Arquitetura Monolítica *versus* Microsserviços

Evandro Italo Silva Peixoto

Giovani Fonseca Ravagnani Disperati

Instituto Federal de Educação, Ciência e Tecnologia de São Paulo - Campus Guarulhos

Resumo

O estudo aborda as diferenças e particularidades entre as arquiteturas de software monolítica e de microsserviços, apresentando suas principais características, vantagens e desvantagens. A arquitetura monolítica concentra todo o código e funcionalidades em uma única estrutura, facilitando a implantação inicial, manutenção e integração, sendo mais indicada para projetos com menor complexidade, equipes pequenas e escalabilidade previsível. Já a arquitetura de microsserviços divide o sistema em vários serviços independentes, proporcionando maior flexibilidade, escalabilidade horizontal e autonomia de desenvolvimento, o que a torna ideal para projetos mais complexos e dinâmicos. A partir dessas premissas o trabalho compara ambas as abordagens em dois estudos de caso: a migração de sistemas da Secretaria de Estado da Tributação do Rio Grande do Norte de uma arquitetura monolítica para microsserviços e a transição do Prime Video de microsserviços para uma arquitetura monolítica. São analisados aspectos como governança, prazo de projeto, atualização, complexidade, escalabilidade, custos e integração, permitindo uma avaliação clara das circunstâncias em que cada arquitetura se destaca. Por fim, o estudo apresenta fatores que podem tornar um projeto mais aderente a uma arquitetura monolítica, bem como práticas que facilitam uma futura migração para microsserviços, caso necessário. O objetivo é fornecer uma visão abrangente e prática sobre a escolha entre essas arquiteturas, auxiliando arquitetos de software a tomarem decisões mais fundamentadas e adequadas às necessidades específicas de seus projetos.

Palavras-chave: Engenharia de Software. Arquitetura de Software. Microsserviços. Monólitos. Refatoração de arquitetura.

1. Introdução

Nos últimos anos, a evolução tecnológica tem desempenhado um papel fundamental na forma como as empresas desenvolvem e entregam seus sistemas de software. A arquitetura de software, em particular, tornou-se um campo de estudo e prática essencial para garantir a eficiência, escalabilidade e manutenção dos aplicativos.

Nesse contexto, duas abordagens têm se destacado: a arquitetura monolítica e a arquitetura de microsserviços.

A arquitetura monolítica, tradicionalmente adotada por muitas organizações, concentra todo o código e funcionalidades de um aplicativo em um único monólito. Em contrapartida, a arquitetura de microsserviços adota uma abordagem modular, na qual as

funcionalidades são divididas em pequenos serviços independentes, cada um executando uma função específica. Essa mudança de paradigma tem sido impulsionada pela busca por maior flexibilidade, escalabilidade e uma resposta mais ágil às demandas de negócios em constante transformação.

Neste trabalho, exploraremos e compararemos as duas abordagens de arquitetura de software a partir de dois estudos de caso reais, escolhidos por sua relevância e representatividade em cenários de migração entre arquiteturas. O primeiro caso examina a migração dos sistemas legados da Secretaria de Estado da Tributação do Rio Grande do Norte, que passou de uma arquitetura monolítica para microsserviços. Este estudo foi selecionado por ilustrar os desafios e benefícios de modernizar sistemas legados por meio de sua refatoração para microsserviços. O segundo caso aborda a transição do Prime Video, da Amazon, de uma arquitetura baseada em microsserviços para uma arquitetura monolítica. A escolha desse estudo se justifica por apresentar uma perspectiva contrária à tendência predominante, permitindo uma análise rica das motivações e resultados dessa decisão. Ao analisar esses dois casos, pretendemos oferecer uma visão equilibrada e detalhada sobre as implicações das migrações entre essas arquiteturas, considerando aspectos como desenvolvimento, implantação, manutenção e escalabilidade.

Além disso, destacaremos as experiências reais e os resultados alcançados, visto que compreender as nuances entre arquiteturas monolíticas e de microsserviços é essencial para o sucesso do software desenvolvido e para o negócio à qual será aplicado (PEGRUCCI; TIOSSO; REIS, 2020).

