Microservices de Martin Fowler: Quando o Hype Encontra a Realidade

Fowler define microservices como uma forma de desenvolver uma aplicação como um conjunto de pequenos serviços, cada um rodando em seu próprio processo e se comunicando através de APIs HTTP simples. Parece óbvio hoje em dia, mas em 2014 isso era revolucionário. A maioria ainda estava lutando com aplicações monolíticas gigantes que demoravam uma eternidade para compilar e fazer deploy.

O que mais me impressiona é como Fowler consegue ser preciso sem ser dogmático. Ele não diz "faça assim ou morra", mas sim "vimos essas características em sistemas que funcionam bem". É uma diferença sutil, mas fundamental.

A parte central do artigo são as nove características dos microservices. Componentização via Serviços parece básico, mas quantas vezes já vi equipes tentando "componentizar" criando mais bibliotecas internas? A diferença é que com serviços, se você quebra algo, pelo menos quebra só aquela parte. Com bibliotecas compartilhadas, uma mudança pode explodir meio sistema.

Organização em torno de Capacidades de Negócio foi um soco no estômago para mim. Lembro de projetos onde tínhamos o "time de front-end", o "time de back-end" e o "time de banco". Para implementar uma funcionalidade simples, precisávamos coordenar três equipes diferentes. É a Lei de Conway na prática: sua arquitetura vai refletir sua estrutura organizacional, querendo ou não.

A ideia de Produtos, não Projetos, "you build it, you run it", mudou minha perspectiva sobre responsabilidade. Não é só sobre código, é sobre ownership verdadeiro. Quando você é acordado às 3h da manhã porque SEU serviço caiu, garanto que você vai pensar duas vezes antes de fazer aquela gambiarra "temporária".

Endpoints Inteligentes e Pipes Simples é uma filosofia que só faz sentido quando você já sofreu com ESBs (Enterprise Service Bus) complexos. A ideia é simples: mantenha a lógica nos pontos finais e use protocolos bobos como HTTP. Nada de roteamento mágico ou transformações no meio do caminho.

Governança Descentralizada foi uma revelação pessoal. Por anos achei que padronização total era o santo graal. Mas quando você tem 50 serviços diferentes, usar Java para tudo porque "é o padrão da empresa" vira um tiro no pé. Algumas coisas pedem Node.js, outras Python. A vida é mais fácil quando você deixa cada time escolher a ferramenta certa.

O ponto sobre Gestão Descentralizada de Dados ainda gera controvérsia. Já participei de discussões acaloradas sobre isso. Claro que ter cada serviço com seu próprio banco é mais complexo, mas a autonomia que isso proporciona é incrível. Não precisa mais esperar o DBA liberar aquela mudança no schema que vai afetar outros 10 sistemas.

Fowler não vende microservices como solução para todos os problemas. Ele é bem claro sobre as dificuldades.

Design para Falhas não é opcional - é sobrevivência. Quando você tem uma aplicação monolítica, ou funciona tudo ou quebra tudo. Com microservices, sempre tem alguma coisa quebrada em algum lugar. Aprender a viver com falhas parciais é uma mudança de mindset gigante.

A parte sobre Automação de Infraestrutura era quase profética. Em 2014, Docker ainda estava engatinhando e Kubernetes nem existia. Mas Fowler já sabia que sem automação pesada, microservices seriam um pesadelo operacional. Hoje, com containers e orquestradores, ficou mais fácil, mas o princípio continua: se você não automatiza, não escala.

Algumas coisas ficaram datadas. A discussão sobre monitoramento, por exemplo, é bem superficial comparada com o que temos hoje com observabilidade, distributed tracing e service meshes.

O que sinto falta é uma discussão mais aprofundada sobre quando NÃO usar microservices. O artigo menciona isso, mas poderia ser mais enfático. Já vi muitas startups tentando começar com microservices porque "é o jeito moderno" e se afogando em complexidade desnecessária.

Esse artigo foi um divisor de águas. Não porque inventou microservices (como o próprio Fowler reconhece, empresas como Netflix e Amazon já faziam isso), mas porque deu nome aos bois e criou um vocabulário comum.

A maior lição para mim foi entender que arquitetura não é sobre escolher entre "certo" e "errado", mas sobre trade-offs conscientes. Microservices não são melhores que monólitos - são diferentes, com vantagens e desvantagens específicas.

Para quem está liderando times, a discussão sobre estrutura organizacional é ouro puro. Conway's Law não é apenas uma curiosidade acadêmica - é uma força da natureza que você pode trabalhar a favor ou contra.

A realidade é mais nuançada. Microservices resolvem alguns problemas e criam outros. O importante é entender quando usar cada abordagem. E para isso, começar entendendo os fundamentos que Fowler estabeleceu continua sendo essencial.

Dez anos depois, o artigo de Fowler sobre microservices continua relevante não pelas tecnologias que menciona, mas pelos princípios que estabelece. É uma leitura que muda a forma como você pensa sobre arquitetura de software, organizações e a própria natureza do desenvolvimento de sistemas.