UNIVERSIDADE NOVE DE JULHO – UNINOVE

PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA E GESTÃO DO CONHECIMENTO

WILIANS DOUGLAS CONDE SOUZA

GUIA DE ORIENTAÇÕES NA MIGRAÇÃO DE SISTEMAS DE INFORMAÇÃO DO AMBIENTE MONOLÍTICO PARA AMBIENTE DE MICROSSERVIÇOS NA COMPUTAÇÃO EM NUVEM EM MICROEMPRESAS DE *SOFTWARE*

São Paulo

2023

WILIANS DOUGLAS CONDE SOUZA

GUIA DE ORIENTAÇÕES NA MIGRAÇÃO DE SISTEMAS DE INFORMAÇÃO DO AMBIENTE MONOLÍTICO PARA AMBIENTE DE MICROSSERVIÇOS NA COMPUTAÇÃO EM NUVEM EM MICROEMPRESAS DE *SOFTWARE*

Dissertação de mestrado apresentada ao Programa de Pós-graduação em Informática e Gestão do Conhecimento da Universidade Nove de Julho – UNINOVE, como requisito parcial para a obtenção do grau de Mestre em Informática e Gestão do Conhecimento.

Prof. Dr. Ivanir Costa

São Paulo

2023

Souza, Wilians Douglas Conde.
Guia de orientações na migração de sistemas de informação do ambiente monolítico para ambiente de microsserviços na computação em nuvem em microempresas de software. / Wilians Douglas Conde Souza. 2023.
239 f.
Dissertação (Mestrado) - Universidade Nove de Julho - UNINOVE, São Paulo, 2023.
Orientador(a): Prof. Dr. Ivanir Costa.
1. Arquitetura de software. 2. Arquitetura Monolítica. 3. Microsserviços. 4. Computação em nuvem. 5. Microempresa de Software.
I. Costa, Ivanir. II. Título.
CDU 004

WILIANS DOUGLAS CONDE SOUZA

GUIA DE ORIENTAÇÕES NA MIGRAÇÃO DE SISTEMAS DE INFORMAÇÃO DO AMBIENTE MONOLÍTICO PARA AMBIENTE DE MICROSSERVIÇOS NA COMPUTAÇÃO EM NUVEM EM MICROEMPRESAS DE *SOFTWARE*

Dissertação de mestrado apresentada ao Programa de Pós-graduação em Informática e Gestão do Conhecimento, da Universidade Nove de Julho – UNINOVE, como obtenção do grau de Mestre em Informática e Gestão do Conhecimento, pela Banca Examinadora, formada por:

São Paulo, 30 de maio de 2023

_____

Prof. Dr. Ivanir Costa – Orientador Uninove

_____

Membro externo Prof. Dr. Fabio Kazuo Ohashi – Mackenzie

_____

Membro interno Prof. Dr. Renato Jose Sassi – Uninove

São Paulo

2023

Dedico esta dissertação a duas pessoas especiais que fizeram parte da minha vida e que, infelizmente, não estão mais presentes fisicamente, mas estarão sempre em meu coração: Juarez Ferreira de Souza, o tio Jú, sinônimo de doçura, que sempre me inspirou com sua gentileza e sabedoria; e Rafael Conde, conhecido carinhosamente como Zolhos, que representava a resiliência em sua essência e me ensinou lições valiosas.

A ambos, meu eterno agradecimento pelas inúmeras contribuições que fizeram em minha jornada. Seus ensinamentos e exemplo continuam vivos em minha memória. Que suas lembranças e influências permaneçam como guias em minha caminhada, reafirmando o quanto foram essenciais para o meu crescimento.

AGRADECIMENTOS

Agradeço à minha companheira de vida, Mônica, e ao meu filho, Pedro, o apoio e o incentivo ao longo desta jornada. Aos meus pais, Ilson e Maria, o apoio em todas as etapas da minha vida.

Ao meu orientador, Prof. Dr. Ivanir Costa, agradeço a orientação e contribuições neste trabalho.

Ao Prof. Dr. Marcos Antonio Gaspar, o tempo investido em comentários que aprimoraram as ideias neste trabalho.

Aos Professores Dr. Renato José Sassi, Dr. Aguinaldo Aragon Fernandes e Dr. Fabio Kazuo Ohashi, a participação nas bancas de qualificação e defesa, e os valiosos comentários e direcionamentos.

Ao meu colega e amigo de programa, Breno Petrili, por compartilhar responsabilidades e apoiar-me ao longo desta jornada.

Agradeço também a Carolina Lemos Rodrigues por me apoiar nos primeiros passos em direção à academia. Sua ajuda foi fundamental.

Aos meus amigos/irmãos Bruno Conde, por sua visão horizontal, ampla e vasta, e Lucas Marques, por sua visão vertical, profunda e detalhista. Suas perspectivas únicas e contribuições foram fundamentais para enriquecer este trabalho.

RESUMO

A crescente utilização de sistemas de informação, cada vez mais abrangentes, exige que todos os tipos de informação estejam disponíveis em qualquer local e a qualquer momento para o funcionamento eficaz das organizações em todas as suas estruturas. Nesse cenário, o mercado de *software* de aplicações corporativas executadas em ambiente de Computação em Nuvem tem superado o de *software* em computadores locais e centralizados. No entanto, as microempresas de *software*, que representam 48% do mercado brasileiro, enfrentam restrições financeiras, falta de pessoal qualificado, resiliência e flexibilidade, dificultando a adoção correta das novas arquiteturas de *software*, como os microsserviços em Computação em Nuvem. Diante desse contexto, o presente trabalho teve como objetivo desenvolver um guia de orientações para apoiar microempresas de *software* brasileiras na migração de sistemas de informação desenvolvidos em arquitetura monolítica para a arquitetura baseada em microsserviços em Computação em Nuvem. Como sustentação teórica, o estudo aplicou métodos científicos como a revisão sistemática da literatura e a pesquisa de campo, utilizando um questionário como instrumento. As principais contribuições deste estudo são a elaboração de um guia com orientações extraídas da literatura e selecionadas por meio da aplicação da Curva ABC, e a validação dessas orientações por especialistas por meio do método *survey* controlado Delphi. Ao apresentar as restrições enfrentadas pelas microempresas de *software* e fornecer orientações práticas, espera-se que este guia facilite a adoção da arquitetura de microsserviços em Computação em Nuvem por essas empresas, permitindo-lhes evoluir e se adaptar às demandas do mercado.

Palavras-chave: Arquitetura de *software*. Arquitetura monolítica. Microsserviços. Computação em Nuvem. Microempresa de *software*.

ABSTRACT

The increasing use of comprehensive information systems demands that all types of information be available anywhere and anytime for effective functioning of organizations across their structures. In this scenario, the market for cloud-based enterprise application software has surpassed that of software installed on local and centralized computers. However, software microenterprises, which represent 48% of the Brazilian market, face financial constraints, lack of qualified personnel, resilience, and flexibility, making it difficult to correctly adopt new software architectures such as microservices in cloud computing. In this context, this study aimed to develop a guidance framework to support Brazilian software microenterprises in migrating information systems developed in monolithic architecture to microservices-based architecture in cloud computing. The study applied scientific methods such as systematic literature review and field research, using a questionnaire as an instrument. The main contributions of this study are the development of a guidance framework with directions extracted from the literature and selected through the application of the ABC Curve, and the validation of these directions by experts through the controlled Delphi survey method. By presenting the constraints faced by software microenterprises and providing practical guidance, it is expected that this framework will facilitate the adoption of microservices architecture in cloud computing by these companies, enabling them to evolve and adapt to market demands.

Keywords: Software architecture. Monolithic architecture. Microservices. Cloud Computing. Software SME.

LISTA DE ILUSTRAÇÕES

FIGURAS

QUADROS

LISTA DE TABELAS

LISTA DE ABREVIATURAS

ESB – *Enterprise Services Bus*

HTTP – *Hypertext Transfer Protocol*

ISO – *International Organization for Standardization*

JSON – *JavaScript Object Notation*

MEs – Microempresas

MSA – *Microservices Architecture*

MVC – *Model-View-Controller*

REST – *Representational State Transfer*

RPC – *Remote Procedure Call*

SaaS – *Software as a Service*

SI – Sistema de Informação

SOA – *Service Oriented Architecture*

SOAP – *Simple Object Access Protocol*

UML – *Unified Modeling Language*

XML – *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Nesta seção, são apresentados a contextualização, o problema e as questões de pesquisa, os objetivos, a justificativa, a delimitação do tema e a estrutura deste trabalho.

1.1 CONTEXTUALIZAÇÃO

As micro e pequenas empresas desempenham um papel fundamental na economia brasileira, contribuindo com mais de um quarto do Produto Interno Bruto (PIB). Com cerca de 9 milhões de micro e pequenas empresas no país, elas correspondem a 27% do PIB, além de serem responsáveis por 52% dos empregos formais e 40% dos salários pagos (SEBRAE, 2022).

No setor de desenvolvimento de *software* brasileiro, há 7.642 empresas, com cerca de 95% delas classificadas como micro e pequenas empresas, sendo 48% microempresas – MEs (com menos de 10 funcionários) e 47% pequenas empresas (de 10 a 99 funcionários) (ABES, 2023).

No que se refere aos Sistemas de Informação (SIs), os sistemas corporativos são responsáveis por gerenciar informações e possibilitar visões estratégicas, por exemplo, *Enterprise Resource Planning* – ERP e *Customer Relationship Management* – CRM (MOTIWALLA; THOMPSON, 2012).

Os sistemas corporativos são normalmente desenvolvidos com arquiteturas monolíticas, em que todo o contexto do sistema se encontra em uma única estrutura de códigos. Nessa solução, considerada tradicional, os sistemas tendem a ser difíceis e custosos de serem modificados, já que qualquer mudança, por menor que seja, irá impactar outros recursos, sendo necessário aplicar um grande esforço no processo de validação para garantir a integridade da solução (SAVCHENKO; RADCHENKO; TAIPALE, 2015).

Em contrapartida, a arquitetura baseada em microsserviços é um *design* arquitetural que decompõe as funcionalidades de um SI em pequenos serviços autônomos que trabalham em conjunto, caracterizados como estruturas reduzidas de *software*. Esses serviços têm interfaces padronizadas para se comunicarem e têm como características a escalabilidade e a disponibilidade. Quando utilizados em conjunto com a Computação em Nuvem (*Cloud Computing*), essas características se

tornam naturais e facilitam as soluções sob demanda (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016). No entanto, é importante considerar que a adoção de microsserviços pode ser desafiadora para MEs (NEWMAN, 2015). Ainda segundo esse autor, essa arquitetura demanda recursos e habilidades significativas para ser implementada e gerenciada de forma adequada. Portanto, ao decidir pela adoção de microsserviços, é fundamental levar em conta o contexto e as restrições das organizações menores.

A tecnologia digital Computação em Nuvem tem o potencial de transformar uma grande parte da indústria de Tecnologia da Informação (TI), tornando o *software* ainda mais atraente como forma de serviço. Desenvolvedores com inovação e ideias para novos serviços de internet não exigem, na atualidade, grandes investimentos em *hardware* e custo humano para implantar seu serviço. Não é necessário o provisionamento excessivo de um serviço cuja demanda pode não atender às suas previsões, desperdiçando recursos caros. Além disso, as empresas com grandes demandas por processamento podem obter resultados rapidamente, pois seus programas podem ser escalados rapidamente (ARMBRUST et al., 2010).

A arquitetura de aplicações de *software* baseada em microsserviços vem ganhando adoção no mercado de desenvolvimento de *software* após casos de sucesso. Proposta pela Netflix e adotada por outras grandes organizações como Amazon, EBay e Uber (FOWLER, 2015), essa arquitetura permite maior flexibilidade, escalabilidade e agilidade no desenvolvimento e na entrega de produtos. Deve-se mencionar que, em 2020, pela primeira vez o mercado de *software* de aplicações corporativas em nuvem superou o mercado sem o uso de nuvem, impulsionado, em parte, pela pandemia do coronavírus (TIINSIDE, 2022).

No entanto, as MEs enfrentam pressões significativas para se manterem competitivas. Fatores como flexibilidade, criatividade, dinamismo e capacidade de entrega rápida de produtos de qualidade tornam-se cruciais para garantir a sua sobrevivência (HUANG et al., 2021). Nesse contexto, a adoção de microsserviços surge como uma possível solução para atender às demandas de maior flexibilidade, escalabilidade e agilidade no desenvolvimento de *software* (TAIBI; LENARDUZZI; PAHL, 2019).

A escalabilidade horizontal, característica dos microsserviços, permite lidar eficientemente com o crescimento da demanda, garantindo que a infraestrutura

possa acompanhar o aumento na quantidade de usuários e transações (NEWMAN, 2015). Além disso, a modularidade proporcionada pelos microsserviços pode agilizar o desenvolvimento e a manutenção do sistema, permitindo que as MEs possam responder de forma mais ágil às mudanças e atualizações necessárias (NEWMAN, 2015).

Outra questão relevante é a utilização da nuvem, que oferece benefícios como flexibilidade, economia de recursos e integração eficiente com outros serviços (DRAGONI; SILLITTI; SUCCI, 2020). Ainda segundo esses autores, a adoção de microsserviços em nuvem pode permitir que as MEs dimensionem seus recursos de acordo com as necessidades do negócio, evitando investimentos desnecessários em infraestrutura. Além disso, a integração com outros serviços disponíveis na nuvem possibilita uma abordagem mais ágil e eficiente no desenvolvimento de soluções.

Esses benefícios, apontados na literatura, podem permitir que as MEs enfrentem os desafios do mercado de forma mais eficaz, respondendo rapidamente às demandas dos clientes, adaptando-se às mudanças tecnológicas e ganhando uma vantagem competitiva significativa.

1.2 QUESTÃO DE PESQUISA

Com o intuito de alcançar a redução de custos e obter sistemas escaláveis, resilientes e flexíveis, as organizações têm optado por migrar os seus sistemas legados para uma arquitetura fracamente acoplada, a qual se baseia em pequenos fragmentos de *software* independentes e utilizam a Computação em Nuvem como plataforma (ARMBRUST et al., 2010).

Essa abordagem arquitetural tem possibilitado que as empresas se desfaçam de seus *data centers* e migrem para um provedor de nuvem, visando obter serviços que garantam a continuidade de suas operações, junto a todas as vantagens oferecidas nos processos de gerenciamento desses ambientes. A título de exemplo, pode-se citar o Microsoft Azure, que assegura uma disponibilidade de serviço em 99,9% do tempo (MICROSOFT, 2022b), e a Amazon Web Services, que oferece uma garantia de 99,95% (AWS, 2018).

A fim de aproveitar o poder de processamento e flexibilidade da Computação em Nuvem, é necessário desmembrar os *softwares* existentes em pequenas partes, possibilitando ajustes finos no dimensionamento de cada uma. Em contraste, a

hospedagem de um sistema de *e-commerce*, desenvolvido em uma arquitetura monolítica, em um ambiente de Computação em Nuvem é realizada em uma única estrutura, o que significa que as configurações desse ambiente serão aplicadas a todo o sistema, não apenas ao recurso que precisa ser redimensionado.

Embora esse modelo ainda tenha benefícios em termos de escalabilidade, o custo operacional cresce exponencialmente, pois cada recurso de processamento é cobrado individualmente, conforme o conceito “pague conforme o uso”. Por outro lado, as empresas que utilizam uma arquitetura baseada em microsserviços com Computação em Nuvem podem pagar menos pela hospedagem, já que essa arquitetura tem como objetivo decompor grandes estruturas monolíticas em pequenos serviços independentes, permitindo maior flexibilidade e ajustes nos custos (NEWMAN, 2015).

Por exemplo, em um cenário de *e-commerce*, em momentos sazonais, o processo de integração com cartão de crédito pode ter suas capacidades computacionais aumentadas em dezenas de vezes, bastando poucos passos na plataforma da Microsoft Azure para aplicar esse ajuste sem pausar as respostas para os clientes, e os custos operacionais também crescem de forma pontual (MICROSOFT, 2022a). A Amazon Web Services também oferece garantia de 99,95% de disponibilidade dos serviços (AWS, 2018).

O Quadro 1 apresenta os benefícios constantes na literatura, utilizados como referência nesta pesquisa, abrangendo diversas aplicações e não se limitando ao porte da empresa.

Quadro 1 – Problemas e benefícios da arquitetura de microsserviços

Tipo	Autor (referência)	Título do trabalho	Apontamento
Problema	Fowler (2015)	*MicroservicePremium*	Necessidade de gerenciamento de infraestrutura: com a separação em microsserviços, o gerenciamento de infraestrutura pode se tornar mais complexo.
Problema	Newman (2015)	*Building Microservices*	Maior complexidade de testes: a natureza distribuída dos microsserviços pode tornar mais difícil testar a integração entre diferentes serviços e garantir a qualidade geral do sistema.