2. Materiais e Métodos

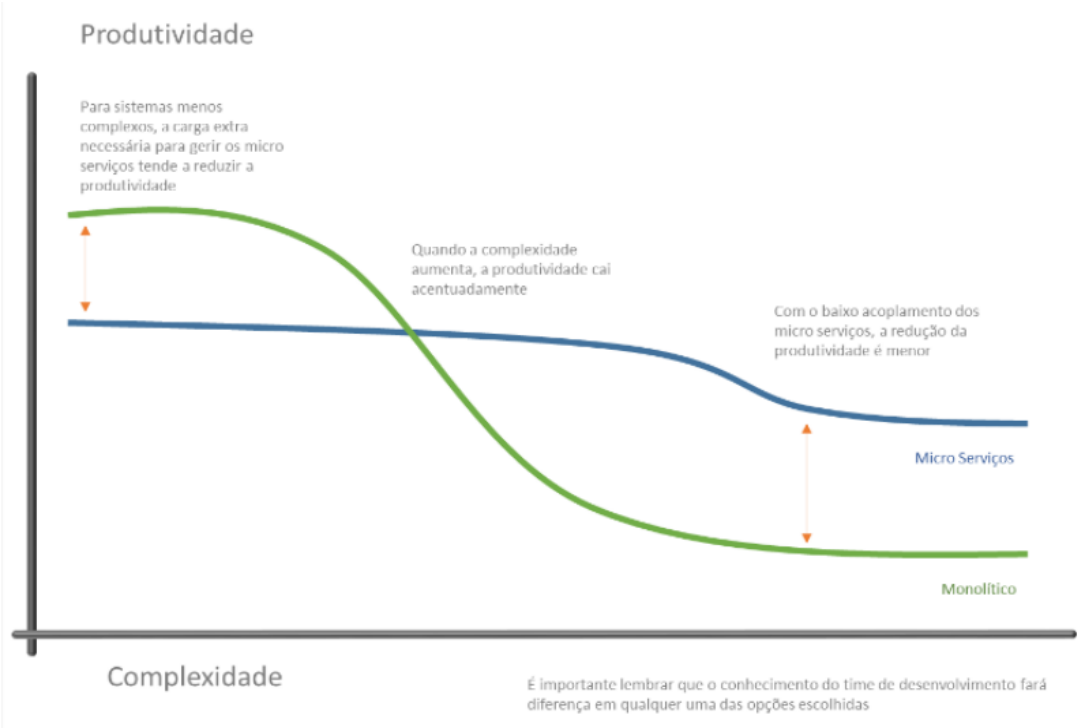
A metodologia empregada neste trabalho fundamenta-se em uma pesquisa bibliográfica com finalidade de seleção de estudos sobre migração entre arquiteturas monolítica e microsserviços. Foram selecionados casos reais que tratam sobre a estrutura de migração entre as arquiteturas, oferecendo, assim, uma fundamentação teórica para realização da comparação entre elas: os estudos selecionados foram baseados na relevância prática e na representatividade no contexto de migrações arquiteturais entre sistemas monolíticos e de microsserviços. A escolha priorizou casos reais de migração que envolvem desafios significativos de refatoração de sistemas legados e transições arquiteturais com grande impacto no desempenho e na escalabilidade dos sistemas. Nesse sentido, buscou-se abordar as nuances da mudança, a tomada de decisão para migração

para uma nova arquitetura ou rollback para a arquitetura anterior, bem como prós e contras, custos, tempo de projeto e condução da migração. Também foi levantada bibliografia específica que introduz o tema de ambas arquiteturas, a fim de contextualizar a discussão posterior e especificar as diferenças entre elas.

3. Resultados e Discussão

A definição clássica de arquitetura, apresentada por Shaw et al. (1996), afirma que a arquitetura de software define o sistema em termos de componentes computacionais e os relacionamentos entre esses componentes (VAROTO, 2002). Esse conceito é fundamental para compreender as diferenças entre a arquitetura monolítica e a de microsserviços, especialmente no que diz respeito à forma como os componentes interagem e se organizam dentro de cada abordagem. A Figura 1 ilustra essas diferenças, destacando aspectos como a produtividade e a complexidade envolvida em cada tipo de arquitetura.

Figura 1 - Comparação da arquitetura monolítica com a arquitetura de microserviços envolvendo quesitos de produtividade e complexidade

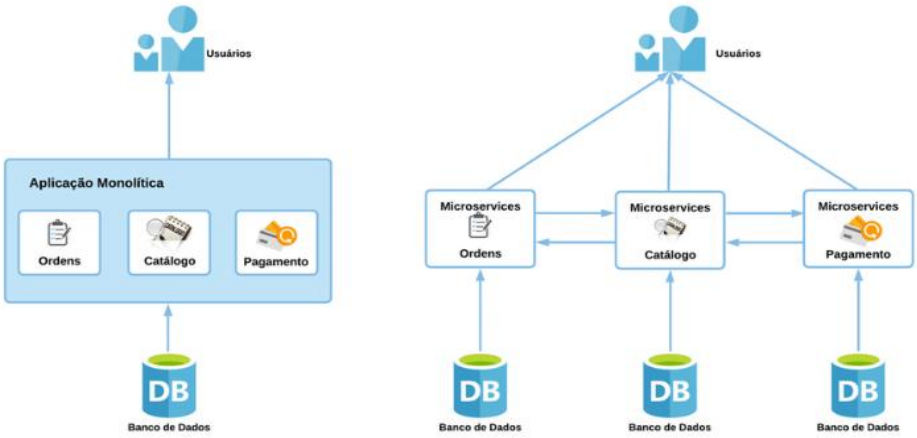


Fonte: Adaptado de FOWLER, 2014.

Embora sejam distintas, ambas as arquiteturas são eficazes na resolução de desafios específicos. A escolha entre elas deve ser cuidadosamente avaliada pelo

arquiteto de soluções, que deve determinar qual abordagem é mais adequada para o projeto em questão, ao invés de adotar uma ou outra simplesmente por influência de tendências passageiras (MENDES, 2021). A Figura 2 exemplifica essa comparação, apresentando as arquiteturas monolítica e de microsserviços lado a lado, o que facilita a visualização das diferenças estruturais e operacionais entre ambas, auxiliando na compreensão dos fatores que influenciam a escolha da solução mais adequada para cada cenário.

