A arquitetura de microsserviços tem ganhado destaque como uma abordagem inovadora e eficiente para o desenvolvimento de aplicações de negócios, permitindo que as equipes sejam mais ágeis e respondam rapidamente às mudanças do mercado. No entanto, a descentralização da gestão de dados, um dos seus princípios fundamentais, traz consigo uma série de desafios e considerações. Em um ambiente de negócios, diferentes áreas, como vendas e suporte, podem ter visões diferentes sobre um cliente, gerando lacunas que dificultam a integração entre sistemas. A utilização do conceito de contexto limitado do Domain Driven Design (DDD) permite uma separação clara entre esses contextos, favorecendo uma compreensão mais profunda das relações entre os serviços e facilitando a gestão descentralizada dos dados.

Os microsserviços também descentralizam as decisões sobre armazenamento de dados, permitindo que cada serviço utilize seu próprio banco de dados, prática conhecida como Polyglot Persistence. Isso possibilita a escolha de diferentes tecnologias de banco de dados para atender às necessidades específicas de cada serviço, em contraste com a abordagem monolítica que tende a usar um único banco de dados para toda a aplicação. No entanto, essa descentralização acarreta desafios no gerenciamento de inconsistências, uma vez que a coordenação de atualizações sem transações distribuídas pode ser complexa. As equipes precisam adotar uma abordagem de coordenação sem transações, reconhecendo que a consistência entre serviços pode ser apenas eventual, e problemas podem ser resolvidos através de operações de compensação.

A automação de infraestrutura desempenha um papel crucial na gestão de microservices, especialmente em um contexto de Continuous Delivery (CD) e Continuous Integration (CI). As equipes que utilizam microsserviços frequentemente implementam automação em seus processos de build e deploy, tornando a entrega de software mais confiável e menos intimidante. O objetivo é que o processo de implantação se torne "entediante", independentemente de estar lidando com uma aplicação monolítica ou com múltiplos microsserviços.

Outro aspecto importante é o design para a falha. Em uma arquitetura de microsserviços, qualquer chamada de serviço pode falhar, e as aplicações precisam ser projetadas para lidar com essas falhas de maneira resiliente. Exemplos práticos, como o Simian Army da Netflix, mostram como a indução de falhas em produção pode ser uma estratégia eficaz para testar a resiliência e o monitoramento das aplicações. A ênfase no monitoramento em tempo real é fundamental, permitindo a detecção precoce de comportamentos indesejados que possam surgir da colaboração entre serviços.

Além disso, a transparência e o monitoramento sofisticado são essenciais na gestão de microsserviços. As equipes devem implementar configurações robustas de monitoramento e registro para cada serviço, com dashboards que exibem métricas operacionais e de desempenho, garantindo que qualquer problema seja identificado rapidamente. A adoção de uma abordagem de design evolutivo também é comum entre os profissionais de microsserviços, que buscam promover uma decomposição de serviços

que permita alterações rápidas e controladas no software. Essa prática leva em conta a ideia de que componentes devem ser substituíveis e atualizáveis de forma independente.

A granularidade na liberação de serviços é uma das vantagens da arquitetura de microsserviços, permitindo que apenas os serviços alterados sejam reimplantados, o que acelera o processo de lançamento. No entanto, essa flexibilidade traz desafios, como garantir que as mudanças em um serviço não afetem negativamente seus consumidores. A preferência por evitar o versionamento de serviços, exceto como último recurso, é uma prática comum na comunidade de microsserviços.

Apesar das experiências positivas associadas à adoção de microsserviços, é importante adotar uma postura de cautela. As verdadeiras consequências das decisões arquitetônicas podem levar anos para se manifestar, e ainda existem preocupações sobre como as arquiteturas de microsserviços se comportarão a longo prazo. Questões como a dificuldade em definir limites de componentes e a complexidade das interações entre serviços são desafios que ainda precisam ser explorados.

Em suma, a arquitetura de microsserviços apresenta uma série de vantagens e desafios que devem ser cuidadosamente considerados. A abordagem é vista como uma oportunidade valiosa para empresas que buscam maior agilidade e resiliência em seus sistemas, embora a transição de uma arquitetura monolítica para microsserviços deva ser feita com cautela e planejamento. A jornada em direção à adoção de microsserviços pode ser promissora, mas exige uma reflexão contínua sobre as práticas e a evolução das arquiteturas de software no contexto empresarial.