Problema	Dragoni et al. (2017)	*Microservices: yesterday, today, and tomorrow*	Possibilidade de falha de comunicação: a arquitetura de microsserviços é baseada em serviços independentes, o que pode levar a problemas de comunicação e indisponibilidade do serviço.
Problema	Garriga et al. (2018)	*Towards microservices security trade-offs: Lessons learned*	Problemas de segurança: a complexidade e a distribuição dos microsserviços podem aumentar os riscos de segurança, tornando o sistema mais vulnerável a ataques externos.
Problema	Mukherjee, Roy e Bose (2020)	*Defining an appropriate trade-off to overcome the challenges and limitations in Software Security Testing*	Desafios na implementação da segurança: a implementação da segurança em uma arquitetura baseada em microsserviços pode ser mais complexa do que em um sistema monolítico, pois cada serviço pode ter requisitos de segurança diferentes.
Problema	Dragoni, Sillitti e Succi (2020)	*Microservices in the Cloud: An Overview of Challenges and Solutions*	Complexidade da orquestração: a complexidade de gerenciar microsserviços em grande escala pode levar a problemas de orquestração, como dificuldade em monitorar o desempenho de cada microsserviço e garantir sua escalabilidade.
Problema	Villamizar et al. (2015)	*Evaluation of the Overhead of a Commercial and an Open-Source Cloud Platform*	Aumento do uso de recursos: a fragmentação de aplicativos em microsserviços pode levar a um aumento no uso de recursos, como armazenamento e processamento, devido à necessidade de replicação de dados e comunicação entre serviços.
Benefício	Fowler (2015)	*MicroservicePremium*	Cada serviço pode ser escalado de forma independente para atender à demanda do aplicativo; isso permite o dimensionamento conforme as necessidades de infraestrutura, tanto de forma horizontal quanto de forma vertical.
Benefício	Newman (2015)	*Building Microservices*	Decomposição de grandes estruturas de *software* em pequenos serviços independentes, permitindo maior flexibilidade e controle de custos nas soluções de *software*.
Benefício	Rahman e Gao (2015)	*A reusable automated acceptance testing architecture for microservices in behavior-driven development*	Facilidade de implantar em diversas plataformas de nuvem; os microsserviços são adequados para ambientes com a necessidade de escalabilidade, baixa tolerância a falhas, elevado tempo da disponibilidade e atualizações contínuas no sistema.

Benefício	Viennot et al. (2015)	*A microservices architecture for heterogeneous-database web applications*	Os microsserviços podem ser desenvolvidos com uma tecnologia diferente do projeto principal a fim de atender à demanda utilizando a ferramenta mais adequada, inclusive com banco de dados não relacionais.
Benefício	Harper et al. (2016)	*Microdatabases for the Industrial Internet*	A separação das bases de dados e o fato de os serviços serem pequenos e descentralizados proporcionam a redução da complexidade nas interfaces de comunicação; maior privacidade e segurança nos dados e melhor gestão de conectividade.
Benefício	Villamizar et al. (2017)	*Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures*	O uso de microsserviços em nuvem permite que as empresas reduzam seus custos de infraestrutura em até 77,08%.
Benefício	Furda et al. (2018)	*Migrating enterprise legacy source code to microservices: on multitenancy, statefulness, and data consistency*	Permitem escalar sob demanda; alta disponibilidade; diminuem o tempo médio de resposta distribuindo a carga entre os microsserviços disponíveis; alcançam maior confiabilidade do sistema por meio de redundância; o balanceador de carga pode aumentar a probabilidade de receber uma resposta bem-sucedida.
Benefício	Lenarduzzi et al. (2020)	*Does migrating a monolithic system to microservices decrease the technical debt?*	Redução da dívida técnica[1] das equipes no longo prazo.
Benefício	Blinowski, Ojdowska e Przybylek (2022)	*Monolithic vs. Microservice Architecture: a performance and scalability evaluation*	Os microsserviços são autônomos, autocontidos e têm implementação independente; aumentam a possibilidade de monetização do sistema, pois podem ser comercializados no formato *Software as a Service* (SaaS)

Fonte: autor.

Com base nos autores listados no Quadro 1, pode-se afirmar que os microsserviços apresentam diversas características que oferecem benefícios significativos para as empresas de *software*, incluindo as MEs, que buscam migrar

1 Dívida técnica é quando a equipe de desenvolvimento de *software* escolhe um *design* ou abordagem fácil de implementar no curto prazo, mas com grande impacto negativo no longo prazo.

seus sistemas legados. Embora o Quadro 1 abranja diferentes aplicações e não se limite ao porte da empresa, é possível extrair informações relevantes para as MEs de *software* que desejam adotar microsserviços como parte de sua estratégia de modernização. Essas características podem incluir maior flexibilidade, escalabilidade, agilidade no desenvolvimento e entrega de produtos, além de outros benefícios citados pelos autores presentes no quadro, contribuindo para a melhoria dos sistemas e dos processos das MEs.

Fowler (2015) argumenta que a capacidade de escalonamento independente dos microsserviços permite que os recursos sejam dimensionados de acordo com as necessidades do aplicativo, mas também destaca que a separação em microsserviços pode tornar o gerenciamento da infraestrutura mais complexo.

Newman (2015) aponta que a fragmentação dos aplicativos em microsserviços pode oferecer maior flexibilidade e controle de custos nas soluções de *software*, mas também destaca a natureza distribuída dos microsserviços como um desafio na hora de testar a integração entre diferentes serviços e garantir a qualidade geral do sistema.

Rahman e Gao (2015) destacam a tolerância a falhas e elevado tempo de disponibilidade e atualizações como benefícios dos microsserviços, além da facilidade de implantação em diversas plataformas de nuvem.

Viennot et al. (2015) apontam que os microsserviços são independentes da tecnologia do projeto original, permitindo que diferentes tecnologias e ferramentas sejam usadas para atender à demanda.

Harper et al. (2016) destacam como benefícios dos microsserviços a redução da complexidade nas interfaces de comunicação, o aumento da privacidade e da segurança dos dados, e a melhoria da gestão de conectividade.

Villamizar et al. (2017) destacam que os microsserviços podem reduzir os custos de infraestrutura em até 77,08%.

Blinowski, Ojdowska e Przybylek (2022) apontam que os microsserviços são autônomos, autocontidos e têm implementação independente, o que aumenta a possibilidade de monetização do sistema.

É importante destacar que os benefícios dos microsserviços apresentados anteriormente podem ser limitados em termos de resultados para as MEs de *software*, uma vez que os artigos encontrados na literatura, referentes ao tema,

geralmente se referem a grandes corporações que têm equipes dedicadas e já aplicam as etapas descritas na literatura.

Dessa forma, questiona-se se os mesmos benefícios seriam alcançados de forma direta e mensurável pelas MEs de *software*, bem como se os benefícios apresentados são aplicáveis a esse tipo de organização. É necessário levar em conta a complexidade de se implementar uma arquitetura de microsserviços em um ambiente com recursos e equipes limitados. Nesse sentido, é importante avaliar cuidadosamente os desafios e benefícios específicos para a realidade de cada ME de *software* antes de decidir pela migração de sistemas legados para uma arquitetura de microsserviços.

Nesse cenário, uma questão de pesquisa se apresenta: Como apoiar as MEs de *software* na migração de sistemas com arquitetura monolítica para uma arquitetura baseada em microsserviços em um ambiente de Computação em Nuvem?

1.3 OBJETIVOS

Os objetivos em um projeto de pesquisa são o que confere coerência ao plano de desenvolvimento do projeto. O objetivo geral ressalta onde o estudo pretende chegar ou o que deseja alcançar, e os objetivos específicos são as etapas ou atividades que devem ser cumpridas para que possamos atender ao objetivo geral (FIORENTINI; LORENZATO, 2006; COSTA; COSTA, 2014).

1.3.1 Objetivo geral

Esta pesquisa tem como foco as MEs que desenvolvem *softwares* para uso interno ou que planejam migrar seus sistemas legados, a fim de usufruir dos recursos de Computação em Nuvem e que queiram migrar para a arquitetura baseada em microsserviços.

O objetivo geral é desenvolver e validar um guia de orientações na migração de sistemas de informação da arquitetura monolítica para a arquitetura de microsserviços com o uso da Computação em Nuvem para apoiar as MEs brasileiras.

1.3.2 Objetivos específicos

Para atingir o objetivo geral, os seguintes objetivos específicos foram definidos:

- Analisar, extrair e classificar da literatura as orientações no uso de arquiteturas de microsserviços em relação às arquiteturas monolíticas nos SIs;
- Validar o conjunto de orientações extraídas da literatura junto a especialistas na migração de SIs com arquitetura monolítica para arquitetura de microsserviços em nuvem;
- Validar o guia de orientações desenvolvido junto a especialistas do mercado brasileiro.

1.4 JUSTIFICATIVA

As MEs de *software*, que representam quase metade do mercado brasileiro e geram 1,7 milhão de empregos diretos e 2,6 milhões de empregos indiretos (ABES, 2023), enfrentam desafios constantes para se manterem competitivas. Com recursos limitados para investir em pesquisa e evolução tecnológica, muitas vezes acabam adotando soluções já utilizadas por grandes empresas, mesmo sem a *expertise* (conhecimento) necessária (LI et al., 2020). No entanto, devido à pressão constante para a entrega rápida e eficiente de *software*, essas empresas seguem tendências e soluções já estabelecidas no mercado (HUANG et al., 2021).

Nesse contexto, a arquitetura de microsserviços com Computação em Nuvem tem se tornado uma tendência para o desenvolvimento de *software*, sendo adotada por empresas como Amazon, Spotify e Airbnb (BLINOWSKI; OJDOWSKA; PRZYBYLEK, 2022). A pesquisa da Forrester Research (2019), que entrevistou 1.040 gestores de TI, revelou que 51% das empresas já estavam implementando ou planejando implementar microsserviços em suas operações. Além disso, a Grand View Research (2021) prevê um crescimento anual de 23,4% no mercado global de microsserviços entre 2021 e 2028.

De acordo com a Gartner (2021), até 2025, cerca de 85% das empresas adotarão arquitetura de microsserviços como parte de seus esforços de modernização de aplicativos, com o objetivo de acelerar a inovação e a entrega de

serviços digitais. Essa tendência mostra que a adoção de microsserviços é uma realidade e deve ser considerada pelas empresas que buscam modernizar seus sistemas e se manter competitivas no mercado.

Apesar dos benefícios que a adoção de microsserviços pode trazer, como escalabilidade, flexibilidade e redução de custos operacionais, é importante que as empresas considerem os desafios envolvidos na implementação, como a falta de recursos qualificados nas equipes de desenvolvimento e a complexidade da integração com outros sistemas (SINGH; KUMAR; SINGH, 2018).

Para as MEs, a transição para a arquitetura de microsserviços pode ser onerosa, pois exige maiores custos com seres humanos (LI et al., 2020). No entanto, a arquitetura de microsserviços pode proporcionar facilidade na gestão de um pequeno *software*, o que pode ser uma grande vantagem em um cenário de recursos humanos e financeiros escassos (HISRICH; PETERS; SHEPHERD, 2019).

A adoção da arquitetura de microsserviços pode ajudar a reduzir a dívida técnica das equipes no longo prazo (LENARDUZZI et al., 2020), mas deve ser feita de forma consciente das limitações e desafios envolvidos, considerando as particularidades das MEs de *software* (FREITAS; YAMASHITA, 2017). Como destacado pelos autores Khalique et al. (2018), as micro, pequenas e médias empresas são importantes geradoras de emprego e representam mais de 90% das empresas do mundo, tornando a sobrevivência dessas empresas uma questão crucial.

Nesse sentido, a criação de um guia de orientação para a migração da arquitetura monolítica para arquitetura de microsserviços em nuvem para MEs será uma importante contribuição dessa pesquisa.

1.5. DELIMITAÇÃO DO TEMA

Esta pesquisa teve como objetivo principal a elaboração de um guia de orientações para MEs de *software* brasileiras que desenvolvem produtos (*softwares*) e desejam migrar seus sistemas construídos com arquitetura monolítica para arquitetura de microsserviços em Computação em Nuvem.

O foco desta pesquisa será as MEs de *software* que têm poucos recursos financeiros e humanos para elaborar as suas próprias pesquisas e evoluções tecnológicas, e, por isso, tendem a seguir soluções já utilizadas por grandes

empresas sem a *expertise* (conhecimento) necessária. Não será o foco desta pesquisa as empresas de pequeno, médio e grande porte na área de *software*, tampouco as MEs que apenas prestam consultoria nessa área.

Diversos tópicos que podem ser considerados importantes dentro de outros contextos de avaliação, como conteinerização, granularidade do microsserviço, governança, segurança, *performance* (latência, por exemplo) e gerenciamento de transações, não serão abordados neste trabalho, embora tenham sido mapeados na Revisão Sistemática da Literatura (RSL) realizada como parte do embasamento teórico deste estudo. Esses tópicos foram considerados fora do escopo da questão de pesquisa proposta e foram cortados após a aplicação da curva ABC, que priorizou os temas mais relevantes e essenciais para a elaboração do guia de orientações proposto.

Dessa forma, a pesquisa visa contribuir para a adoção consciente e eficaz da arquitetura de microsserviços por MEs de *software*, considerando suas particularidades e recursos disponíveis, garantindo benefícios a longo prazo e não comprometendo a sobrevivência dessas empresas.

1.6. ESTRUTURA DO TRABALHO

O trabalho está organizado em oito seções, incluindo a introdução e a conclusão. A seção 2 desenvolve o referencial teórico do tema; a seção 3 apresenta a RSL; a seção 4 mostra a metodologia de pesquisa; a seção 5 apresenta a execução do método Delphi; a seção 6 mostra o desenvolvimento do guia de orientações; e a seção 7 aborda as conclusões, as contribuições, as limitações e os estudos futuros.

2. FUNDAMENTAÇÃO TEÓRICA

Esta seção apresenta a plataforma teórica prevista para o desenvolvimento da pesquisa, apresentando os conceitos relacionados com Arquitetura de *Software*, Arquitetura de Microsserviços, Arquitetura Monolítica, Computação em Nuvem e Microempresa, que são assuntos correlatos com a pesquisa.

2.1. ARQUITETURA DE *SOFTWARE*: CONCEITOS E FINALIDADES

A origem dos estudos sobre arquitetura de *software* remonta ao período de 1968 a 1970, quando o cientista da computação holandês Edsger Wybe Dijkstra propôs a ideia de um *software* estruturado por camadas. Em seguida, Fred Brooks e Ken Iverson trataram da estrutura de computadores sob a perspectiva da programação, e em 1970, contribuíram para a engenharia de *software*. A engenharia de *software* e a arquitetura de *software* estão intimamente relacionadas e são interdependentes.

A engenharia de *software* é o processo de aplicar princípios de engenharia para desenvolver *softwares* de alta qualidade de forma eficiente e eficaz. A arquitetura de *software*, por sua vez, é a estrutura ou esqueleto do *software* que fornece a base para sua construção e evolução (BASS; CLEMENTS; KAZMAN, 2012). Ela define a estrutura, os componentes e as interações do *software*, ajudando a garantir que o *software* seja escalável, robusto, eficiente e mantenha alta qualidade ao longo do tempo. A engenharia de *software* envolve a aplicação de métodos e técnicas para desenvolver, manter e evoluir o *software*, enquanto a arquitetura de *software* fornece a estrutura e a base para a construção do *software*.

A escolha de uma arquitetura adequada para o *software* é fundamental para o sucesso do projeto, uma vez que a arquitetura de *software* é um dos principais motivos pelos quais projetos de *software* falham. Um sistema mal projetado ou um descuido durante a implementação pode ocasionar sérios problemas estruturais (HEESCH et al., 2014), tais como desempenho insatisfatório, alto acoplamento de funcionalidade e dificuldade de integração com outros sistemas.

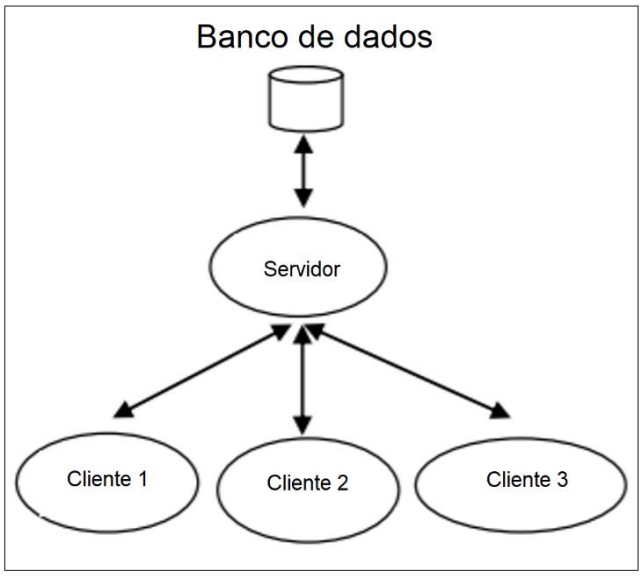
A literatura especializada indica que a arquitetura de *software* deve ser cuidadosamente planejada e implementada para garantir a qualidade do *software* ao

longo do tempo. A arquitetura de *software* deve ser escalável, flexível e adaptável às mudanças nos requisitos do *software* e do negócio, além de ser compreendida e mantida por todos os envolvidos no projeto, facilitando assim a sua evolução. Em uma visão geral, atributos de qualidade são alcançados por boas soluções arquiteturais, e segundo a *International Organization for Standardization* (ISO) 25000, norma para qualidade de produto de *software*, a segurança é uma característica de qualidade, de natureza arquitetural (ABNT, 2005).

Dentro dos estilos arquiteturais existentes, pode-se listar os que têm maior relevância para sistemas corporativos:

- **Cliente-Servidor:** muito utilizado em projetos de *software* e é dividido em duas camadas fisicamente separadas. O servidor é responsável pelo gerenciamento dos dados, enquanto o cliente é responsável pela captura dos dados. Uma aplicação web é um dos exemplos mais comuns dessa arquitetura, no qual o sistema está implantado no servidor e o acesso é feito pelo cliente via navegador de internet (CLEMENTS et al., 2003). A Figura 1 ilustra a estrutura da arquitetura cliente-servidor em um *software*.