Figura 2 - Representação das arquiteturas monolítica e de microsserviços lado a lado.



Fonte: ECNUCOMPI, 2019.

Os resultados e discussões apresentados a seguir são baseados em uma comparação entre os estudos conduzidos por Yan de Lima Justino (2018), que examina a migração de sistemas legados da Secretaria de Estado da Tributação do Rio Grande do Norte de uma arquitetura monolítica para microsserviços, e o trabalho de Marcin Kolny, engenheiro de software da AWS, que explora a transição do Prime Video de uma arquitetura de microsserviços para uma arquitetura monolítica. Essas duas abordagens contrastantes oferecem uma perspectiva abrangente sobre os desafios e benefícios associados a cada tipo de arquitetura.

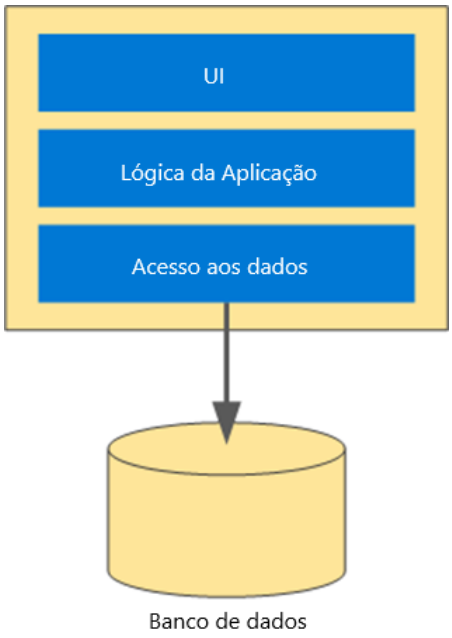
3.1 Arquitetura Monolítica

A arquitetura monolítica, relativa ao monólito, refere-se ao modelo de desenvolvimento em que uma aplicação é construída dentro de uma única estrutura executável. Nesse modelo, o resultado final é uma entidade única, compacta, indivisível

e impenetrável, que será posteriormente executada (LUCIO et al., 2017).

A arquitetura monolítica é amplamente utilizada em softwares corporativos desde os primeiros dias de implementação de aplicativos para a Web. Nessa abordagem, todos os componentes do lado do servidor são agrupados em uma única unidade. Muitos aplicativos desenvolvidos em linguagens como Java, Ruby e até C++ consistem em um único arquivo (RICHARDSON, 2014). A Figura 3 ilustra a estrutura de uma arquitetura monolítica, destacando a integração de todos os componentes em uma única entidade funcional, o que reflete a simplicidade de sua organização, mas também evidencia as limitações quanto à escalabilidade e flexibilidade em cenários mais complexos.

Figura 3 - Representação de arquitetura monolítica



Fonte: Adaptado de Nallainathan, 2013.

