Microservices: a definition of this new architectural term

Rafaela Oliveira Ribeiro

O artigo apresenta o termo “Microservices” como um estilo arquitetural que vem ganhando força para o desenvolvimento de aplicações empresariais. Em vez de construir um sistema inteiro de uma só vez, a proposta é dividir a aplicação em vários serviços pequenos, cada um rodando de forma independente e comunicando-se por mecanismos leves, geralmente APIs HTTP ou mensageria. A ideia central é organizar os serviços em torno de capacidades de negócio, permitindo que cada parte evolua no seu próprio ritmo e possa ser implantada e escalada sem depender do restante do sistema.

Para explicar essa mudança, os autores comparam microservices com a abordagem monolítica. No monólito, qualquer alteração, por menor que seja, exige um novo build e deploy de toda a aplicação. Além disso, com o tempo, a modularidade interna do monólito tende a se deteriorar, e a escalabilidade costuma acontecer no nível do todo, e não das partes que realmente precisam de mais recursos. Já os microservices oferecem fronteiras mais firmes entre módulos e permitem deploys independentes por serviço. Isso combina bem com a realidade de nuvem e com ciclos de mudança rápidos.

Um aspecto importante é a organização por capacidade de negócio e a mentalidade de produto. Em vez de times que entregam um projeto e depois o repassam para manutenção, a visão é de “você constrói, você opera”. O mesmo time cria, implanta, monitora e evolui o serviço ao longo de seu ciclo de vida. Essa proximidade com a operação e com os usuários tende a gerar decisões técnicas mais responsáveis e melhorias contínuas focadas nos resultados do negócio.

Na comunicação entre serviços, os autores defendem “endpoints inteligentes e encanamento simples”. A lógica deve estar nas bordas (nos serviços), enquanto o meio de comunicação permanece simples, sem um barramento pesado orquestrando tudo. Em práticas, isso se traduz no uso de HTTP/REST e filas leves como RabbitMQ ou ZeroMQ, com coreografia entre serviços em vez de orquestração centralizada. Ao migrar de chamadas em memória para chamadas remotas, é essencial evitar conversas “tagarelas” e adotar interfaces mais grossas, reduzindo a latência e a fragilidade.

Outro ponto é a descentralização da governança e dos dados. Em microservices, cada equipe pode escolher a tecnologia que melhor resolve seu problema (linguagem, framework e banco), favorecendo a ideia de ‘ferramenta certa para o trabalho’. Do lado dos dados, prevalece o conceito de ‘polyglot persistence’, no qual cada serviço administra seu próprio armazenamento. Em vez de transações distribuídas, que são complexas, adota-se consistência eventual com operações de compensação quando necessário.

A infraestrutura e a entrega contínua têm papel decisivo. Equipes acostumadas a Integração Contínua e Entrega Contínua automatizam testes, provisionamento e deploy, tornando a promoção entre ambientes algo “entediantemente confiável”. Em produção, porém, o cenário muda: muitos serviços exigem observabilidade madura, com métricas técnicas e de negócio, dashboards por serviço, e mecanismos de resiliência como timeouts, retries, circuit breakers e testes de falha planejados. O objetivo é detectar rapidamente comportamentos emergentes indesejados e corrigi-los antes que impactem o usuário.

Os autores também adotam um tom de otimismo cauteloso. Microservices trazem benefícios claros quando há limites de domínio bem definidos, necessidade de escalar e implantar partes de forma independente e uma cultura sólida de automação e DevOps. Ao mesmo tempo, alertam para os riscos: definir fronteiras de serviço ruins torna refatorações difíceis; a complexidade pode migrar das partes internas para as integrações; e equipes pouco experientes podem criar arquiteturas desordenadas, talvez até piores do que um monólito. Por isso, uma recomendação prática é começar com um monólito bem modular e, conforme as dores apareçam, extrair serviços gradualmente.

Como resenha, o texto destaca que microservices não são uma moda gratuita, mas também não são uma solução mágica. Eles reforçam a modularidade por meio de limites explícitos e autonomia de times, promovem evolução contínua e rapidez de entrega, e se alinham à realidade de nuvem e mudanças frequentes. Em contrapartida, exigem maturidade técnica, automação, disciplina de contratos e monitoramento robusto. A conclusão é equilibrada: vale considerar microservices com seriedade, desde que o contexto do produto, o nível de habilidade do time e a capacidade operacional sustentem a complexidade extra que essa arquitetura introduz.