Figura 1 – Estrutura de um *software* cliente-servidor



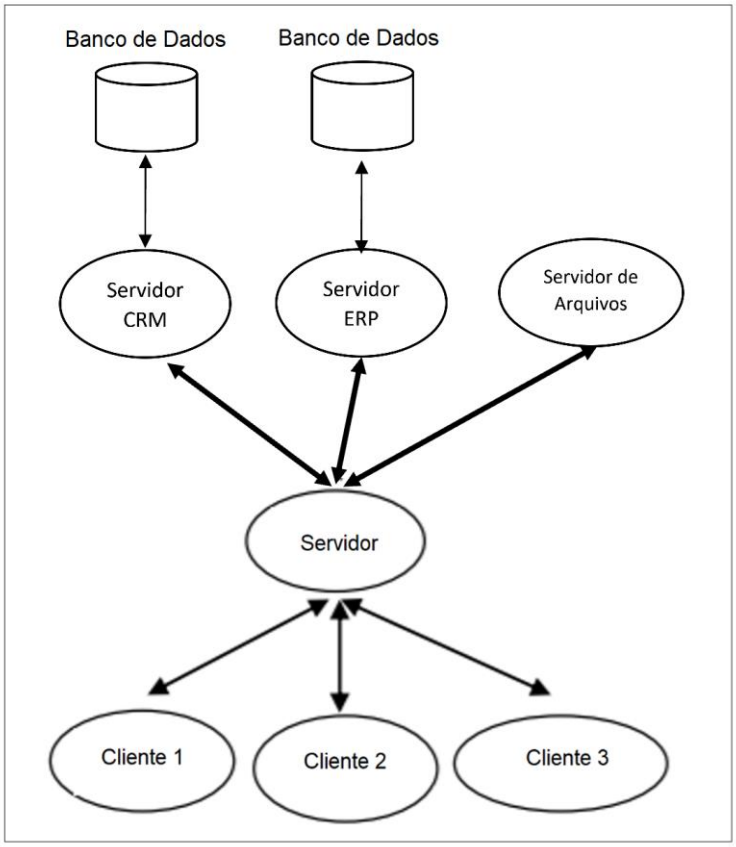
Fonte: adaptado de Jing, Helal e Elmagarmid (1999).

- **Distribuída:** os componentes estão fisicamente separados, porém têm relação; há uma organização lógica e física para o funcionamento dessa abordagem, em que a lógica irá instrumentar a comunicação entre as

estruturas por meio da rede que irá compor a infraestrutura descentralizada (COULOURIS et al., 2001). Geralmente, as estruturas são utilizadas com base de dados também distribuída (DOMASCHKA; HAUSER; ERB, 2014), trazem implicações devido à concorrência e comunicação remota, mas também benefícios como paralelismo no processamento e escalabilidade (LEUNGWATTANAKIT et al., 2014).

A Figura 2 apresenta a estrutura de um *software* com arquitetura distribuída.

Figura 2 – Estrutura de um *software* com arquitetura distribuída



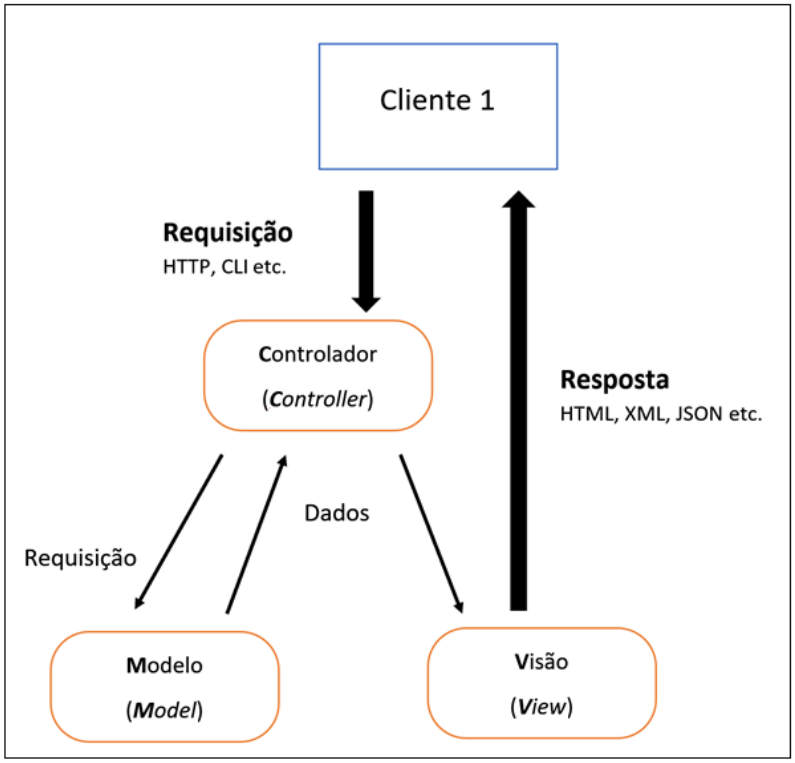
Fonte: adaptado de Oppl (2011).

- **Multicamadas:** o projeto é dividido em camadas, com o objetivo de separar as responsabilidades; por exemplo, no padrão *Model-View-Controller* (MVC) é separado com três camadas, criando uma organização por modelo (*Model*): classes de entidade e classes de serviços, nos quais se encontram o modelo de dados e as regras de negócio; visão (*View*): arquivos das páginas web, nos quais são feitas a interação com o usuário; e o controlador (*Controller*):

classes responsáveis por interligar e controlar o modelo com a visão (CLEMENTS et al., 2003).

A Figura 3 mostra a estrutura de um *software* na arquitetura em multicamadas.

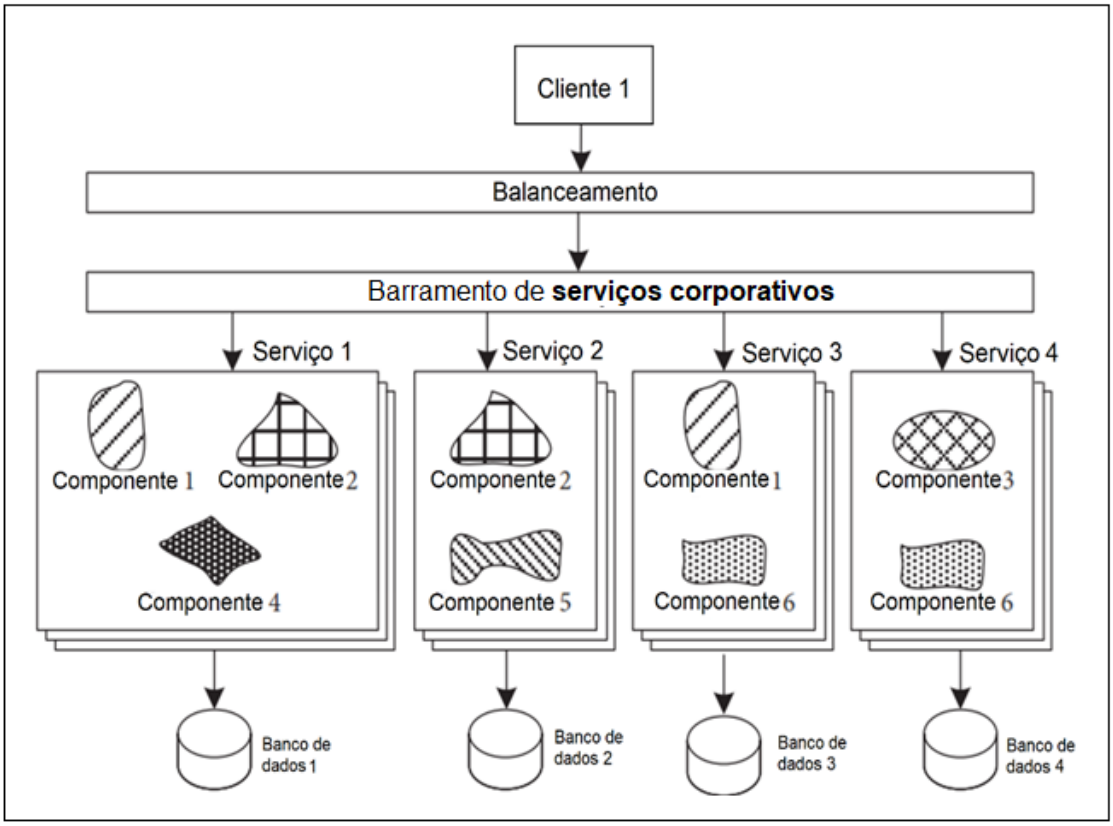
Figura 3 – Estrutura de um *software* multicamadas



Fonte: Adaptado de Pop e Altar (2014).

- **Arquitetura Orientada a Serviços (SOA):** é um padrão de arquitetura de *software* de baixo acoplamento que torna os componentes reutilizáveis, usando interfaces de serviços que são estruturadas de forma que possam ser providas ou consumidas por outras funcionalidades, independentemente da tecnologia, que podem ser *web services* nos padrões: *Simple Object Access Protocol* (SOAP) ou *Representational State Transfer* (REST), *Enterprise Services Bus* (ESB), entre outros métodos de integrações, como apresentado na Figura 4 (CLEMENTS et al., 2003).

Figura 4 – Estrutura de um *Software* Orientado a Serviços



Fonte: adaptado de Li et al. (2020).

A Figura 4 ilustra a estrutura de um *software* na SOA, na qual o cliente 1 faz uma requisição para um serviço responsável pelo balanceamento das requisições. Esse, por sua vez, acessa o barramento de serviços corporativos. A empresa em questão tem quatro serviços (representados na imagem como “serviço 1”, “serviço 2”, “serviço 3” e “serviço 4”). Cada serviço tem o seu próprio banco de dados, e esses serviços são compostos de diversos componentes.

2.2. COMPUTAÇÃO EM NUVEM (*CLOUD COMPUTING*): CONCEITOS E FINALIDADE

A Computação em Nuvem refere-se tanto a aplicativos entregues como serviços na internet quanto ao *hardware* e ao *software* de sistemas nos *data centers* que fornecem esses serviços (ARMBRUST et al., 2010). Ainda segundo esses autores, uma grande vantagem desse modelo é a possibilidade de pagar pelo uso dos recursos de computação conforme o uso necessário (por exemplo, processadores por hora e armazenamento por dia) e liberá-los assim que possível;

com isso, essa solução é capaz de oferecer serviços abaixo dos custos de um *data center* de médio porte e ainda assim gerar bom lucro.

A nova realidade do trabalho híbrido está levando as organizações a deixar de capacitar sua força de trabalho com soluções tradicionais de computação para clientes, como *desktops* e outras ferramentas físicas no escritório; por isso, a previsão é de crescimento, como ilustrado no Quadro 2.

O Quadro 2 apresenta os gastos mundiais com todos os tipos de Computação em Nuvem de 2021 e as estimativas para 2022 e 2023.

Quadro 2 – Gastos com Computação em Nuvem (em milhões de dólares)

	2021	**2022**	**2023**
Cloud Business Process Services (BPaaS)	51,410	55,598	60,619
Cloud Application Infrastructure Services (PaaS)	86,943	109,623	136,404
Cloud Application Services (SaaS)	152,184	176,622	208,080
Cloud Management and Security Services	26,665	30,471	35,218
Cloud System Infrastructure Service (IaaS)	91,642	119,717	156,276
Desktop as a Service (DaaS)	2,072	2,623	3,244
Total Market	**410,915**	**494,654**	**599,840**

Fonte: adaptado de Gartner (2022).

De acordo com a Gartner (2022), o mercado de serviços de nuvem pública, especificamente o segmento de SaaS, continua em ascensão, sendo o maior segmento em 2022, cuja previsão era atingir US$ 176,6 bilhões em gastos de usuários finais. Isso se deve ao fato de as empresas estarem adotando diversas estratégias para ingressar no mercado de SaaS, como por meio de mercados de nuvem, além de continuarem a decompor e transformar aplicativos maiores e monolíticos em partes que podem ser compostas para processos de DevOps mais eficientes.

Vale ressaltar que o uso da nuvem para hospedar microsserviços é algo natural, pois um dos principais benefícios é a capacidade de escalabilidade vertical e horizontal, permitindo que as funcionalidades possam ser escaladas de forma independente (NEWMAN, 2015).

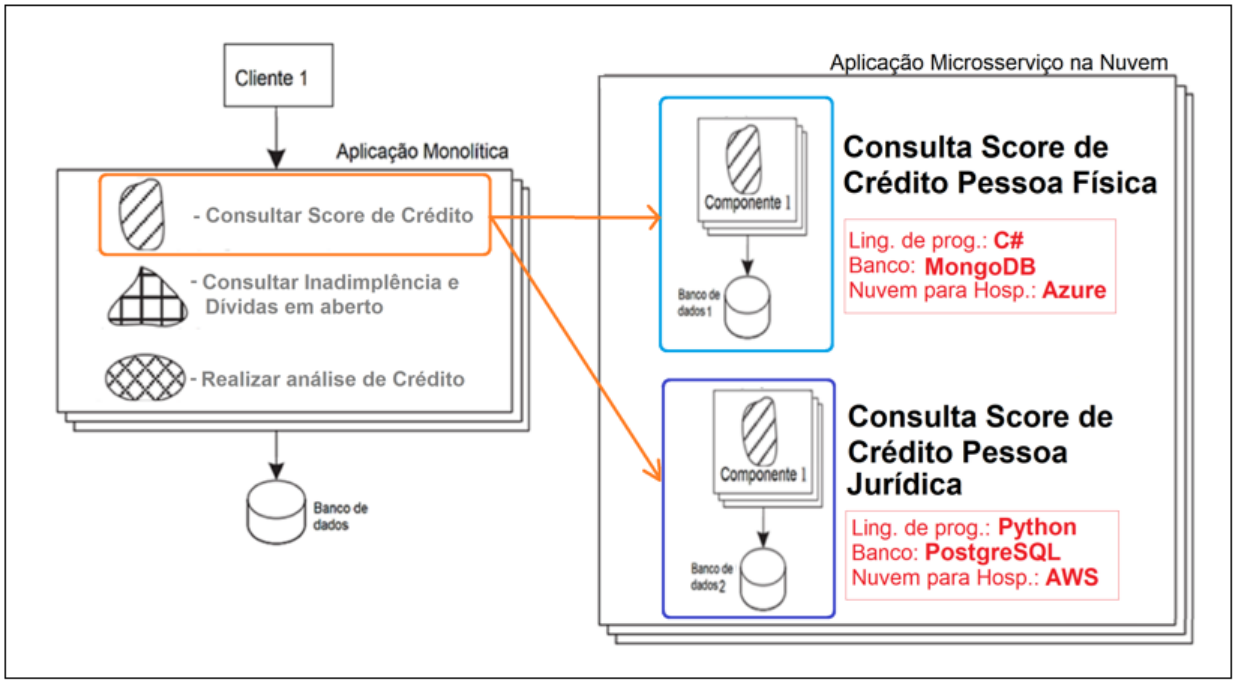
A Microsoft Azure (MICROSOFT, 2022b) descreve que a escalabilidade vertical, ou *scale up*, envolve a possibilidade de dimensionar verticalmente um único nó, adicionando ou removendo recursos como CPUs, memória ou armazenamento em um único computador. Embora isso possa aumentar a complexidade do gerenciamento, exige programação mais sofisticada para alocar tarefas entre recursos e lidar com problemas como taxa de transferência e latência entre nós.

Por outro lado, a escalabilidade horizontal, ou *scale out*, permite dimensionar horizontalmente um sistema, ou seja, adicionar ou remover nós de um aplicativo distribuído. Por exemplo, pode-se expandir um servidor web de um para três. Aplicativos de alto desempenho, como análises sísmicas e biotecnologia, dimensionam cargas de trabalho horizontalmente para suportar tarefas que antes exigiam supercomputadores caros. Outras cargas de trabalho, como grandes redes sociais, excedem a capacidade do maior supercomputador e só podem ser tratadas por sistemas escaláveis (MICROSOFT, 2022b). Tanto a Amazon Web Services quanto a Microsoft Azure oferecem recursos de dimensionamento automático, além de *dashboards* para gerenciamento e acompanhamento da operação em tempo real (AWS, 2022; MICROSOFT, 2022b).

Além disso, o modelo SaaS oferece às MEs de *software* uma grande oportunidade para monetizar seus produtos e serviços de diferentes maneiras (BLINOWSKI; OJDOWSKA; PRZYBYLEK, 2022). Ainda segundo os autores, é possível oferecer assinaturas mensais ou anuais para acesso a aplicativos e serviços, além de realizar vendas de dados. Ao desmembrar uma aplicação

monolítica em microsserviços na nuvem, é possível estruturá-los para serem comercializados no formato SaaS, o que pode ajudar as MEs a aproveitar as vantagens da Computação em Nuvem para expandir seus negócios e diversificar seus modelos de negócios, aumentando suas chances de sucesso no mercado. Como apresentado na Figura 5, a estruturação de microsserviços em nuvem para serem comercializados como SaaS é uma maneira de expandir o alcance dos produtos e serviços oferecidos pelas MEs de *software*.