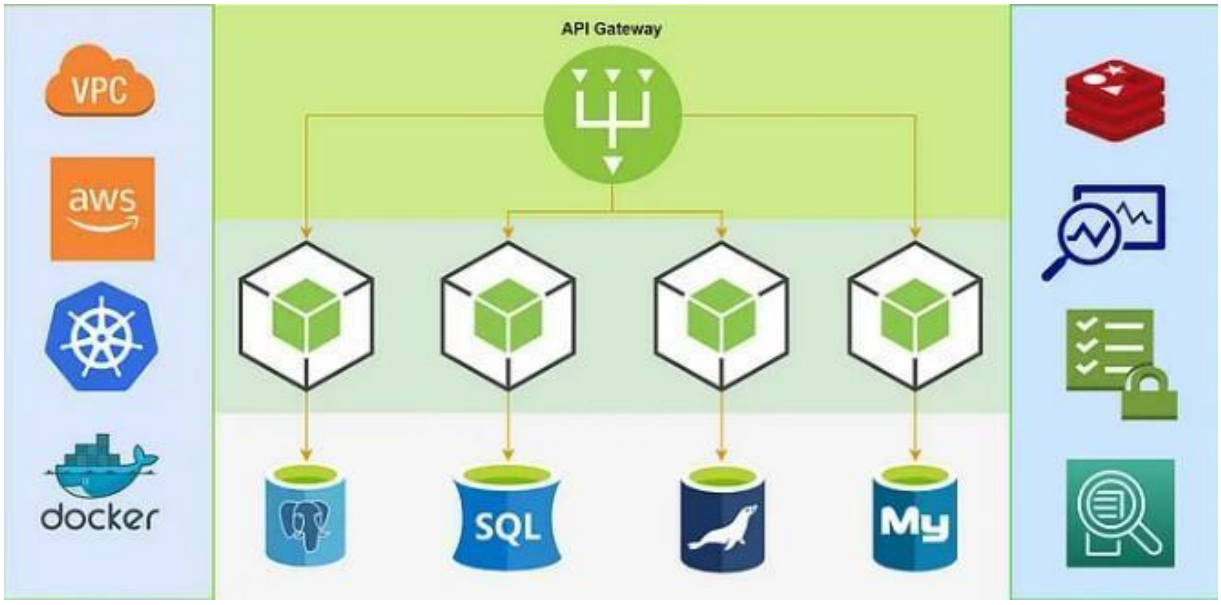
3.2 Arquitetura de Microsserviços

De acordo com Lewis e Fowler (2014), o termo "Arquitetura de Microsserviços" surgiu nos últimos anos com o objetivo de atender a aplicações de software independentes, sem a necessidade de dependência de outras aplicações. De forma geral, essa arquitetura é caracterizada pela divisão em vários projetos autônomos, comumente chamados de serviços.

A arquitetura de microsserviços é uma alternativa às arquiteturas monolíticas no campo da Engenharia de Software, onde cada parte que compõe as funcionalidades do sistema é um microsserviço independente. Essa abordagem permite que cada microsserviço seja desenvolvido com os recursos e linguagens que melhor se adequam ao seu contexto específico (KANIZAWA; PINTO, 2022).

Os microsserviços oferecem maior liberdade para reagir e tomar decisões de maneira mais ágil, permitindo que as aplicações acompanhem as constantes mudanças de requisitos (NEWMAN, 2015). A Figura 4 ilustra essa arquitetura, mostrando como os microsserviços são organizados de forma independente, o que permite que cada serviço funcione de maneira autônoma, facilitando a escalabilidade, a flexibilidade no desenvolvimento e a manutenção contínua, fatores que são essenciais para lidar com requisitos dinâmicos.

Figura 4 - Representação de arquitetura em Microsserviços



Fonte: KIPRONO, Ian. 2023.

3.3 Comparativo entre as arquiteturas nos cases selecionados

O estudo comparativo entre as arquiteturas monolítica e de microsserviços revela diversas implicações relevantes para a tomada de decisão em projetos de software. Cada uma dessas abordagens possui características próprias que impactam diretamente diferentes aspectos de projetos de software, tais como: governança, prazos de projeto, flexibilidade em atualizações, complexidade de desenvolvimento e manutenção, escalabilidade, custos e integração. Em relação à governança, a arquitetura monolítica se destaca pela padronização e controle centralizado, o que facilita a manutenção e centralização das operações. Por outro lado, a arquitetura de microsserviços permite a descentralização dos componentes, maior variedade tecnológica e autonomia das equipes, sendo cada serviço independente e com ciclo de vida próprio. Isso oferece maior flexibilidade, mas exige uma governança mais robusta para lidar com a autonomia das equipes e a independência dos serviços. Esses aspectos são discutidos no trabalho de Justino (2018), ao abordar a reorganização da estrutura de serviços da Secretaria de Estado da Tributação do Rio Grande do Norte (SET), que favoreceu a entrega contínua e o suporte a partir do uso, evidenciando a importância de uma governança eficiente na arquitetura de microsserviços. Ao se comparar com o caso do Prime Video, Kolny (2023) discute que a governança da arquitetura de microsserviços, embora ofereça descentralização e maior autonomia, trouxe desafios na coordenação e na gestão da

escalabilidade, que acabaram por não justificar o uso da arquitetura distribuída. Nesse contexto, pode se afirmar que a decisão de migrar de uma arquitetura de microsserviços para uma monolítica melhorou a governança ao centralizar os processos, simplificando o controle e reduzindo os custos operacionais.

No que tange ao prazo de projeto, a arquitetura monolítica proporciona um desenvolvimento inicial mais rápido, devido à sua simplicidade de integração e implantação, além de apresentar menor complexidade. Em contraste, a arquitetura de microsserviços envolve um desenvolvimento mais distribuído e uma implantação gradual, resultando em um ciclo de vida descentralizado. Embora isso permita maior flexibilidade, demanda um planejamento e coordenação mais complexos. Justino (2018) discute esse aspecto no processo de reengenharia de software para modernização de sistemas legados, destacando que a migração para microsserviços exigiu um planejamento criterioso, com prazos definidos e entregas iterativas. Kolny (2023), por outro lado, também destaca no caso do Prime Video que, embora a arquitetura distribuída fosse inicialmente vista como uma forma de escalar rapidamente, a complexidade de coordenação entre os diversos microsserviços resultou em gargalos e prazos de projeto mais longos do que o previsto, levando à decisão de adotar um monólito.

Quanto às atualizações, a arquitetura monolítica tende a ser menos flexível, já que uma atualização geralmente implica em testes completos e uma maior coordenação, o que pode ser uma desvantagem em projetos que exigem mudanças frequentes. Por outro lado, os microsserviços possibilitam atualizações graduais e específicas, permitindo maior agilidade nas entregas e facilitando o gerenciamento de mudanças. No entanto, essa abordagem requer uma infraestrutura mais complexa para garantir a integridade dos sistemas. Justino (2018) discute essa questão no contexto da migração do sistema UVT, onde a Secretaria de Estado da Tributação do Rio Grande do Norte (SET) enfrentava dificuldades com a arquitetura monolítica, que exigia a recompilação do sistema inteiro para pequenas atualizações. A migração para microsserviços, por sua vez, permitiu maior flexibilidade, com a adoção de atualizações incrementais, proporcionando um gerenciamento mais eficiente das mudanças. No entanto, o caso do Prime Video demonstrou que, embora as atualizações em microsserviços fossem mais granulares, o alto custo de coordenação e orquestração de serviços distribuídos resultou em dificuldades que foram mitigadas após a migração para uma arquitetura monolítica.

