade maior, algo que Fowler alerta: ela deve ser usada com cautela, pois a falta de padronização pode gerar complexidades indesejadas a longo prazo.

O artigo também faz uma crítica ao tradicional modelo de "projetos", sugerindo que equipes devam focar em "produtos", ou seja, serem responsáveis pelo ciclo de vida completo de um serviço, incluindo sua operação em produção. Isso é particularmente relevante em um cenário de desenvolvimento ágil e de DevOps, onde a entrega contínua e a operação eficiente se tornaram parte fundamental do processo de software. Essa transição de mentalidade parece inevitável para garantir a qualidade e a manutenção de longo prazo.

Outro ponto que capturou minha atenção foi a discussão sobre persistência poliglota, ou seja, a possibilidade de cada microserviço gerenciar seu próprio banco de dados. Embora isso adicione uma camada de complexidade, faz sentido em um mundo onde flexibilidade e velocidade de resposta ao mercado são cruciais. A conexão que Fowler faz entre essa abordagem e a ideia de consistência eventual, aceitando um certo grau de inconsistência para favorecer a agilidade, é fascinante. Para quem, como eu, está acostumado a pensar em sistemas como algo que deve ser sempre preciso e consistente, essa mudança de paradigma é desafiadora, mas faz total sentido no ritmo acelerado dos negócios atuais.

Fowler também enfatiza a importância da automação da infraestrutura, especialmente no contexto de integração e entrega contínuas (CI/CD). A automação reduz a complexidade operacional e torna o processo de lançamento de software algo quase "entediante", um conceito que parece paradoxal, mas que reflete o nível de controle e eficiência que a automação proporciona. Para alguém que está se

preparando para o mercado de trabalho, essa é uma área que definitivamente precisa de atenção, já que dominar a automação será um diferencial competitivo.

Por outro lado, o autor é realista quanto aos desafios dos microserviços. Ele aborda a necessidade de projetar sistemas distribuídos pensando em falhas, o que aumenta a complexidade. O exemplo da Netflix, que usa ferramentas para simular falhas e testar a resiliência dos serviços, mostra o nível de maturidade que as equipes precisam ter para trabalhar com essa arquitetura de forma eficaz. É inspirador, mas também revela que adotar microserviços exige uma curva de aprendizado e um nível técnico elevado.

Fowler faz um alerta importante sobre o nível de habilidade necessário para implementar essa arquitetura com sucesso. Equipes menos experientes podem acabar criando mais problemas do que soluções, caso tentem adotar os microserviços de maneira precipitada. Nesse sentido, começar com uma arquitetura monolítica modular e evoluir gradualmente para microserviços parece uma abordagem mais sensata para muitas empresas.

Ao concluir, Fowler é otimista sobre o futuro dos microserviços, reconhecendo suas raízes em princípios antigos, como os do Unix, mas sem ignorar os desafios que essa arquitetura apresenta. Para mim, o artigo foi esclarecedor ao mostrar que, embora os microserviços ofereçam inúmeras vantagens, como escalabilidade e modularidade, eles também trazem desafios que exigem um preparo técnico e estratégico robusto. Adotar microserviços deve ser uma decisão ponderada, considerando o contexto da equipe e da organização.

No final das contas, para alguém que está ingressando na área de tecnologia, os microserviços representam uma resposta moderna aos problemas das arquiteturas monolíticas, especialmente em um mundo cada vez mais voltado para a computação em nuvem. O artigo de Martin Fowler é uma leitura indispensável para qualquer estudante de tecnologia, pois vai além de apenas explicar os aspectos técnicos dos microserviços, abordando também suas implicações estratégicas e organizacionais.