Figura 5 – Desmembramento de uma aplicação monolítica para microsserviços e comercialização no formato *Software as a Service*



Fonte: adaptado de Li et al. (2020).

No exemplo da Figura 5, a empresa converteu um módulo de sua aplicação monolítica em microsserviços na nuvem, o que gerou dois novos microsserviços: consultar *score* de crédito pessoa física e consultar *score* de crédito pessoa jurídica. Essa abordagem oferece a possibilidade de comercialização dos serviços no formato SaaS, permitindo que outras empresas contratem somente o que é relevante para elas. Cada equipe responsável pelas linhas de produtos pode fazer suas escolhas tecnológicas sem afetar o serviço da outra equipe, aumentando a eficiência e a produtividade do processo de desenvolvimento. Por exemplo, para o serviço de pessoa física, a equipe optou por utilizar C#, banco de dados MongoDB e

hospedagem na nuvem da Microsoft, a Azure, enquanto para o serviço de pessoa jurídica, a equipe escolheu o Python, o banco de dados PostgreSQL e a hospedagem na nuvem da Amazon. Em resumo, a migração para a arquitetura de microsserviços pode gerar oportunidades de negócios e oferecer maior eficiência e produtividade para as empresas, além de permitir a escolha de tecnologias específicas para cada serviço.

2.3. MONOLÍTICA E MICROSSERVIÇOS: CONCEITOS E FINALIDADE

Na Engenharia de *Software*, existem atualmente dois paradigmas que dominam o desenvolvimento de sistemas corporativos modernos: a arquitetura monolítica e a arquitetura baseada em microsserviços (FOWLER, 2015).

A arquitetura monolítica é uma abordagem tradicional em que um *software* é construído com uma única estrutura de código que pode incluir vários serviços, mas esses serviços não são executáveis de maneira independente (TERZIĆ et al., 2018). Eles se comunicam com usuários finais e sistemas externos por meio de diferentes interfaces.

Por outro lado, a arquitetura baseada em microsserviços foi anunciada em 2011 na 33rd Degree Conference em Kraków (FOWLER, 2015) e começou a ganhar popularidade e força em 2014, com a adoção da Netflix e o crescimento do uso de tecnologias baseadas em Computação em Nuvem, como Kubernetes e Docker (VIGGIATO et al., 2018).

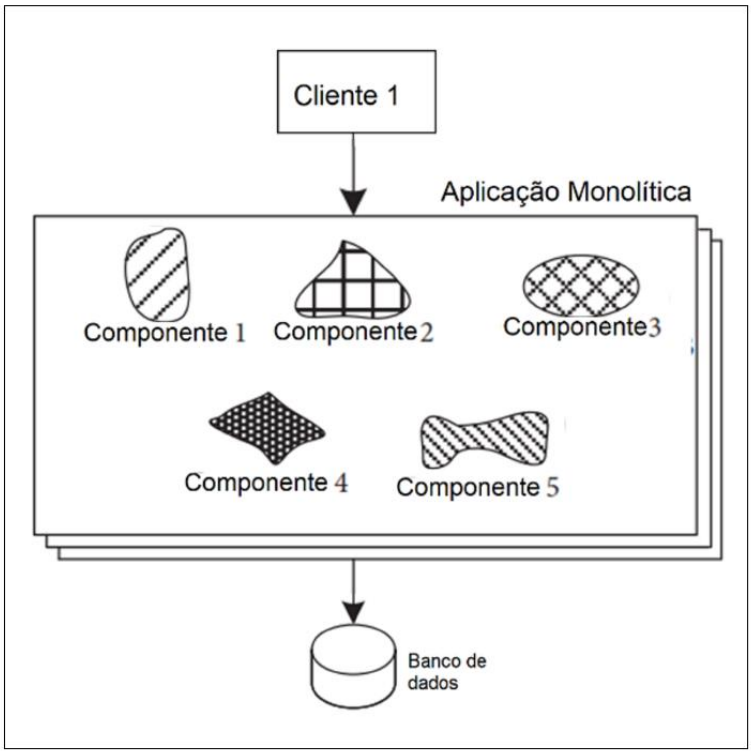
Os microsserviços foram amplamente adotados por empresas globais, como Amazon, eBay, Spotify, Uber, Airbnb, LinkedIn, Twitter e Coca-Cola. Entender as diferenças entre essas arquiteturas é fundamental para escolher a melhor opção para cada tipo de sistema e cenário de negócio (BLINOWSKI; OJDOWSKA; PRZYBYLEK, 2022).

2.3.1. Arquitetura monolítica

De acordo com Savchenko, Radchenko e Taipale (2015), no *design* monolítico, a estrutura de código-fonte do sistema é centralizada em um único local, ou seja, quando uma aplicação é construída em uma grande e única estrutura,

independe da complexidade, do tamanho do código-fonte e sua estrutura é em apenas uma unidade, como apresentado na Figura 6.

Figura 6 – Estrutura de um *software* monolítico



Fonte: adaptado de Li et al. (2020).

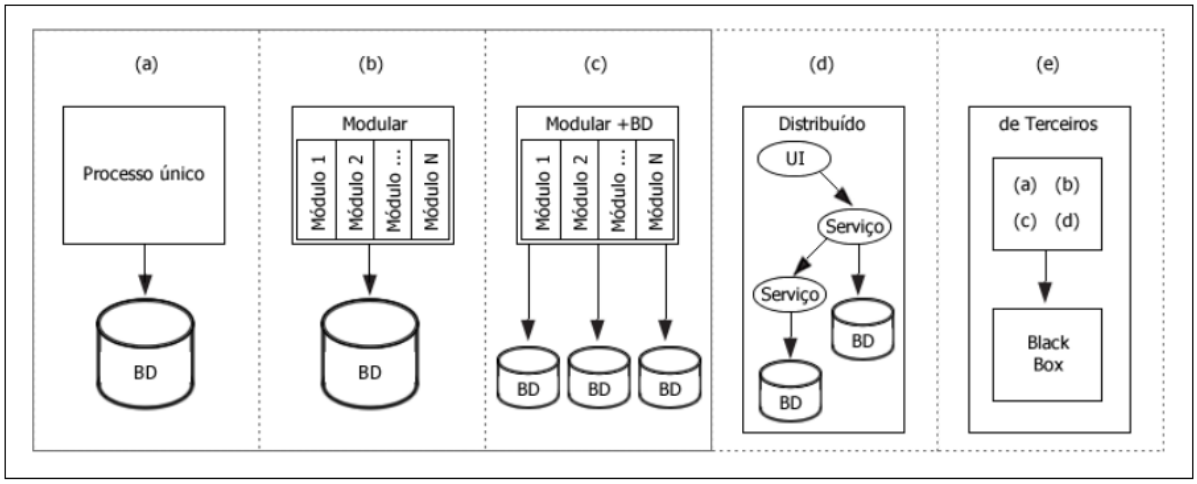
Os sistemas corporativos geralmente são construídos de acordo com o modelo clássico de três camadas; portanto, consistem em: (1) camada de apresentação (*interface* com usuário); (2) camada de negócios; e (3) camada de acesso a dados (KALSKE; MÄKITALO; MIKKONEN, 2018). O *software* do lado do servidor é um monolito, um único executável lógico, do ponto de vista do sistema operacional, executado como um único processo (FOWLER, 2015).

Segundo Newman (2015), há cinco tipos identificados de monolíticos:

- Monolítico de processo único: representa a maioria dos monolíticos; todo o código é publicado em um único processo (Figura 7A);
- Monolítico modular: divisão em módulos independentes, mas que pertencem a um único pacote (Figura 7B);
- Monolítico modular com banco de dados decomposto: talvez o mais desafiador no que se refere à publicação (Figura 7C);

- Monolítico distribuído: consiste em múltiplos serviços separados; independentemente da razão, o sistema inteiro precisa ser publicado junto (Figura 7D);
- Monolítico caixa preta de terceiros: são *softwares* desenvolvidos por outros, sem acesso ao código-fonte, como *softwares* de prateleiras com a publicação na própria infraestrutura. Pode ser, ainda, um produto consumido, tipo SaaS (Figura 7E).

Figura 7 – Tipos de sistemas monolíticos



BD: banco de dados; UI: *User Interface* (Interface do usuário).

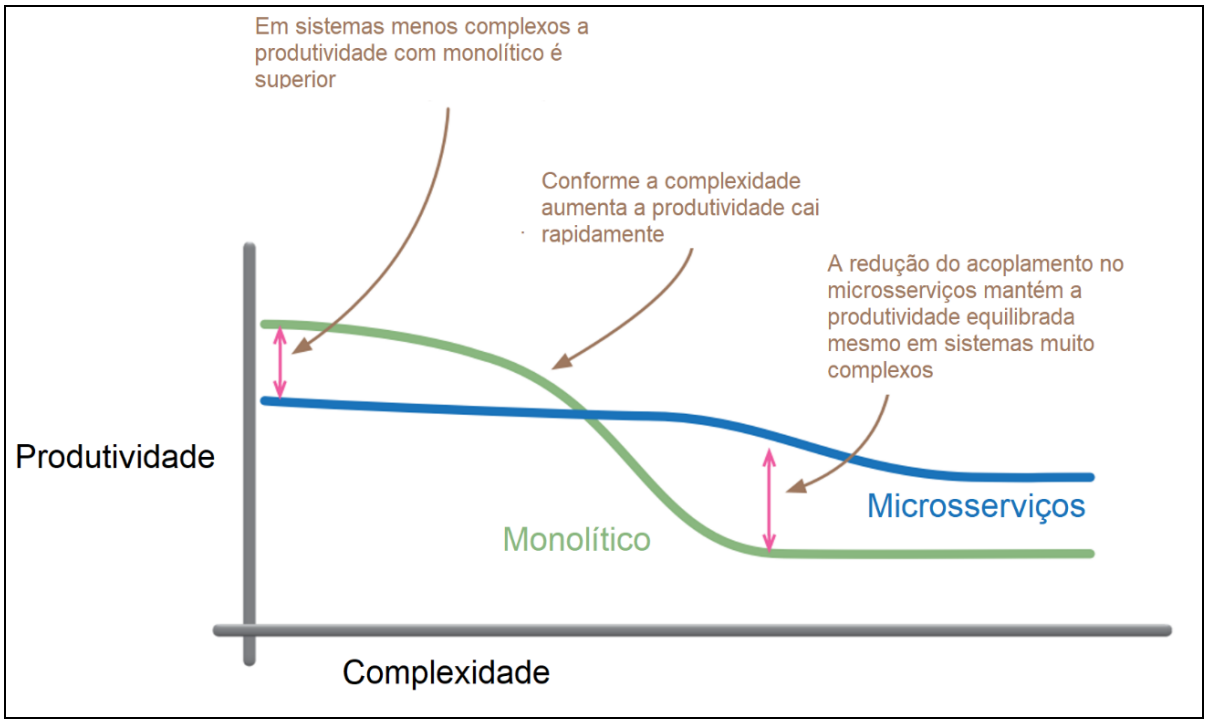
Fonte: adaptado de Newman (2015).

As principais características dessa arquitetura são:

- Código-fonte centralizado: o código-fonte do *software* encontra-se em um único repositório (SAVCHENKO; RADCHENKO; TAIPALE, 2015);
- Escalabilidade: por se tratar de um único executável, é possível escalar o *software* de forma horizontal, em que o artefato gerado é replicado horizontalmente, permitindo um balanceamento de carga no acesso ao sistema (FOWLER, 2015);
- Implantação total da aplicação: como a solução contém todo o código-fonte, só é possível publicar o sistema inteiro, ou seja, a implantação é referente ao sistema como um todo (SAVCHENKO; RADCHENKO; TAIPALE, 2015).

A vantagem mais significativa da arquitetura monolítica é sua simplicidade. Em comparação com sistemas distribuídos, os monolíticos são mais fáceis para testar, implantar, depurar e monitorar, permitindo a execução por uma equipe menor e com menos qualificação (BLINOWSKI; OJDOWSKA; PRZYBYLEK, 2022). Segundo Fowler (2015), monolíticos são menos complexos de início se comparados aos microsserviços, principalmente devido à curva de aprendizado menor requerida, como mostra a Figura 8.

Figura 8 – Produtividade *versus* complexidade entre monolítico e microsserviços



Fonte: adaptado de Fowler (2015).

Alguns cuidados são necessários ao adotar a arquitetura monolítica no desenvolvimento de sistemas, especialmente quando eles crescem e acumulam um grande número de recursos e usuários. Isso ocorre porque todas as funções em um único projeto podem impedir a expansão e dificultar a manutenção. Além disso, a atualização do sistema requer *clustering* (agrupamento) em ambos os níveis de aplicativos e bancos de dados, o que pode ser caro. Finalmente, grandes volumes de dados sendo compartilhados com grande concorrência podem gerar gargalos na leitura e escrita (LI et al., 2020).

2.3.2. *Application Programming Interface* (API)

Application Programming Interfaces (APIs) são fundamentais para o desenvolvimento de sistemas modernos, uma vez que permitem a comunicação entre diferentes componentes de *software* de forma padronizada e eficiente. Segundo Red Hat (2023), a interface de uma API é como um contrato de serviço entre duas aplicações, definindo como elas se comunicam por meio de solicitações e respostas. Essa arquitetura é comumente utilizada com o conceito de cliente e servidor, em que a aplicação que envia a solicitação é o cliente e a que envia a resposta é o servidor.

Um exemplo prático de utilização de APIs é a aplicação de previsão do tempo em um telefone, que se comunica com um sistema de *software* do instituto meteorológico por meio de APIs para obter os dados meteorológicos diários. É importante ressaltar que todos os serviços da Web são APIs e que, no contexto de APIs, a palavra “aplicação” se refere a qualquer *software* com uma função distinta AWS (2023).

Segundo Red Hat (2023), existem quatro maneiras diferentes pelas quais as APIs podem funcionar, dependendo de quando e por que elas foram criadas:

- APIs SOAP: utilizam o *Simple Object Access Protocol* (Protocolo de Acesso a Objetos Simples). Cliente e servidor trocam mensagens usando XML. É uma API menos flexível, mais popular no passado.
- APIs RPC: utilizam o *Remote Procedure Calls* (Chamadas de Procedimento Remoto). O cliente conclui uma função (ou um procedimento) no servidor e o servidor envia a saída de volta ao cliente.
- APIs *WebSocket:* mais moderno, utilizam objetos JSON[2] para transmitir dados. Oferecem suporte à comunicação bidirecional entre aplicativos cliente e o servidor. O servidor pode enviar mensagens de retorno de chamada a clientes conectados, tornando esse tipo de API mais eficiente que a API REST.
- APIs REST*:* são as APIs mais populares e flexíveis encontradas na web atualmente. O cliente envia solicitações ao servidor como dados. O servidor

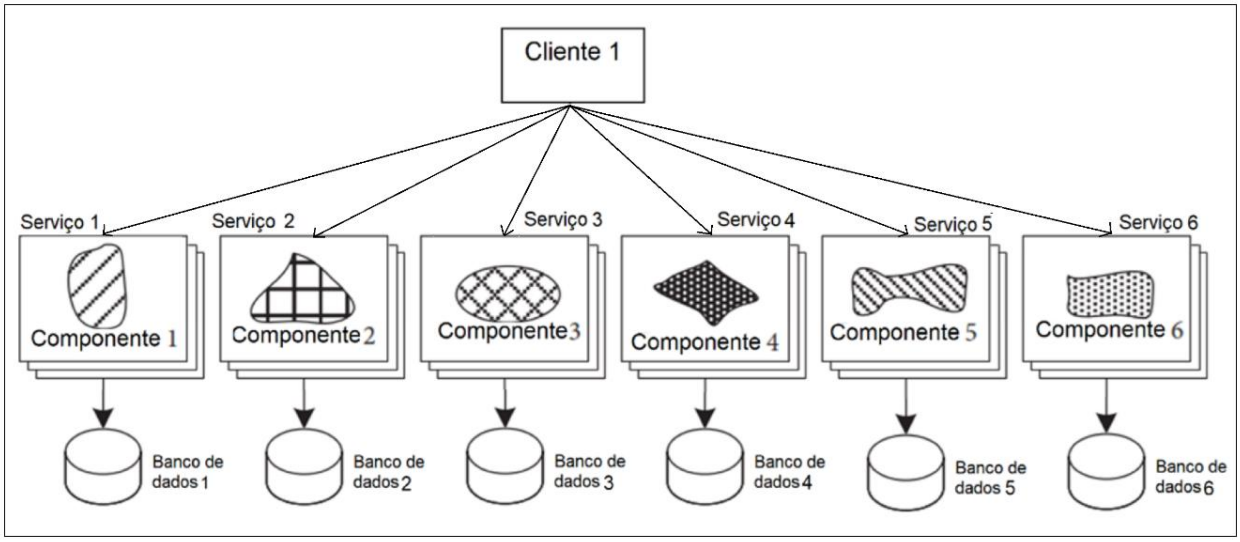
2 O formato JSON (*JavaScript Object Notation*) é um formato aberto usado como alternativa ao XML para a transferência de dados estruturados entre um servidor de web e uma aplicação web.

usa essa entrada do cliente para iniciar funções internas e retorna os dados de saída ao cliente.

2.3.3. Arquitetura de microsserviços

Microsserviços é um *design* arquitetural que decompõe suas funcionalidades por responsabilidade em pequenos serviços autônomos que trabalham em conjunto, o que os caracteriza como estruturas reduzidas de *software* que têm interfaces padronizadas para se comunicarem, além da possibilidade de as funcionalidades serem escaladas isoladamente, como apresentado na Figura 9 (NEWMAN, 2015).