A complexidade também varia consideravelmente entre as duas arquiteturas. A arquitetura monolítica é menos complexa em termos de desenvolvimento e manutenção,

o que a torna mais acessível para projetos menores ou com menos exigências de escalabilidade. Entretanto, é menos flexível quando comparada aos microsserviços, que, apesar de serem mais complexos em sua implementação, oferecem maior resiliência e escalabilidade, características ideais para sistemas de grande porte. Justino (2018) destaca que a arquitetura monolítica, embora seja mais simples, apresenta limitações em termos de escalabilidade e flexibilidade. Em contrapartida, os microsserviços, apesar de demandarem uma implementação mais complexa, proporcionam maior autonomia e resiliência, como evidenciado no trabalho sobre a reengenharia do sistema UVT. No caso do Prime Video, Kolny (2023) destaca que a complexidade da arquitetura distribuída tornou-se um fator crítico para a decisão de migrar para uma arquitetura monolítica, já que a manutenção de múltiplos serviços independentes impunha um custo operacional elevado e um risco adicional em termos de coordenação entre serviços.

No quesito escalabilidade, a arquitetura monolítica é mais adequada para projetos que requerem escalabilidade vertical, ou seja, o aumento de recursos em um único servidor. Já os microsserviços permitem a escalabilidade horizontal, em que cada serviço pode ser escalado independentemente, oferecendo maior flexibilidade e resiliência. Esse tipo de escalabilidade é particularmente útil em sistemas que exigem alta disponibilidade e distribuição. No estudo de Justino (2018), o sistema analisado enfrentou desafios relacionados à escalabilidade na arquitetura monolítica, uma vez que foi construído para operar em uma única thread de processamento, o que limitava sua capacidade de crescimento. A migração para microsserviços, por outro lado, possibilitou a adoção de escalabilidade horizontal, proporcionando melhor desempenho e maior resiliência ao sistema. Kolny (2023), no entanto, descreve um caso diferente: embora a escalabilidade horizontal fosse um dos objetivos com os microsserviços, a alta carga de orquestração e os custos de armazenamento e processamento distribuído limitaram a capacidade de escalar o Prime Video, o que foi resolvido com a adoção de uma arquitetura monolítica mais centralizada.

Em termos de custos, a arquitetura monolítica apresenta custos iniciais mais baixos, uma vez que sua infraestrutura é mais simples e o desenvolvimento menos complexo. No entanto, à medida que o sistema cresce, os custos de manutenção podem aumentar significativamente devido à rigidez da arquitetura. Já os microsserviços, embora apresentem um custo inicial mais elevado devido à necessidade de uma infraestrutura distribuída e equipes multidisciplinares, tendem a ser mais eficientes a longo prazo em termos de manutenção e escalabilidade. Justino (2018) menciona que, apesar dos maiores

custos iniciais associados à adoção de microsserviços, essa abordagem se justifica em projetos que exigem maior escalabilidade e flexibilidade, resultando em uma manutenção mais eficiente e melhor alocação de recursos ao longo do tempo. No entanto, o caso do Prime Video mostrou que os custos iniciais de operação e escalabilidade da arquitetura de microsserviços eram significativamente maiores do que o esperado, levando à migração para uma arquitetura monolítica, que reduziu os custos em mais de 90%, conforme relatado por Kolny (2023).

Finalmente, no quesito integração, a arquitetura monolítica se destaca por sua simplicidade, pois todo o sistema utiliza a mesma tecnologia e um banco de dados centralizado, o que facilita a integração externa. Em contraste, os microsserviços, apesar de oferecerem maior flexibilidade, demandam um cuidado maior na integração, já que falhas podem comprometer a integridade dos dados e a consistência dos serviços. Para mitigar esses riscos, a arquitetura de microsserviços precisa ser bem projetada e acompanhada por mecanismos robustos de monitoramento e gestão. Justino (2018) ressalta que a arquitetura monolítica, por ser centralizada, facilita a integração entre os componentes do sistema, entretanto o autor ressalta que foi necessário adotar novas práticas para garantir a comunicação eficiente entre os serviços, assegurando a integridade dos dados e a consistência do sistema UVT. No caso do Prime Video, a complexidade de integrar múltiplos microsserviços, que envolvia diversas chamadas entre sistemas e armazenamento intermediário em S3, foi um dos maiores fatores de custo e ineficiência, o que motivou a migração para um monólito que integrava os componentes de forma mais direta e eficiente, eliminando custos operacionais adicionais e simplificando o fluxo de dados, conforme relatado por Kolny (2023).

3.4 Fatores de aderência à arquitetura monolítica em detrimento à arquitetura de microsserviços

A partir dos estudos analisados, é possível extrapolar algumas considerações gerais. A arquitetura monolítica, embora muitas vezes vista como ultrapassada frente às soluções mais modernas, como os microsserviços, apresenta diversas vantagens que justificam sua adoção em determinados cenários. Justino (2018) e Kolny (2023) discutem exemplos em que a arquitetura monolítica se mostra mais adequada, sobretudo em projetos que demandam simplicidade, custos controlados e uma governança centralizada.

Quando o projeto é relativamente simples e não requer a separação de

componentes complexos e independentes, a arquitetura monolítica oferece uma solução prática e eficiente. Por exemplo, ao construir um aplicativo para gerir um pequeno negócio local (ERP), todos os módulos (estoque, financeiro, compras, custos, contabilidade) devem ser interligados. Assim, usualmente não há necessidade de separar serviços específicos, o que faz com que uma arquitetura monolítica seja mais simples de manter e implantar. Ainda nesse sentido, Kolny (2023) observa que, no caso do Prime Video, a simplificação proporcionada pela migração para um monólito trouxe ganhos significativos na redução da complexidade do sistema, o que corrobora essa visão mesmo em sistemas de maior complexidade.

Além disso, em projetos conduzidos por equipes pequenas, a centralização das operações, típica da arquitetura monolítica, facilita a comunicação e a coordenação entre os membros, como observado por Justino (2018). Ou seja, com menos pessoas e equipes envolvidas e menos partes móveis, a coordenação e a comunicação podem ser mais simples. Por exemplo, em uma pequena startup ou softwarehouse com apenas alguns membros e sem equipes com responsabilidades claramente distintas em manutenção de módulos diferentes do sistema, pode ser mais eficiente construir um único aplicativo monolítico em vez de dividir em vários serviços. O caso do Prime Video também exemplifica esse ponto, com Kolny (2023) ressaltando que a equipe conseguiu operar de forma mais eficiente após a migração para um monólito, eliminando a necessidade de gerenciar múltiplos microsserviços distribuídos.

Outro aspecto relevante é o desempenho. Em cenários onde a comunicação entre componentes exige baixa latência, a arquitetura monolítica pode ser mais adequada. Isso ocorre especialmente quando a comunicação entre os componentes é intensiva. Por exemplo, se está sedo construído um aplicativo de jogos online em tempo real a comunicação entre jogadores deve ser rápida e precisa, logo uma arquitetura monolítica pode ser mais adequada do que microsserviços. Isso se dá pois microsserviços podem ter overhead de comunicação e aumentar a latência, tal como relatado no caso do Prime Video, onde Kolny (2023) mostra que a latência gerada pelos microsserviços distribuídos se mostrou um obstáculo significativo, sendo mitigada pela adoção de uma arquitetura monolítica, que reduziu os atrasos na comunicação entre os componentes do sistema.

A falta de experiência da equipe com microsserviços também pode justificar a escolha por uma arquitetura monolítica. Justino (2018) observa que, em contextos onde a equipe não tem familiaridade com essa arquitetura, optar por um monólito pode reduzir os riscos de implementação e a curva de aprendizado. Kolny (2023) reforça essa ideia ao

relatar que a simplificação proporcionada pela centralização da arquitetura no Prime Video eliminou a complexidade adicional trazida pelos microsserviços, resultando em uma operação mais simples e eficiente.

Nos casos em que o ciclo de vida do projeto é curto ou os prazos são apertados, a arquitetura monolítica pode ser mais vantajosa por permitir um desenvolvimento mais rápido e simplificado. Kolny (2023) destaca que a migração para um monólito no Prime Video simplificou os processos, possibilitando uma entrega mais rápida sem a necessidade de gerenciar múltiplos serviços independentes.

Quanto à escalabilidade, se os requisitos forem previsíveis e não houver necessidade de escalabilidade granular dos componentes, a arquitetura monolítica pode ser suficiente. Kolny (2023) relata que, no Prime Video, a escalabilidade vertical proporcionada pelo monólito atendeu de forma satisfatória à crescente demanda, sem a necessidade de uma infraestrutura distribuída.

No quesito integração, a arquitetura monolítica também apresenta pontos fortes, sobretudo em sistemas que não requerem muitas integrações externas. Kolny (2023) relata que, no Prime Video, a complexidade gerada pelas múltiplas integrações entre microsserviços foi eliminada com a centralização do sistema, o que simplificou a comunicação entre componentes e reduziu os custos operacionais.

Por fim, os custos de infraestrutura podem ser significativamente menores em uma arquitetura monolítica, especialmente quando não há necessidade de escalabilidade dinâmica ou balanceamento de carga. Justino (2018) aponta que essa foi uma escolha vantajosa para os sistemas da SET. Contudo, Kolny (2023) demonstra que a migração para um monólito no Prime Video resultou em uma redução de mais de 90% nos custos de infraestrutura, tornando-se uma solução econômica e eficiente.