Figura 9 – Estrutura de um *software* baseado em microsserviços



Fonte: adaptado de Li et al. (2020).

Uma das diferenças entre as abordagens de microsserviços e monolíticas é que os monolitos são dependentes de um único tipo de banco de dados, enquanto nas arquiteturas distribuídas cada aplicação pode usar o tipo de banco de dados que melhor se adapta ao cenário, o que faz com que os sistemas distribuídos estejam mais preparados para utilizar soluções de bancos de dados não relacionais como o NoSQL (VIENNOT et al., 2015).

Em sistemas críticos baseados em Computação em Nuvem, a necessidade de escalabilidade, a baixa tolerância a falhas, o tempo elevado de disponibilidade e atualizações contínuas no sistema são indispensáveis (RAHMAN; GAO, 2015).

Segundo esses autores, esse cenário pode favorecer o uso da arquitetura de microsserviços, pois suas características de serviços pequenos e independentes podem contribuir para reduzir esses problemas.

A implementação dos microsserviços com a separação das bases de dados traz um novo conceito de solução: o microbanco de dados. Além de serviços e bancos de dados serem pequenos e descentralizados, trazem três grandes benefícios: redução da complexidade nas interfaces de comunicação, maior privacidade e segurança nos dados e melhor gestão de conectividade (HARPER et al., 2016).

As principais características dessa arquitetura são:

- Responsabilidade única: segue os princípios SOLID, nos quais uma única unidade deve ter apenas uma responsabilidade e em nenhum momento duas unidades devem compartilhar uma responsabilidade ou uma unidade pode ter mais de uma responsabilidade (BLINOWSKI; OJDOWSKA; PRZYBYLEK, 2022).
- Agilidade: promove uma organização de equipes pequenas e independentes que são proprietárias de seus serviços. As equipes atuam dentro de um contexto pequeno e claramente compreendido e têm autonomia para trabalhar de forma mais independente e rápida. O resultado é a aceleração dos ciclos de desenvolvimento (AWS, 2022).
- Escalabilidade: permite que cada serviço seja escalado de forma independente para atender à demanda do recurso de aplicativo oferecido. Isso permite o dimensionamento conforme as necessidades de infraestrutura, tanto de forma horizontal quanto de forma vertical (FOWLER, 2015).
- Implantação parcial da aplicação: os microsserviços são autônomos, autocontidos e têm implementação independente. Eles contêm todas as dependências, como bibliotecas, ambientes de execução, servidores web e contêineres ou máquinas virtuais. Assim, os microsserviços aumentam a possibilidade de monetização do sistema, pois pode ser comercializado no formato SaaS (BLINOWSKI; OJDOWSKA; PRZYBYLEK, 2022).

Segundo Rahman e Gao (2015), é preciso tomar algumas precauções ao adotar o conceito de icrosserviços no desenvolvimento de sistemas, pois essa abordagem pode aumentar a complexidade devido às integrações entre os serviços que ocorrem em tempo de execução, ao contrário dos sistemas monolíticos, cujos

erros de comunicação ocorrem em tempo de compilação, já que estão na mesma estrutura.

Nos sistemas monolíticos, não havia a preocupação com a segurança entre os módulos, já que todos estavam compilados em um mesmo pacote. Entretanto, na arquitetura baseada em microsserviços, a segurança das informações deve ser uma preocupação constante. Os serviços devem ser invocados somente após a autenticação, e o ideal seria estabelecer níveis de privilégio de acordo com cada requisição (NEHME et al., 2019).

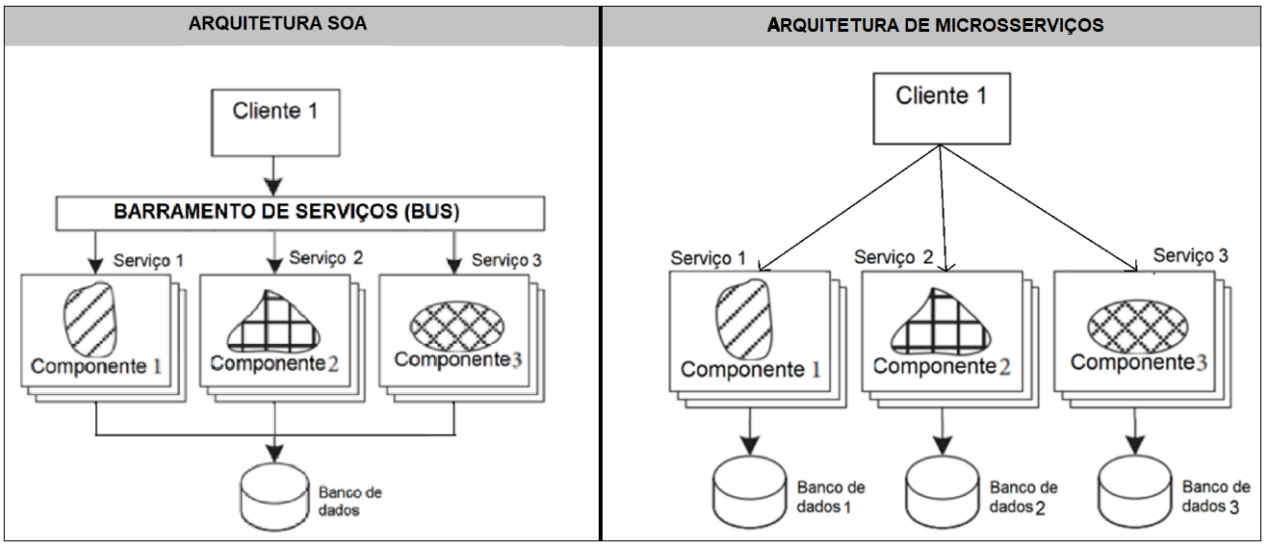
Outro aspecto difícil de definir é a granularidade do microsserviço, ou seja, quão pequeno cada serviço deve ser. Uma definição equivocada pode trazer desequilíbrio entre o custo da governança de serviços e a quantidade de serviços criados (LI et al., 2020). Segundo esses autores, a equipe envolvida nesse tipo de projeto deve ser mais qualificada, o que acarreta custos mais elevados.

2.3.4. Comparativo entre *Software* Orientado a Serviços e microsserviços

A comparação entre a arquitetura de microsserviços e a SOA é importante para compreender as características e as abordagens dessas arquiteturas. A SOA, cujo conceito se popularizou nos anos 2000, é bastante conhecida. Embora ambas compartilhem princípios e objetivos comuns, como o uso de serviços independentes para modularizar aplicações e permitir sua reutilização em diversos contextos, suas abordagens apresentam distinções significativas.

A SOA é uma arquitetura baseada em serviços de natureza monolítica, enquanto a arquitetura de microsserviços é uma abordagem distribuída baseada em serviços independentes e coesos, que se comunicam por meio de protocolos de rede padronizados e interfaces bem definidas (DRAGAN; BARNES, 2019), como ilustrado na Figura 10.

Figura 10 – Comparativo entre *Software* Orientado a Serviços e microsserviços



Fonte: adaptado de Li et al. (2020).

Na SOA, há uma camada intermediária denominada “barramento de serviços” (BUS), que é responsável por gerenciar a comunicação entre os serviços e as aplicações clientes, incluindo a camada de apresentação (UI). O barramento de serviços funciona como um intermediário, fornecendo uma interface unificada e padronizada para a comunicação entre as aplicações e os serviços subjacentes (NAIK; PATEL, 2017).

Já na arquitetura de microsserviços, não há uma camada dedicada para interceptar as solicitações do cliente. Cada microsserviço pode expor a sua própria API, que é acessada diretamente pela aplicação cliente. Dessa forma, não há a necessidade de um barramento intermediando a comunicação, o que pode levar a uma redução de latência e melhorias de desempenho. Entretanto, a comunicação direta entre o cliente e os microsserviços pode tornar mais desafiadores o gerenciamento e a segurança dos serviços (FOWLER, 2015).

Uma das vantagens da SOA é a capacidade de reutilização de serviços em diferentes contextos e sistemas, o que pode levar à maior eficiência e economia de recursos (NAIK; PATEL, 2017). Outra vantagem é que a SOA pode ser mais adequada para sistemas complexos, em que as funcionalidades são interdependentes e requerem uma forte coordenação entre elas. Nesse sentido, a SOA pode oferecer uma abordagem mais coesa para gerenciar essas funcionalidades (ESSA et al., 2019). A SOA pode fornecer uma abordagem mais adequada para a integração de sistemas legados e aplicativos de terceiros, pois

permite maior flexibilidade na comunicação entre os serviços (PAZAZOGLOU; TRAVERSO, 2017).

Por outro lado, a arquitetura de microsserviços oferece maior granularidade e modularidade em comparação com a SOA, permitindo que as empresas gerenciem melhor a complexidade de sistemas distribuídos (DRAGAN; BARNES, 2019). A escalabilidade horizontal é outra vantagem do microsserviço, uma vez que os serviços podem ser replicados e distribuídos em diferentes máquinas, o que possibilita lidar com altas cargas de trabalho de forma mais eficiente (NEWMAN, 2015).

Além disso, a flexibilidade na escolha de tecnologias e linguagens de programação é uma vantagem do microsserviço, uma vez que cada serviço pode ser implementado independentemente e em diferentes plataformas (FOWLER, 2015). A arquitetura de microsserviços também pode ser mais adequada para ambientes de nuvem, uma vez que a flexibilidade e a escalabilidade horizontal dos microsserviços são ideais para lidar com sistemas distribuídos em larga escala (MELL; GRANCE, 2011; DRAGAN; BARNES, 2019).

Todavia, é importante mencionar que a adoção de uma arquitetura de microsserviços pode trazer desafios em relação ao gerenciamento e à segurança dos serviços. Como cada microsserviço pode expor sua própria API, é necessário garantir que a integração entre eles seja feita corretamente e que as APIs estejam protegidas contra possíveis ataques cibernéticos. Além disso, a comunicação direta entre o cliente e os microsserviços pode tornar mais complexa a tarefa de monitorar e gerenciar a infraestrutura (REED; SHARMA; ISERMANN, 2016).

2.3.5. Comparativo entre arquitetura monolítica e microsserviços

As arquiteturas de microsserviços e monolítica são modelos distintos de desenvolvimento de *software*, com suas vantagens e desvantagens.

Na arquitetura monolítica, todo o sistema é desenvolvido em um único bloco de código e é mais adequada para projetos menores e menos complexos (FOWLER, 2015). Já na arquitetura de microsserviços, o sistema é dividido em serviços independentes que se comunicam por meio de interfaces, permitindo maior flexibilidade e escalabilidade em projetos maiores e mais complexos (NEWMAN, 2015).

Ainda segundo Newman (2015), a escolha entre as arquiteturas deve ser baseada em uma análise cuidadosa dos requisitos do projeto e do contexto em que será utilizado. Algumas dessas características podem ser observadas no Quadro 3.

Quadro 3 – Comparação da arquitetura de microsserviços com a monolítica

Característica	Microsserviços	Monolítica
Atualizações em produção	Atualizações mais rápidas e menos arriscadas (KAZANAVIČIUS; MAŽEIKA, 2020).	Atualizações mais lentas e arriscadas (KAZANAVIČIUS; MAŽEIKA, 2020).
Autocontidos	Cada microsserviço é autocontido e pode ser implantado de forma independente, o que pode resultar em maior agilidade na implantação e na manutenção do sistema (NEWMAN, 2015).	Todos os componentes estão em um único processo, o que pode tornar a implantação e a manutenção do sistema mais complexas (FOWLER, 2015).
Coletar dados de múltiplos locais	Requer uma abordagem mais complexa em relação à comunicação e à integração dos dados de múltiplos serviços, o que pode afetar o tempo de resposta e exigir mais cuidado (NEWMAN, 2015).	Com todos os dados em um único processo, é possível gerenciar mais facilmente os dados de múltiplos locais, já que podem ser feitos dentro do próprio processo (FOWLER, 2015).
Complexidade	Maior complexidade, pois deve gerenciar diversos fatores como disponibilidade, resiliência, segurança e monitoramento (HASSAN et al., 2021).	Menor complexidade inicial. A complexidade cresce à medida que o sistema cresce (FOWLER, 2015).
Desempenho	A comunicação é baseada em APIs e protocolos leves; porém, como o processamento é distribuído, fatores externos, como problemas na rede ou sobrecarga de servidores, podem causar lentidão no sistema (KAZANAVIČIUS; MAŽEIKA, 2020).	Desempenho geralmente mais consistente, pois a comunicação é direta, por meio de chamadas internas (KAZANAVIČIUS; MAŽEIKA, 2020).
Escalabilidade	Escalabilidade granular e adaptável (NEWMAN, 2015).	Escalabilidade limitada e menos flexível (NEWMAN, 2015).
Excesso de requisições	Pode ocorrer um excesso de requisições entre serviços, o que	Excesso de requisições pode ser mais raro, já que as chamadas são feitas

	pode impactar no desempenho e na eficiência do sistema (KAZANAVIČIUS; MAŽEIKA, 2020).	dentro do mesmo processo (FOWLER, 2015).
Gestão dos serviços	Requer maior atenção em relação à gestão dos serviços, já que cada serviço é uma unidade independente (NEWMAN, 2015).	Mais simples, já que todos os componentes estão em um único processo (FOWLER, 2015).
Governança	Requer uma abordagem mais proativa em relação ao monitoramento da infraestrutura para garantir a segurança e o bom funcionamento dos serviços (HASSAN et al., 2021).	É mais simples devido à sua estrutura unificada, facilitando o entendimento das dependências e das interações entre os componentes (NEWMAN, 2014).
Integração Contínua (*Continuous Integration* – CI) e Entrega Contínua (*Continuous Delivery* – CD)	É mais simples a implantação CI/CD (NEWMAN, 2015).	Mais complexa, uma vez que o código do sistema é construído como um único bloco, o que pode dificultar a implantação de atualizações e alterações de forma rápida e eficiente (NEWMAN, 2015).
Limites de responsabilidade	Definir os limites de cada microsserviço não é uma tarefa trivial e pode trazer sérias consequências se não for realizada adequadamente (NEWMAN, 2015).	Como todo o sistema é gerenciado como um único bloco de código, a definição dos limites de responsabilidade é mais fácil e clara (DRAGONI; SILLITTI; SUCCI, 2020).
Multilinguagem	Permite que cada microsserviço seja desenvolvido em uma linguagem de programação diferente, o que pode resultar em maior flexibilidade e agilidade no desenvolvimento do sistema (NEWMAN, 2015).	Todos os componentes são desenvolvidos em uma única linguagem de programação, o que pode tornar o desenvolvimento do sistema mais limitado (FOWLER, 2015).
Nativo para estrutura de nuvem	É uma arquitetura mais adequada para a estrutura de nuvem, já que permite a escalabilidade e a implantação em ambientes de nuvem de forma mais eficiente (KAZANAVIČIUS; MAŽEIKA, 2020).	Para extrair todos os recursos que a nuvem oferece, essa não é a arquitetura mais adequada, já que a escalabilidade e a implantação em ambientes de nuvem podem ser mais limitadas (FOWLER, 2015).
Qualificação da equipe	Requer maior qualificação da equipe (KAZANAVIČIUS; MAŽEIKA, 2020).	Requer menor qualificação da equipe (FOWLER, 2015).

Redução de custo	Redução de custo a longo prazo, já que a escalabilidade e a implantação em ambientes de nuvem podem ser mais eficientes (NEWMAN, 2015).	Maior custo a longo prazo, já que a escalabilidade e a implantação em ambientes de nuvem podem ser mais limitadas (FOWLER, 2015).
Reutilização de código	Facilita a reutilização de código e componentes (NEWMAN, 2015).	Mais difícil (NEWMAN, 2015).
Segurança	Pode ser mais complexa exigindo uma abordagem granular e adaptável para cada serviço, combinada com a aplicação de medidas de segurança de forma abrangente em todo o sistema (KAZANAVIČIUS; MAŽEIKA, 2020).	Segurança pode ser mais simplificada (KAZANAVIČIUS; MAŽEIKA, 2020).
Testes	Facilita testes independentes e isolados (KAZANAVIČIUS; MAŽEIKA, 2020).	Testes podem ser mais complexos e demorados (KAZANAVIČIUS; MAŽEIKA, 2020).
Vender no formato SaaS	Permite maior flexibilidade em relação à venda no formato SaaS, pois cada serviço pode ser vendido individualmente (NEWMAN, 2015).	Mais limitada (FOWLER, 2015).

Fonte: autor.

Embora a adoção de microsserviços por MEs de *software* apresente desafios, essa arquitetura vem se consolidando como uma tendência crescente nas empresas, comprovada por diversas pesquisas recentes. De acordo com a pesquisa da Forrester Research (2019), mais da metade das empresas já estava implementando ou planejando implementar microsserviços em suas operações. Além disso, a Grand View Research (2021) prevê que o mercado global de microsserviços crescerá a uma taxa composta anual de 23,4% entre 2021 e 2028. A pesquisa "*State of the Cloud* 2021" da Flexera (2021), que entrevistou 750 profissionais de TI de empresas de diversos setores e regiões geográficas, também destacou que mais da metade das empresas pesquisadas está usando ou planejando usar microsserviços em seus aplicativos.

Esses dados reforçam a importância do uso de microsserviços para melhorar a eficiência e a produtividade das empresas, e, consequentemente, para aumentar a sua competitividade no mercado.

2.4. O USO DE MICROSSERVIÇOS PELAS MES: DESAFIOS E OPORTUNIDADES

A tendência para a adoção de microsserviços é bastante forte quando se trata de MEs. Segundo a pesquisa "*State of Microservices 2021*" da Camunda (2021), 92% das empresas pesquisadas já estão usando ou planejando usar microsserviços em um futuro próximo. Isso pode estar associado aos benefícios que a adoção de microsserviços pode trazer, como escalabilidade, flexibilidade, facilidade de manutenção e redução de custos operacionais. A migração para microsserviços também pode ajudar as empresas a melhorar a eficiência e a satisfação do cliente.

De acordo com 76% dos especialistas consultados na pesquisa de campo, empresas de todos os portes podem se beneficiar ao utilizar microsserviços na nuvem em suas operações, conforme indicado na Tabela 1, disponível na seção 5.2.1. Essa afirmação é validada por pesquisas da Forrester Research (2019), da Flexera (2021) e da Grand View Research (2021), que apontam para a tendência crescente de adoção de microsserviços pelas empresas.

A adoção de microsserviços por MEs apresenta desafios significativos. Um dos principais desafios enfrentados é a complexidade da arquitetura de microsserviços. Conforme destacado por Dragoni, Sillitti e Succi (2020), a divisão de um sistema monolítico em vários serviços menores e independentes pode resultar em maior complexidade na infraestrutura e no gerenciamento dos serviços. Isso requer um conhecimento técnico mais avançado e recursos adicionais para a implementação e a manutenção dos microsserviços. Outro desafio comum é a necessidade de uma mudança cultural e organizacional dentro da ME; a adoção de microsserviços requer uma mentalidade voltada para a colaboração, a agilidade e a automação. Isso pode exigir uma reestruturação dos processos internos e a capacitação dos funcionários para trabalhar de forma mais integrada e orientada a serviços (KALSKE; MÄKITALO; MIKKONEN, 2018).

No entanto, a adoção de microsserviços também oferece oportunidades significativas para MEs. Sathishkumar e Kavitha (2020) destacam que uma das principais oportunidades está relacionada à escalabilidade. Os microsserviços permitem que as empresas dimensionem seus serviços de forma independente, permitindo um crescimento mais flexível e ágil. Isso pode ser especialmente vantajoso para MEs que buscam expandir seus negócios de forma eficiente.

Além disso, a adoção de microsserviços pode trazer melhorias na disponibilidade e na confiabilidade dos sistemas, como apontado por Furda et al. (2018). A arquitetura de microsserviços facilita a implementação de estratégias de alta disponibilidade e tolerância a falhas, garantindo que os serviços continuem funcionando mesmo em caso de problemas em um ou mais componentes. Isso pode aumentar a confiança dos clientes e melhorar a reputação da ME no mercado.

Em resumo, a adoção de microsserviços por MEs apresenta desafios relacionados à complexidade da arquitetura e à mudança cultural, mas também oferece oportunidades de escalabilidade e melhoria na disponibilidade dos sistemas. É fundamental que as MEs considerem esses fatores ao avaliar a viabilidade e os benefícios da adoção de microsserviços em seus negócios.

3. REVISÃO SISTEMÁTICA DA LITERATURA

Nesse capítulo são apresentadas a execução da revisão sistemática da literatura (RSL) efetuada em trabalhos acadêmicos existentes sobre os assuntos correlatos ao tema da pesquisa e a análise dos artigos selecionados nas bases digitais.

3.1. EXECUÇÃO DA REVISÃO SISTEMÁTICA DA LITERATURA

A RSL foi realizada no tema “migração de sistemas monolíticos para Microsserviços com Computação em Nuvem em MEs”, utilizando o *framework Preferred Reporting Itens for Systematic Review and Meta-Analysis* (PRISMA), (PRISMA, 2015).

Seguindo as etapas descritas no *framework* e com objetivo de responder à pergunta de pesquisa “Como apoiar as MEs de *software* na migração de sistemas com arquitetura monolítica para uma arquitetura baseada em microsserviços em um ambiente de Computação em Nuvem?”, foram utilizados os constructos apresentados no Quadro 4.

A partir desses constructos, foram realizadas as buscas nas bases digitais IEEE, Scopus e Web of Science nos trabalhos publicados e que passaram pela revisão por pares. Esses trabalhos foram analisados e considerados na pesquisa incluindo o título do artigo, o resumo, a introdução e a conclusão. Como critério temporal, utilizou-se o período de 7 anos (2016 a 2023), que permitiu observar desde o primeiro artigo encontrado sobre migração de SI com arquitetura monolítica para microsserviços até a data deste trabalho.

Na prospecção realizada, utilizou-se o caractere asterisco (*) para que a busca fosse a mais abrangente possível. Expressões como “*software architecture*” foram desmembradas em dois constructos com o objetivo de ampliar os resultados.

Quadro 4 – Termos da pesquisa dos constructos

Identificação do constructo	Termo em português	Termo em inglês
C1	software*	software*
C2	arquitetura*	architecture*

C3	microsserviço*	microservice*
C4	monolít*	monolith*
C5	nuvem*	cloud*
C6	microempresa	“small business”
C7	microempresa	smb
C8	microempresa	sme

Fonte: autor.

Foram realizados quatro tipos de consulta, cujos resultados são apresentados no Quadro 5.

Quadro 5 – Resultados da consulta dos constructos

Consulta	***String***	**IEEE**	**Scopus**	**WoS**	**Total**
Consulta 1	C1 *and* C2 *and* C3 *and* C4 *and* C5 *and* C6	1	0	0	1
Consulta 2	C1 *and* C2 *and* C3 *and* C4 *and* C5	70	96	78	244
Consulta 3	C1 *and* C2 *and* C3 *and* C4 *and* C5 *and* C7	0	0	0	0
Consulta 4	C1 *and* C2 *and* C3 *and* C4 *and* C5 *and* C8	0	0	0	0
Total de resultados obtidos na pesquisa					245

Fonte: autor.

Utilizando o constructo C6 (“*small business*”), somente um artigo foi encontrado nas bases pesquisadas (Consulta 1). Removendo-se esse constructo, foram encontrados 244 artigos (Consulta 2). O termo “*cloud computing*” foi abreviado para “cloud*”, pois, utilizando toda a expressão, 23 artigos relevantes eram excluídos

do resultado. Dessa forma, a Consulta 2 foi adotada nesta pesquisa, pois o artigo encontrado na Consulta 1 está contido na Consulta 2. Além disso, a Consulta 3 não retornou nenhum artigo, e a Consulta 4 também não obteve nenhum resultado.

Os 244 estudos obtidos estão distribuídos nas bases de dados pesquisadas da seguinte forma: IEEE = 70; Scopus = 96 e Web of Science = 78. A partir dos resultados obtidos nas buscas, foram aplicados os critérios de inclusão e exclusão apresentados por Liao et al. (2017), listados no Quadro 6.

Quadro 6 – Critérios de inclusão e exclusão de trabalhos considerados na pesquisa

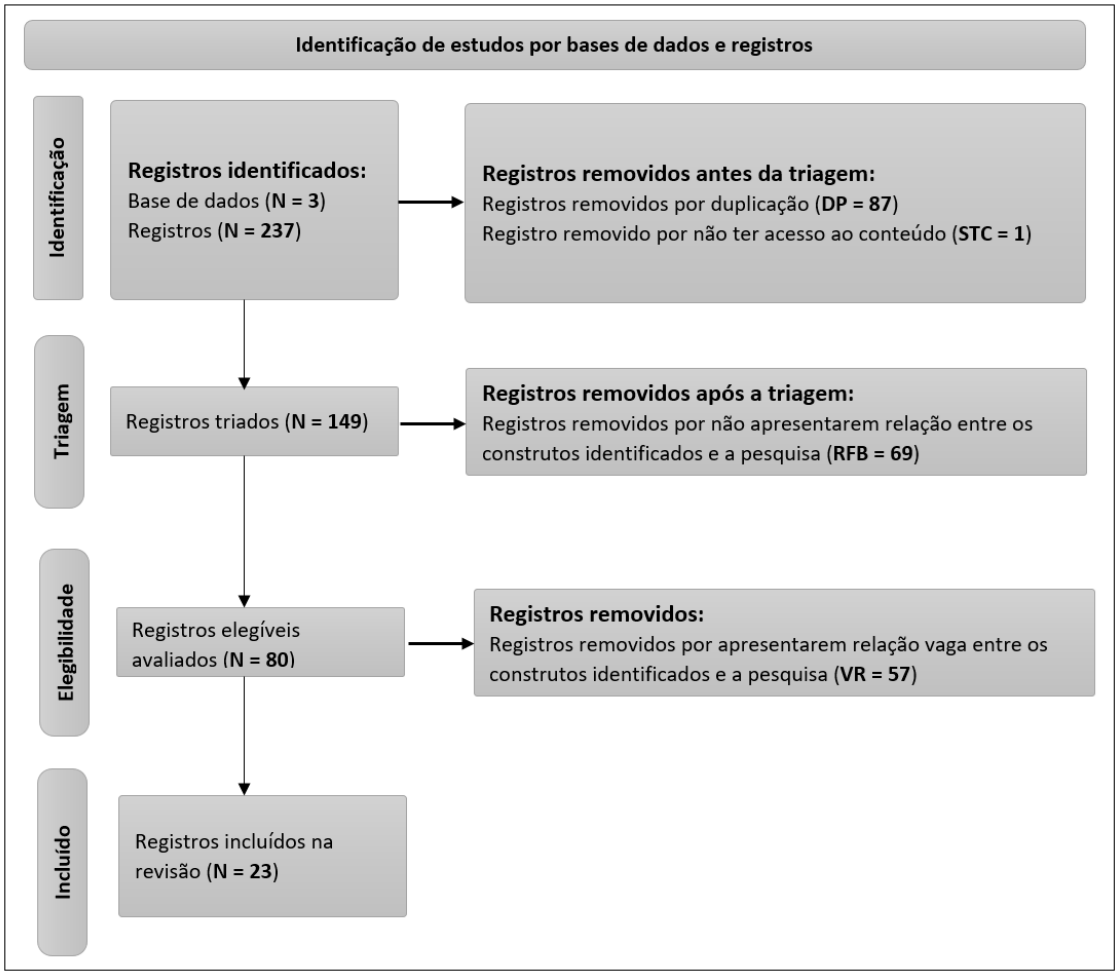
I/E	Critérios	Comentários
E	RFB	O artigo tem os constructos apenas no título, no *abstract* ou nas palavras-chave, mas não tem relação com o texto.
E	STC	Texto completo inacessível.
E	NR	Os artigos que retornaram não estão ligados ao tema proposto no artigo em nenhum momento.
E	DP	Os artigos que retornaram foram identificados em mais de uma base digital.
I	VR	VR1: migração de monolítico para microsserviços é usado como exemplos.
I	PR	PR1: detalham o processo de migração de monolítico para microsserviços. PR2: *framework*/metodologia para a migração de monolítico para microsserviços. PR3: estudo de caso de migração de monolítico para microsserviços. PR4: extração de microsserviços de um monolítico.
I	TR	Os esforços da pesquisa são explicitamente dedicados à relação a este artigo.

I: inclusão; E: exclusão; NR: não relacionado; DP: duplicado; VR: vagamente relacionado; PR: parcialmente relacionado; TR: totalmente relacionado.

Fonte: adaptado de Liao et al. (2017).

A aplicação do método PRISMA resultou em 20 artigos selecionados dos trabalhos acadêmicos pesquisados, conforme as fases propostas por Moher et al. (2010) e mostradas na Figura 11.

Figura 11 – Apresentação gráfica do modelo científico de pesquisa – Método PRISMA



Fonte: adaptado de Moher et al. (2010).

Na fase Identificação, como mostra a Figura 11, a busca nas bases digitais recuperou 244 registros utilizando-se os constructos C1 a C8. Desses registros, 87 eram duplicados e 1 estava com o seu conteúdo indisponível, resultando em 156 registros para a aplicação da fase Triagem.

Na fase Triagem, na qual são aplicados os critérios de inclusão e exclusão, conforme as regras de Liao et al. (2017), verificou-se que 69 artigos não atenderam ao critério de relação com o tema, o que resultou em 84 artigos para a aplicação da fase Elegibilidade.

Na fase Elegibilidade, na qual são aplicados os critérios de inclusão e exclusão, conforme as regras de Liao et al. (2017), verificou-se que 64 artigos têm uma vaga relação com o tema, o que resultou em 20 artigos para a aplicação da fase Inclusão.

Na fase Inclusão, restaram 20 artigos que são parcialmente relacionados (PR) ao tema deste trabalho, pois abordam migração de arquitetura monolítica para arquitetura baseada em microsserviços. Todavia, esses artigos, em uma análise de conteúdo, não descrevem detalhadamente as etapas e os critérios utilizados, tampouco abordam a questão da viabilidade da implementação em MEs de *software*. Esses artigos estão listados no Quadro 7.

Quadro 7 – Lista dos artigos selecionados na Revisão Sistemática da Literatura

Ano	Autor(es)	Título
2016	Balalaie, Heydarnoori e Jamshidi	Migrating to Cloud-Native Architectures Using Microservices: An Experience Report
2017	Villamizar et al.	Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures
2017	Fan e Ma	Migrating Monolithic Mobile Application to Microservice Architecture: An Experiment Report
2017	Acevedo, Gómez y Jorge e Patiño	Methodology to transform a monolithic software into a microservice architecture
2018	Furda et al.	Migrating enterprise legacy source code to microservices: on multitenancy, statefulness, and data consistency
2018	Kalske; Mäkitalo e Mikkonen	Challenges When Moving from Monolith to Microservice Architecture
2018	Mishra; Kunde e Nambiar	Cracking the monolith: challenges in data transitioning to cloud native architectures
2019	Lauretis	From monolithic architecture to microservices architecture
2019	Silva, Carneiro e Monteiro	Towards a roadmap for the migration of legacy software systems to a microservice based architecture

2020	Prasandy, Murad e Darwis	Migrating application from monolith to microservices
2020	Bajaj et al.	Partial migration for re-architecting a cloud native monolithic application into microservices and FaaS
2020	Kazanavičius e Mažeika	Analysis of legacy monolithic software decomposition into microservices
2021	Auer et al.	From monolithic systems to Microservices: An assessment framework
2021	Wolfart et al.	Modernizing legacy systems with microservices: A roadmap
2021	Freire et al.	Migrating production monolithic systems to microservices using aspect oriented programming
2021	Brito, Cunha e Saraiva	Identification of microservices from monolithic applications through topic modelling
2021	Haugeland et al.	Migrating Monoliths to Microservices-based Customizable Multi-tenant Cloud-native Apps
2021	Mendonça et al.	The Monolith strikes back: Why istio migrated from microservices to a monolithic architecture
2021	Gravanis; Kakarontzas e Gerogiannis	You don’t need a Microservices Architecture (yet): Monoliths may do the trick
2021	Balzamov et al.	Development of a methodology for migration of monolithic systems to microservice architecture using cloud technologies

Fonte: autor.

3.2. ANÁLISE DOS ARTIGOS SELECIONADOS

A seguir serão analisados detalhadamente os 20 artigos selecionados que fundamentam esta pesquisa e que estão listados no Quadro 7, em ordem crescente de ano de publicação.

Balalaie, Heydarnoori e Jamshidi (2016) apresentam um estudo de caso de migração de um *software* monolítico para microsserviços em nuvem, no qual é sugerido que a migração seja realizada de forma incremental e em várias etapas,

sem impactar os usuários finais do sistema. O estudo propõe uma nova arquitetura baseada em entidades em forma de serviço e destaca diversas características essenciais para uma implantação efetiva, como a necessidade de implantação automatizada, governança de dados descentralizada e escalabilidade integrada. Além disso, os autores enfatizam que a introdução de novos serviços traz consigo novos requisitos não funcionais e que equipes sem experiência enfrentariam grandes dificuldades para superar os desafios apresentados. No entanto, apesar dessas observações, não é fornecido um guia de orientação para ajudar equipes sem experiência e pequenas empresas de *software* a enfrentar esse processo de migração.

Villamizar et al. (2017) comparam o custo de executar um sistema monolítico e um sistema de microsserviços na nuvem. Afirmam que para obter benefícios com a adoção de microsserviços, é necessário também implantar a cultura DevOps. Caso contrário, pode-se enfrentar lentidão no processo. Além disso, destacam que a adoção de arquiteturas de microsserviços exige uma nova cultura de desenvolvimento, que deve ser complementada por um conjunto de diretrizes e boas práticas a nível empresarial. Nesse sentido, a adoção de microsserviços deve ser implementada como uma estratégia de negócios de longo prazo e não deve ser vista como um projeto isolado, já que sua adoção requer esforços e habilidades que devem ser desenvolvidos de forma incremental. Concluem que o uso de serviços de nuvem permite que as empresas reduzam seus custos de infraestrutura em até 77,08%. Por fim, recomendam que a agilidade, a redução de custos e a escalabilidade granular sejam equilibradas com os esforços de desenvolvimento, os desafios técnicos e os custos incorridos pelas empresas resultantes de microsserviços. Essa implementação exige a adoção de novas práticas, processos e metodologias.