Esses fatores indicam que a escolha da arquitetura monolítica, apesar de sua simplicidade, pode oferecer vantagens consideráveis em cenários específicos, principalmente quando a simplicidade, a redução de custos e a centralização das operações são os principais requisitos do projeto.

3.5 Estruturação de sistemas monolíticas: formas de se mitigar o impacto em futura migração de arquitetura para microsserviços

Após a análise comparativa entre as arquiteturas monolítica e de microsserviços, fica claro que a escolha entre essas duas abordagens depende do contexto específico de

cada projeto, levando em conta fatores como governança, prazos de entrega, escalabilidade, flexibilidade e custos operacionais. Embora os microsserviços ofereçam maior autonomia e escalabilidade, a arquitetura monolítica continua sendo uma escolha eficiente em cenários de menor complexidade e com maior controle centralizado. No entanto, em alguns casos, é necessário adotar a arquitetura monolítica com uma visão de longo prazo, levando em consideração a possibilidade de migração para microsserviços no futuro. Essa transição, embora benéfica em muitas situações, pode acarretar desafios se não for bem planejada desde o início do projeto. Assim, é importante estruturar o sistema monolítico de modo a facilitar uma eventual migração, minimizando os impactos e permitindo uma transição mais eficiente. Dessa forma, Gomes (2021) discute algumas algumas práticas e estratégias que podem ser implementadas no desenvolvimento de sistemas monolíticos com vistas a uma futura migração para a arquitetura de microsserviço, também podendo ser tidas como etapas intermediárias em uma refatoração de um monólito para microsserviços.

Ao projetar um sistema monolítico, com a perspectiva de uma futura migração para uma arquitetura de microsserviços, é fundamental adotar práticas que facilitem essa transição, mitigando os impactos que podem surgir no processo. A separação adequada das responsabilidades dentro do sistema é uma das principais estratégias a serem seguidas. O ideal é dividir o código em módulos claramente definidos, cada um responsável por uma tarefa específica. Essa abordagem, conhecida como "separação de preocupações" (*separation of concerns*), facilita a identificação de serviços que podem ser extraídos e independizados no futuro (GOMES, 2021).

Além disso, é importante estabelecer interfaces claras e contratos bem definidos entre os diferentes componentes do sistema monolítico. Isso significa que os componentes devem interagir de forma previsível e padronizada, o que garante a independência entre eles. Essa estrutura facilita, futuramente, a substituição ou extração de componentes quando a migração para microsserviços se tornar necessária (GOMES, 2021).

O uso de padrões de design flexíveis, como a injeção de dependência, também é crucial. Esse padrão ajuda a reduzir as dependências rígidas entre os componentes, permitindo que cada parte do sistema seja facilmente substituída ou movida. Isso proporciona uma flexibilidade essencial para a evolução do sistema, especialmente em uma arquitetura distribuída (GOMES, 2021).

Outra prática importante é a implementação de testes abrangentes e automatizados no sistema monolítico. Esses testes são fundamentais para garantir a estabilidade do

sistema durante o processo de migração, pois ajudam a identificar problemas rapidamente e a manter a qualidade do software ao longo do tempo (GOMES, 2021).

A decomposição por funcionalidade é outro ponto importante. A arquitetura deve ser pensada de maneira a permitir que, futuramente, as funcionalidades sejam extraídas como serviços independentes, sem a necessidade de grandes alterações no sistema monolítico original. Isso facilita a divisão do sistema em partes menores e mais gerenciáveis, o que é um dos principais objetivos de uma arquitetura de microsserviços (GOMES, 2021).

Os padrões de comunicação entre os componentes do sistema também devem ser planejados com cuidado. Utilizar mensagens assíncronas, por exemplo, pode ajudar a reduzir as dependências diretas entre os componentes e facilitar sua extração no futuro. Esse tipo de abordagem garante que os componentes possam se comunicar de forma eficiente, mesmo quando forem transformados em serviços independentes (GOMES, 2021).

No que diz respeito ao gerenciamento de dados, manter uma abordagem centralizada no sistema monolítico pode ser útil no estágio inicial, mas é importante planejar a migração desses dados para uma arquitetura distribuída no futuro. Centralizar o gerenciamento de dados pode simplificar a transição inicial, mas ao migrar para microsserviços, a distribuição adequada dos dados entre os serviços independentes será fundamental para manter a integridade e o desempenho do sistema (GOMES, 2021).

Por fim, é essencial implementar ferramentas de monitoramento e análise de métricas no sistema monolítico. Essas ferramentas permitem uma compreensão detalhada do comportamento do sistema, ajudando a identificar pontos críticos que podem ser beneficiados por uma arquitetura de microsserviços. O monitoramento contínuo proporciona insights valiosos que podem orientar o processo de migração, garantindo que a transição seja feita de maneira eficiente e segura (GOMES, 2021).

Ao seguir essas práticas, a arquitetura monolítica pode ser estruturada de forma a reduzir os desafios de uma futura migração para microsserviços, permitindo que o sistema evolua e se adapte às necessidades do negócio ao longo do tempo, com maior facilidade e escalabilidade (GOMES, 2021).