Fan e Ma (2017) acompanharam a migração de uma aplicação mobile com arquitetura monolítica para arquitetura de microsserviços e geraram um relatório com a análise e os resultados. O objetivo da migração dessa aplicação foi melhorar a escalabilidade e a capacidade de manutenção do sistema. Utilizaram uma abordagem gradual (iterativa) da migração para manter a estabilidade geral do sistema. Constataram alguns problemas, como a identificação incorreta de serviços, a atribuição inadequada de responsabilidade e a incompatibilidade de interface. Concluem que, em um sistema baseado em microsserviços, é preferível que cada

serviço use um banco de dados distinto. Destacam que, para usufruir dos benefícios dos microsserviços em nuvem, o processo deve ser automatizado e um *pipeline* deve ser definido. O microsserviço deve ser especializado, simples e tolerante a falhas. Evidenciam que as configurações de ambiente são complexas e exigem mais ferramentas para obter flexibilidade de arquitetura. Por fim, concluem que isso aumenta os custos e a complexidade do sistema.

Acevedo, Gómez y Jorge e Patiño (2017) apresentam uma metodologia dividida em seis etapas para transformar um sistema com arquitetura monolítica em microsserviços. A primeira etapa é o Planejamento e Viabilidade, na qual se determina o nível de maturidade da aplicação a ser migrada. Na segunda etapa, Elicitação e Análise de Requisitos, é utilizado o método ágil Scrum junto a outras técnicas, como entrevistas e especificação de requisitos. Na terceira etapa, Design e Definição de Microsserviços, é utilizada a técnica DDD, que propõe a definição de contextos compostos de componentes do modelo de negócio que sejam consistentes entre si. Na quarta etapa, Desenvolvimento de Microsserviços, é elaborado o microsserviço com o gerenciamento dos conceitos necessários para que a arquitetura seja implementável. Na quinta etapa, Integração de Testes, é destacado que a arquitetura de microsserviços facilita o refinamento dos limites da lógica de negócios, permitindo que os testes de unidade sejam isolados. Por fim, na sexta etapa, Contratos e Ponto a Ponto e Implantação, são apresentadas as possibilidades de instalação em máquinas virtuais ou *containers* e destacada a importância da integração e da entrega contínua.

Furda et al. (2018) apresentam um estudo focado na extração de microsserviços, com destaque para o estudo de estado de sessão deles. De acordo com os autores, cada microsserviço deve ter o seu próprio banco de dados, que não deve ser compartilhado diretamente com nenhum outro serviço. Os microsserviços que guardam o estado dos objetos, como as atividades de compras anteriores e visitas anteriores à loja *on-line*, por exemplo, são chamados de microsserviços com estado. Os autores afirmam que os microsserviços sem estado são mais indicados, pois exploram benefícios da Computação em Nuvem, tais como: elasticidade sob demanda, alta disponibilidade e redução do tempo médio de resposta, distribuindo a carga entre os microsserviços disponíveis. Além disso, a redundância aumenta a confiabilidade do sistema e o balanceador de carga pode aumentar a probabilidade de receber uma resposta bem-sucedida de um dos microsserviços disponíveis, sem

a necessidade de sincronizar ou replicar o estado em um banco de dados de sessão.

Kalske, Mäkitalo e Mikkonen (2018) apresentam os desafios de migrar um SI de estrutura monolítica para microsserviços e afirmam que a abordagem monolítica é suficiente no início e é possível que o tamanho da base de código e a necessidade de dimensionamento refinado nunca sejam necessários. O que significa que é melhor ficar com o monolito e evitar os desafios técnicos e organizacionais que a arquitetura de microsserviços traz. Destacam o desafio de encontrar e definir quais partes do *software* existente devem ser divididas em microsserviços e quais são bons candidatos para microsserviços. É fundamental para o sucesso do projeto e requer um bom conhecimento sobre os casos de uso de negócio. Afirmam que a cultura DevOps precisa ser adotada, pois com a adoção de microsserviços as equipes precisam implantar, monitorar e resolver problemas também na produção. As equipes precisam configurar seus próprios *pipelines* de integração contínua (CI) e entrega contínua (CD).

Mishra, Kunde e Nambiar (2018) realizaram um estudo sobre os desafios associados à transição de dados para arquitetura em nuvem e destacam a flexibilidade que o uso de microsserviços traz, principalmente em relação às possibilidades de se utilizar diversas tecnologias em uma mesma solução, visto que cada microsserviço pode ser desenvolvido em uma linguagem específica. São vários os fatores que precisam ser levados em consideração nesse processo de migração, tais como: o número de funções de negócios coesas presente no aplicativo; os recursos humanos disponíveis para desenvolver e manter os microsserviços; a interface de mensagem entre os componentes; o acesso aos dados, evitando dependência de vários microsserviços no mesmo banco de dados; a escalabilidade; e os requisitos de escala de cada microsserviço. Os autores também abordam a migração de banco de dados relacional para banco de dados não relacional, como MongoDB e Cassandra,[3] ou armazenamento baseado em objeto, o que pode ser considerado necessário para suportar o volume e a velocidade dos dados. Bancos de dados não relacionais também são adequados aos requisitos de esquema de dados em evolução de um microsserviço.

---

3 MongoDB e Cassandra são bancos de dados NoSQL, ou seja, são criados para modelos de dados específicos e têm esquemas flexíveis para a criação de aplicativos modernos.

Lauretis (2019) propõe uma estratégia para a migração de um sistema com arquitetura monolítica para uma arquitetura baseada em microsserviços, utilizando como base o conceito de funcionalidades de negócio. A estratégia é composta de cinco fases: análise de funções, identificação de funcionalidades de negócio, análise de funcionalidades de negócio, atribuição de funcionalidades de negócio e criação de microsserviços. Porém, como se trata de uma pesquisa em andamento, os autores não detalham como realizar cada etapa de forma clara e prática.

Silva, Carneiro e Monteiro (2019) realizaram dois estudos, um piloto e um estudo de caso, com o objetivo de caracterizar as etapas e os desafios relevantes durante a migração de sistemas legados (monolíticos) para microsserviços. Identificaram quatro principais desafios enfrentados durante a migração: a identificação de funcionalidades no legado (que não é uma tarefa fácil), a definição dos limites ideais entre os recursos candidatos para microsserviços, a decisão de quais recursos devem ser convertidos em microsserviços e a análise dos microsserviços candidatos em potencial em relação à sua granularidade e coesão. Concluíram que a migração de um aplicativo legado é uma tarefa que exige esforço significativo e é, muitas vezes, complexa. Destacaram a existência de *frameworks*, como o Hystrix[4], que podem ser usados para apoiar os profissionais durante o desenvolvimento de sistemas baseados em microsserviços, mas nenhum deles oferece suporte completo para a realização da migração.

Prasandy, Murad e Darwis (2020), em seu artigo sobre a migração de sistemas monolíticos para microsserviços, afirmam que a iniciativa pode gerar problemas relacionados aos processos de negócios e propõem soluções para superá-los, como a modularização do código-fonte, do banco de dados e dos servidores em nuvem. Eles destacam cinco etapas a serem seguidas: avaliar e listar os recursos em aplicativos usados com frequência, refatorar o banco de dados, refatorar o código-fonte, testar as APIs REST e usar o *upload* para o servidor com tráfego dividido para evitar possíveis falhas durante o processo de atualização. No entanto, os autores não detalham de forma clara e prática como executar cada etapa e suas soluções se limitam a questões técnicas.

4 Hystrix é uma biblioteca de tolerância a falhas e latência projetada para isolar pontos de acesso em sistemas remotos, serviços e bibliotecas de terceiros.

Bajaj et al. (2020) propõem um modelo de migração parcial de caixa preta com base no log de acesso à web para identificar padrões de carga de trabalho de diferentes módulos em um aplicativo web nativo da nuvem. As dimensões consideradas para a identificação da carga de trabalho de um módulo são sua escalabilidade e requisitos de recursos. Fazem um estudo de caso aplicando tal modelo no *software* "*Teachers Feedback*", originalmente desenvolvido como um aplicativo monolítico. A abordagem é baseada na mineração de logs de acesso ao servidor web para prever a escalabilidade e os requisitos de recursos de seus vários componentes. O modelo proposto tem três etapas: Pré-processamento do Log de Acesso à Web, Segregação de URIs e Identificação de modelos de serviço. Concluem que os aplicativos *serverless* (sem servidor) são totalmente gerenciados pelo próprio provedor de nuvem. Ao refatorar um aplicativo monolítico para melhor escalabilidade e utilização de recursos, alguns componentes do aplicativo podem ser reorganizados como microsserviços e alguns como FaaS (Funções como serviço) para garantir melhor utilização dos recursos. O modelo aplicado no "*Teachers Feedback*" apresentou um resultado satisfatório ao mapear os componentes que melhor se adaptam aos padrões de serviço monolítico/microsserviços/FaaS.

Kazanavičius e Mažeika (2020) realizaram um estudo de caso aplicando três técnicas mapeadas na literatura para a migração de um SI monolítico para microsserviços. Os métodos são baseados em armazenamento, código e domínio de negócios. O método baseado no domínio de negócios é o menos formal e o mais universal, porém exige forte conhecimento do domínio de negócios e detalhes técnicos de implementação. O método baseado em código é o mais formal, mas requer ferramentas adicionais para gerar grafos representando dependências de classes. Já o método baseado em armazenamento não requer nenhuma ferramenta adicional para implementar, mas requer algum conhecimento do domínio de negócios. O Método Baseado no Domínio de Negócios utiliza o conceito de *Domain-Driven Design* (DDD) para identificar os microsserviços candidatos. Essa técnica permite entender o negócio e mapeá-lo em objetos, agregados, serviços e outros elementos que podem ser utilizados para compor os microsserviços. É importante ressaltar que o DDD é uma técnica que exige grande conhecimento do domínio de negócios e, portanto, deve ser aplicado por profissionais especializados. Os resultados do estudo indicaram que a escolha do método depende do conhecimento do implementador, da disponibilidade de ferramentas e da complexidade do SI

monolítico. Cada método tem suas vantagens e desvantagens, e a escolha deve levar em consideração as particularidades do projeto. Além disso, é importante destacar que a migração de um SI monolítico para microsserviços é uma tarefa complexa que exige planejamento cuidadoso, conhecimento técnico e de negócios, além da utilização de boas práticas de engenharia de *software*.

Auer et al. (2021) realizaram uma RSL com o objetivo de coletar os indicadores mais relevantes relacionados a microsserviços e, em seguida, propuseram um *framework* de suporte à decisão baseada em evidências para empresas que pensam em migrar para microsserviços. Destacam um conjunto de características e métricas que devem ser coletadas antes de rearquitetar o sistema monolítico. Para validar esses indicadores, realizaram um levantamento feito na forma de entrevistas com especialistas para derivar o referencial de avaliação baseado na Teoria Fundamentada nos Dados. Fornecem um conjunto composto de informações e métricas que as empresas podem utilizar para decidir se devem migrar ou não para microsserviços. O artigo destaca o uso do DDD como uma abordagem que pode auxiliar na identificação das fronteiras de cada microsserviço e na definição da comunicação entre eles.

Wolfart et al. (2021) destacam que, embora existam trabalhos que abordem a modernização de sistemas legados para microsserviços, muitos deles não levam em consideração aspectos importantes do mundo real, como aspectos organizacionais, operacionais e técnicos, o que pode tornar o uso de microsserviços uma estratégia para modernização de sistemas legados mais complexos. Diante disso, os autores propõem um roteiro composto de oito atividades que podem ser realizadas de forma incremental durante o processo de migração. Essas atividades estão distribuídas nas fases de iniciação, planejamento, execução e monitoramento, e visam cobrir uma ampla gama de aspectos, desde a definição dos objetivos do projeto até a monitoração dos microsserviços implantados. Os autores destacam ainda que o roteiro é composto de atividades de alto nível e pode ser utilizado como ponto de partida para a análise e a adaptação do roteiro às necessidades específicas de cada empresa. No entanto, eles afirmam que o roteiro não foi validado por especialistas e que mais pesquisas são necessárias para avaliar a efetividade do roteiro proposto.

Freire et al. (2021) apresentam uma abordagem para migrar sistemas legados para microsserviços com tempo de inatividade quase zero, utilizando programação orientada a aspectos (AOP) e reflexão para interceptar chamadas no sistema

monolítico e convertê-las em solicitações de serviço para os microsserviços. Para validar a abordagem, eles desenvolveram duas aplicações como provas de conceito e demonstraram que é possível ir para frente ou para trás entre diferentes versões do aplicativo com alterações mínimas de código ou dados. Concluíram que a abordagem apresenta resultados promissores, mantendo custos semelhantes e exigindo menos esforço de codificação do que uma abordagem gerada por especialistas. Além disso, destacam que é possível reverter a implementação para a versão monolítica anterior sem modificar o código ou os dados e sem interromper o aplicativo, o que é uma contribuição importante.

Brito, Cunha e Saraiva (2021) propõem uma nova abordagem para identificar microsserviços em sistemas legados utilizando modelagem de tópicos. A metodologia utiliza técnicas de *clustering* para produzir um conjunto de serviços com base no *software* original. Para validar a abordagem, é implementada uma ferramenta *open source* para exploração de arquiteturas monolíticas e identificação de microsserviços. Além disso, é realizada uma análise quantitativa utilizando métricas de independência de funcionalidade e modularidade de serviços, aplicada em 200 projetos de código aberto do GitHub. Os resultados demonstram uma identificação benéfica dos serviços, com resultados gerais positivos das métricas aplicadas. No entanto, os autores destacam que, entre os projetos coletados, nem todos têm arquitetura monolítica, o que pode ameaçar a validade dos resultados obtidos.

Haugeland et al. (2021) apresentam uma abordagem para migrar SIs monolíticos para microsserviços em nuvem, permitindo a flexibilização de um mesmo serviço para clientes distintos com comportamentos diferentes. A migração é dividida em três etapas: a primeira consiste em analisar e dividir a aplicação em contextos delimitados, identificando as funcionalidades que podem ser agrupadas em um mesmo microsserviço; a segunda etapa é a transformação da infraestrutura para se adequar à estrutura de microsserviços, envolvendo a criação de bancos de dados relacionados aos contextos e a configuração de componentes adicionais como o gateway de API e a troca de mensagens; a terceira etapa é a implementação da funcionalidade dos contextos como serviços separados e conectá-los à infraestrutura, na qual os serviços são separados em microsserviços e configurados para se comunicarem com a infraestrutura. Após a migração, adiciona-se a infraestrutura necessária para multilocação, com a criação de bases distintas para

cada cliente para garantir o isolamento suficiente dos dados de cada cliente. Os autores destacam que a personalização não se limita aos recursos de computação, pois o aplicativo principal e os dados de outros clientes permanecem totalmente isolados do código de personalização. Essa abordagem permite que diferentes clientes compartilhem a mesma infraestrutura de microsserviços sem comprometer a segurança e a privacidade dos dados. Concluem que sua proposta apresenta uma solução para a flexibilização de serviços para clientes distintos com comportamentos distintos.

Mendonça et al. (2021) realizaram um estudo de caso do projeto Istio, lançado em 2017 em uma colaboração entre Google, IBM e Lyft. O Istio adotou inicialmente uma arquitetura de microsserviços em seu plano de controle. No entanto, menos de 3 anos após seu lançamento, os microsserviços foram consolidados em um monolito devido ao fato de muitos dos benefícios normalmente associados aos microsserviços, como distribuição independente e escala independente, não se aplicarem ao Istio. Isso resultou em custos elevados devido à maior complexidade operacional inerente a uma arquitetura de microsserviços, sem se beneficiar dela. A equipe concluiu que o ganho com o uso dos microsserviços não era maior do que o custo de ter que orquestrá-los durante a configuração e a operação, destacando a importância de considerar os prós e os contras ao decidir pela adoção de uma arquitetura de microsserviços.

Gravanis, Kakarontzas e Gerogiannis (2021) relatam a adoção excessiva da arquitetura de microsserviços no mundo da tecnologia, muitas vezes sem considerar se tal arquitetura é realmente viável para a organização ou se trará benefícios em relação aos custos de migração. O estudo destaca que um sistema monolítico bem desenvolvido pode fornecer métodos diretos para adicionar novos recursos e refatorar a base de código existente sem efeitos colaterais indesejados, o que indica uma implementação inadequada em vez de uma falha inerente à arquitetura monolítica. Os autores concluem que a falta de informações substanciais e determinísticas sobre a arquitetura de microsserviços pode levar a uma perspectiva positiva pré-estabelecida sobre ela em comparação com a arquitetura monolítica, mas para fazer uma comparação mais completa e declarar explicitamente qual arquitetura é superior, é necessário considerar aspectos como o processo de seleção de arquitetura inicial, o ambiente de desenvolvimento, as decisões de negócios, entre outros. Em resumo, a adoção de uma arquitetura de microsserviços

não é a melhor solução em todos os cenários e deve ser avaliada com cuidado em relação aos custos e benefícios para a organização.