4. Considerações Finais

Ao analisar as duas abordagens arquiteturais em questão – a monolítica e a de

microsserviços –, torna-se evidente que ambas possuem aplicações específicas, cujos benefícios e desafios devem ser cuidadosamente avaliados de acordo com o contexto do projeto de software. A arquitetura monolítica, ainda que tradicional e frequentemente associada a uma implementação mais simplificada e de menor custo inicial, enfrenta limitações no que diz respeito à escalabilidade e à flexibilidade, especialmente em sistemas que exigem alta resiliência e rápidas respostas a mudanças. Por outro lado, a arquitetura de microsserviços oferece uma solução modular e mais flexível, ideal para projetos dinâmicos e de grande porte, embora demande uma gestão mais complexa e uma infraestrutura distribuída que acarreta custos iniciais mais elevados.

Nos casos analisados, observou-se que a transição de uma arquitetura para outra não é uma decisão trivial e requer uma análise aprofundada das necessidades e das capacidades do projeto. A migração da Secretaria de Estado da Tributação do Rio Grande do Norte para uma arquitetura de microsserviços exemplifica os desafios de refatoração de sistemas legados, demonstrando a importância de uma arquitetura mais flexível em contextos onde a demanda por escalabilidade e atualizações constantes é elevada. Em contrapartida, o caso do Prime Video mostra que a adoção de uma arquitetura monolítica pode ser mais vantajosa em determinados cenários, especialmente quando o foco está na simplificação do sistema e na redução de custos operacionais, como demonstrado pela economia significativa alcançada com a centralização dos processos.

Não existe uma abordagem única e universalmente superior: os casos estudados demonstram que a escolha da arquitetura deve ser pautada por uma análise criteriosa dos requisitos do projeto e das expectativas a longo prazo e a decisão deve levar em conta não apenas os aspectos técnicos, mas também as necessidades de negócio, a estrutura da equipe e os recursos disponíveis como o tamanho da equipe, a necessidade de escalabilidade, os requisitos de manutenção e os custos operacionais.

Dessa forma, a arquitetura monolítica continua a ser uma opção válida para sistemas menos complexos ou com necessidades de escalabilidade limitadas, sendo uma escolha segura para projetos menores e com escopo mais definido, enquanto os microsserviços se justificam em projetos de grande porte, onde a flexibilidade e a escalabilidade são primordiais e cujos requisitos estão em constante evolução.

O papel do arquiteto de software, nesse contexto, é fundamental para avaliar, caso a caso, a abordagem que melhor atenderá aos requisitos técnicos e de negócios do projeto, evitando escolhas baseadas apenas em tendências ou no "hype" tecnológico. A adequação da arquitetura às demandas do projeto é, sem dúvida, um fator determinante para o

sucesso da implementação. Nesse sentido esse estudo fornece um ponto de partida para essas considerações e sugere-se que estudos posteriores possam incorporar essa análise para futuras investigações sobre modelos híbridos de arquitetura e formas de automação de migrações arquiteturais.

5. Referências

GOMES, Vinicius. **Técnicas para migrar o seu sistema monolítico para microsserviços**. Softplan, 18 out. 2021. Disponível em: https://www.softplan.com.br/tech-writers/tecnicas-para-migrar-o-seu-sistema-monolitico-para-microsservicos/. Acesso em: 18 out. 2023.

JUSTINO, Yan de Lima. **Do monolito aos microsserviços: um relato de migração de sistemas legados da Secretaria de Estado da Tributação do Rio Grande do Norte**. 2018. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2018.

KANIZAWA, Daniel Toma; PINTO, Giovani Sacramento. **Arquitetura de microsserviços**. *Revista Interface Tecnológica*, v. 19, n. 2, p. 308–318, 2022.

LEWIS, James; FOWLER, Martin. **Microservices: a definition of this new architectural term**. 2014. Disponível em: http://martinfowler.com/articles/microservices.html. Acesso em: 26 dez. 2014.

LUCIO, João Paulo Dias et al. **Análise comparativa entre arquitetura monolítica e de microsserviços**. Florianópolis, SC: [s.n.], 2017.

MENDES, Igor Silva. **Arquitetura monolítica vs microsserviços: uma análise comparativa**. 2021.

NEWMAN, Sam. **Building Microservices**. O'Reilly Media, Inc., 2015.

PEGRUCCI, Mateus; TIOSSO, Felipe; REIS, Henrique Morais. **Análise de um estudo de caso para comparar as arquiteturas monolítica e de microserviços**. *Revista Interface Tecnológica*, v. 17, n. 2, p. 325–337, 2020.

RICHARDSON, Chris. **Pattern: Microservices Architecture**. *Microservices.io*, 2014.

VAROTO, Aline Cristina. **Visões em arquitetura de software**. Tese (Doutorado) — Universidade de São Paulo, 2002.

KOLNY, Marcin. Scaling up the Prime Video audio/video monitoring service and reducing costs by 90%. Prime Video Tech, 22 mar. 2023. Disponível em: https://www.aboutamazon.com/what-we-do/entertainment?utm_source=primevideotech&utm_medium=primevideotech/ . Acesso em: 18 out. 2023.