Balzamov et al. (2021) explicam que a migração de um sistema monolítico para uma arquitetura de microsserviços é uma decisão que deve ser cuidadosamente avaliada e considerada em relação aos critérios apresentados. Eles destacam a importância de ter uma base de código grande e sem suporte, pois isso pode dificultar a manutenção e a atualização do sistema. Além disso, a falta de flexibilidade para introduzir novas tecnologias e fazer pequenas atualizações e alterações pode ser um sinal de que a arquitetura atual não está mais atendendo às necessidades da organização. A existência de forte acoplamento entre as classes de um aplicativo monolítico, a dificuldade em realizar testes automatizados e a integração problemática de módulos e soluções em outras linguagens de programação são outros critérios que devem ser considerados antes de decidir pela migração. No entanto, os autores também alertam que a adoção de uma arquitetura de microsserviços requer conhecimento e experiência em tecnologias específicas de conteinerização e orquestração. Eles enfatizam que a existência de microsserviços sem esse ecossistema é geralmente impossível. Portanto, é importante ter especialistas em certas tecnologias para garantir que a adoção de microsserviços traga os benefícios esperados, como distribuição independente, escala independente e isolamento de segurança.

3.3 CONCLUSÃO DA ANÁLISE DOS ARTIGOS SELECIONADOS NA REVISÃO SISTEMÁTICA DA LITERATURA

É importante ressaltar que a falta de publicações sobre a migração de sistemas com arquitetura monolítica para microsserviços aplicada em MEs é uma lacuna na literatura acadêmica pesquisada. Esse cenário ressalta a necessidade de pesquisas que explorem a viabilidade e os desafios dessa migração específica.

Considerando que as MEs representam grande parte da economia, a ausência de estudos dedicados a esse segmento é ainda mais importante. Portanto, a pesquisa realizada neste trabalho tem como objetivo preencher essa lacuna e contribuir para o desenvolvimento de soluções que atendam às necessidades das MEs que desejam adotar microsserviços.

4. METODOLOGIA DE PESQUISA APLICADA

Nesta seção são apresentados os fundamentos metodológicos, as etapas da pesquisa, os instrumentos de pesquisa e as orientações propostas para a composição do guia, bem como são descritos os métodos adotados na condução desta dissertação.

4.1. FUNDAMENTOS METODOLÓGICOS

Esta dissertação é de caráter exploratório e abordagem qualitativa, visto que foi realizada uma pesquisa teórica, nas literaturas nacional e internacional, de como apoiar as MEs de *software* na migração de sistemas de informação com arquitetura monolítica para uma arquitetura baseada em microsserviços em um ambiente de Computação em Nuvem.

A partir dessa pesquisa, propõem-se um guia de orientações que apoiará as MEs brasileiras nessa empreitada e que será avaliado, em uma pesquisa de campo, utilizando-se o método de pesquisa Delphi (*Survey* controlado), com aplicação de questionário como instrumento de pesquisa e método de coleta de dados.

4.2. ETAPAS DA PESQUISA

Este trabalho foi organizado em cinco etapas, de acordo com Forza (2002):

- Etapa 1: execução de uma pesquisa nas bases digitais para os constructos da pesquisa e apresentação dos principais autores e conceitos envolvidos com o tema da pesquisa, apresentadas na seção 2.
- Etapa 2: elaboração de uma RSL nas bases digitais executada na seção 3. Dessa forma, foi possível identificar as definições sobre migração de SIs da arquitetura monolítica para a arquitetura de microsserviços abordadas na literatura selecionada. Como resultado dessa revisão foram selecionadas e classificadas as orientações sobre a migração da arquitetura monolítica para a arquitetura em microsserviços em nuvem (item 4.3). Essas orientações classificadas pela curva ABC serão a base do guia e de um questionário como instrumento de pesquisa a ser apresentado (itens 4.4 e 4.5), tanto no

teste piloto para ser validado por especialistas em microsserviços de *software* do mercado brasileiro quanto na execução do método Delphi.

- Etapa 3: projeto piloto (apresentado no item 4.6), que permitirá testar o instrumento de pesquisa (questionário), verificar a qualidade dos dados apurados e fazer eventuais ajustes antes da aplicação do instrumento em campo.
- Etapa 4: Método Delphi – aplicação do instrumento de pesquisa do tipo questionário, em um *Survey* controlado (método Delphi) — apresentado no item 4.7 — e executado na seção 5, para especialistas em desenvolvimento de *software* do mercado brasileiro.
- Etapa 5: desenvolvimento e validação do guia de orientações para as MEs de *software* na migração de sistemas de informação com arquitetura monolítica para arquitetura baseada em microsserviços em um ambiente de Computação em Nuvem (seção 6).

4.3. SELEÇÃO E CLASSIFICAÇÃO DAS ORIENTAÇÕES A PARTIR DA LITERATURA

A RSL realizada na condução e apresentada na seção 3 desta dissertação permitiu identificar as orientações constantes nos artigos selecionados sobre migração de SIs com arquitetura monolítica para arquitetura de microsserviços. Essas diretrizes serviram como base para a criação do guia de orientações.

Os critérios de inclusão e de exclusão do método PRISMA aplicado selecionaram 20 artigos acadêmicos relacionados ao tema. Foram identificadas 21 orientações que permeiam a migração de arquitetura nos SIs e que podem ser observadas no Quadro 8, em ordem de ocorrência.

Quadro 8 – Orientações para a migração identificadas na literatura

Identificação	Orientação	Ocorrências
a	Implementar DevOps (CI/CD)	9
b	Aplicar Domain Driven Design (DDD)	8
c	Segregar Banco de dados	8
d	Possuir Equipe Qualificada	8
e	Iniciar com pequenos módulos	7
f	Utilizar UML (diagrama de classe / Caso de uso)	5
g	Implementar API Gateway	5
h	Implementar Service Discovery	4
i	Implementar Circuit Breaker	4
j	Implementar Docker / Kurbenetes	4
k	Implementar Testes integrados	3
l	Granularidade do MS	3
m	Mudança cultural na empresa	3
n	Implementar Log Centralizado	3
o	Implementar Mensageria	3
p	Extração de MS automática (log servidor)	3
q	Implementar Load Balancer	2
r	Custo Monolítico	1
s	Custo MS	1
t	Scrum para elicitação	1
u	Transação distribuída	1

CI: Integração Contínua; CD: Entrega Contínua; UML: *Unified Modeling Language*; API: *Application Programming Interface*; MS: microsserviços.

Fonte: autor.

As 21 orientações observadas na revisão sistemática são classificadas, na coluna “Identificação”, por letras (de “a” a “u”), o que auxiliará na leitura do Quadro 9, que lista essas orientações e seus respectivos autores proponentes e as ocorrências nas citações.

Quadro 9 – Autores *versus* orientações

Autores	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u
Balalaie *et al* .(2016)	x	x					x	x	x								x				
Villamizar *et al.* (2017)							x					x	x					x			
Fan e Ma(2017)	x	x	x		x		x	x	x		x			x							
Acevedo *et al.* (2017)	x	x		x				x	x	x					x		x		x		
Furda *et al.* (2018)			x																		
Kalske *et al.* (2018)	x		x	x	x				x	x	x	x	x	x						x	x
Mishra *et al.* (2018)			x	x	x	x		x				x			x						
Lauretis L.(2019)																					
Silva et al.(2019)	x	x	x	x	x	x	x														
Prasandy *et al.* (2020)			x	x							x		x								
Bajaj *et al.* (2020)																x					
Kazanavicius e Mazeika (2020)		x	x	x		x															
Auer *et al.* (2021)	x	x		x	x	x															
Wolfart *et al.* (2021)	x	x		x	x	x				x				x							
Freire *et al.* (2021)																x					
Brito *et al.* (2021)																x					
Haugeland *et al.* (2021)		x			x		x								x						
Mendonça *et al.* (2021)	x										x										
Gravanis *et al.* (2021)	x		x																		
Balzamov *et al.* (2021)										x											
Total	9	8	8	8	7	5	5	4	4	4	4	3	3	3	3	3	2	1	1	1	1

Fonte: autor.

Após identificar as orientações relevantes, esta pesquisa utilizou a curva ABC, amplamente empregada nas empresas, para determinar o grau de importância de cada uma delas (BALLOU, 1993). Para isso, as orientações foram categorizadas como A, B ou C de acordo com a sua relevância: as mais importantes foram consideradas como A; as de média importância, como B; e as de baixa importância, como C.

Vale destacar que a classificação ABC é amplamente utilizada em diversas áreas, tanto profissionais quanto pessoais (ALVARENGA; NOVAES, 2000). Entretanto, segundo Pacchini (2019), é importante lembrar que os percentuais para classificação não seguem uma regra matemática fixa para todos os casos, havendo diferentes cortes de valores para cada uma das categorias (A, B e C).

Dessa forma, considerando a falta de uniformidade e referências na literatura, este estudo adotou o critério de classificação proposto por Pacchini (2019): 72% das citações foram classificadas como A; 21%, como B; e 7%, como C.

O Quadro 10 apresenta a classificação das orientações mais citadas, resultando em dez orientações relevantes classificadas como A, representando 72% do total (orientações de “a” a “j” no Quadro 10).

Quadro 10 – Orientações relevantes classificadas de acordo com a Curva ABC

Orientação	# Citações	% Total	ABC	Descrição
a	9	10,5%	A	Implementar DevOps (CI/CD)
b	8	9,3%	A	Aplicar Domain Driven Design (DDD)
c	8	9,3%	A	Segregar Banco de dados
d	8	9,3%	A	Possuir Equipe Qualificada
e	7	8,1%	A	Iniciar com pequenos módulos
f	5	5,8%	A	Utilizar UML (diagrama de classe / Caso de uso)
g	5	5,8%	A	Implementar API Gateway
h	4	4,7%	A	Implementar Service Discovery
i	4	4,7%	A	Implementar Circuit Breaker
j	4	4,7%	A	Implementar Docker / Kurbenetes
k	3	3,5%	B	Implementar Testes integrados
l	3	3,5%	B	Granularidade do MS
m	3	3,5%	B	Mudança cultural na empresa
n	3	3,5%	B	Implementar Log Centralizado
o	3	3,5%	B	Implementar Mensageria
p	3	3,5%	B	Extração de MS automática (log servidor)
q	2	2,3%	C	Implementar Load Balancer
r	1	1,2%	C	Custo Monolítico
s	1	1,2%	C	Custo MS
t	1	1,2%	C	Scrum para elicitação
u	1	1,2%	C	Transação distribuída

CI: Integração Contínua; CD: Entrega Contínua; UML: *Unified Modeling Language*; API: *Application Programming Interface*; MS: microsserviços.

Fonte: autor.

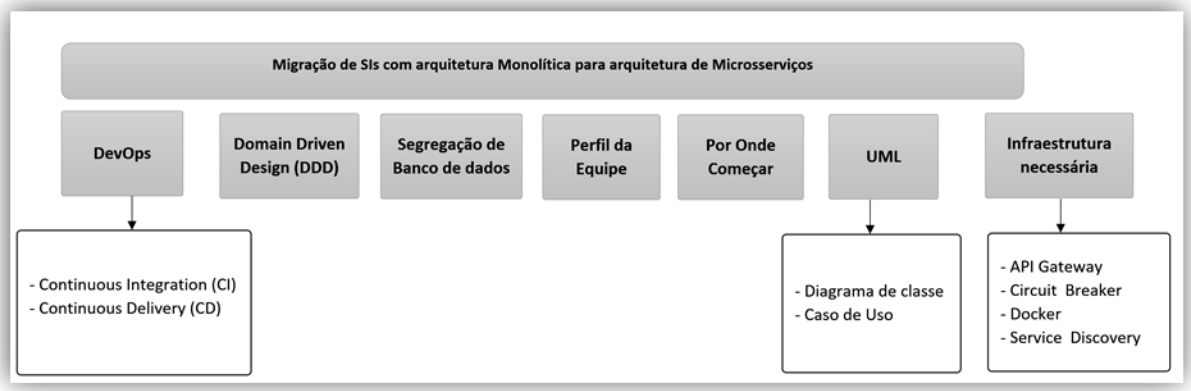
4.4. ORIENTAÇÕES RELEVANTES PROPOSTAS PARA COMPOSIÇÃO DO GUIA

Com base nos critérios de seleção adotados e para os propósitos desta dissertação, o guia será composto das orientações consideradas mais relevantes (orientações de “a” a “j” do Quadro 10) e que foram classificadas com o uso e a aplicação da Curva ABC. A Figura 12 representa graficamente o guia de orientações, validado por especialistas.

Para efeito de simplificação da estrutura do guia, as orientações serão apresentadas da seguinte maneira:

- DevOps: Orientação “a” – Implementar DevOps (CI/CD);
- *Domain Driven Design* (DDD): Orientação “b” – Aplicar *Domain Driven Design* (DDD);
- Segregação de Banco de Dados: Orientação “c” – Segregar banco de dados;
- Perfil da Equipe: Orientação “d” – Ter equipe qualificada;
- Por Onde começar: Orientação “e” – Iniciar com pequenos módulos;
- UML: Orientação “f” – Utilizar UML (Diagrama de classes/Caso de uso);
- Infraestrutura Necessária: composta das orientações “g” (Implementar API Gateway), “h” (Implementar Service Discovery), “i” (Implementar Circuit Breaker) e “j” (Implementar Docker/Kurbenetes).

Figura 12 – Estrutura do guia preliminar de orientações relevantes



SI: Sistema de Informação.

Fonte: autor.

A seguir, as orientações relevantes classificadas são conceituadas.

- **DevOps (CI/CD):** a cultura DevOps é essencial na atualidade, uma vez que as equipes de desenvolvimento precisam lidar com a implantação, o monitoramento e a solução de problemas em produção (ACEVEDO; GÓMEZ Y JORGE; PATIÑO, 2017). Ainda segundo os autores, para isso, é necessário configurar *pipelines* de CI e CD, o que também está diretamente relacionado aos microsserviços. De acordo com Wolfart et al. (2021), com a adoção dessas práticas, torna-se mais fácil gerenciar múltiplos serviços e validar suas ações. Ainda segundo os autores, as equipes precisam assumir

a responsabilidade total por seus serviços, o que pode demandar novas habilidades para implantar e solucionar problemas na produção.

Nesse sentido, a mentalidade DevOps é fundamental. Segundo Kalske, Mäkitalo e Mikkonen (2018), DevOps é um conjunto de práticas destinado a reduzir o tempo entre a confirmação de uma mudança em um sistema e a sua implantação em produção, garantindo alta qualidade. Portanto, é essencial que as equipes de desenvolvimento e operações trabalhem em conjunto e adotem uma abordagem colaborativa para atingir esse objetivo. A cultura DevOps não é apenas uma questão de tecnologia, mas também envolve mudanças culturais e organizacionais dentro das empresas. Com essa cultura, é possível promover maior agilidade, eficiência e qualidade na entrega de *software*.

- ***Domain Driven Design* (DDD):** o método utilizado para encontrar candidatos a microsserviços no sistema original baseou-se na análise de contexto limitado, que se mostrou uma ferramenta chave para identificar possíveis candidatos. Para isso, utilizou-se a abordagem DDD com o objetivo de identificar os domínios específicos presentes na solução e, a partir disso, identificar os módulos de domínio em cada um desses domínios. A análise de abordagem DDD permite a extração de microsserviços de baixo acoplamento, conforme defendido por Kazanavičius e Mažeika (2020).
- **Segregação de Banco de Dados:** a unidade fundamental de qualquer sistema é composta de seus dados, que são cruciais para o funcionamento adequado do sistema. Uma vez que o esquema de dados subjacente e seus padrões de acesso são entendidos pelo sistema, ele deve ser capaz de facilitar a quebra do monolito, evitando a dependência de vários microsserviços na mesma tabela de banco de dados ou no mesmo local do repositório de dados.

  O modelo recomendado é que cada microsserviço tenha o seu próprio repositório de dados. No entanto, a consistência dos dados torna-se um problema quando os dados são replicados no repositório local dos microsserviços ou em qualquer repositório centralizado. Lidar com problemas decorrentes da replicação de instâncias de microsserviço e seus

dados associados pode resultar em inconsistências de dados (MISHRA; KUNDE; NAMBIAR, 2018).

- **Perfil da Equipe:** para que os benefícios dos microsserviços em nuvem sejam aproveitados, é necessário atender a algumas condições, como ter uma equipe de desenvolvedores qualificados em aplicações distribuídas (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016). Além disso, um novo perfil deve ser introduzido na equipe para monitorar os microsserviços e a infraestrutura, garantindo a disponibilidade do serviço, identificando gargalos de desempenho, analisando o uso dos recursos da infraestrutura, entre outras tarefas (WOLFART et al., 2021). Dessa forma, os profissionais são capazes de avaliar e analisar informações do sistema modernizado, contribuindo para aprimorar o desempenho e a eficiência do sistema como um todo.
- **Por Onde Começar:** migrar um sistema de *software* de uma arquitetura monolítica para uma arquitetura de microsserviços não é uma tarefa trivial. Deve-se optar por uma abordagem gradual (iterativa) da migração para manter a estabilidade do sistema geral (FAN; MA, 2017). O maior desafio para a migração é identificar e separar esses serviços candidatos. Pode levar muito tempo e esforço para refatorar[5] os serviços da arquitetura monolítica.

  É por isso que a refatoração para microsserviços deve ser feita em pequenas partes. Provavelmente é melhor começar pelos serviços mais fáceis e óbvios, e quando o time e a organização tiverem mais conhecimento sobre arquitetura de microsserviços, então os serviços podem se tornar mais refinados (KALSKE; MÄKITALO; MIKKONEN, 2018).
- **UML:** o diagrama de classes UML é uma ferramenta útil para mapear classes em microsserviços, porém é preciso ter cautela para não gerar um grande número de microsserviços. Uma abordagem mais lógica é agrupar classes que implementam funcionalidades coesas. Por exemplo, as classes relacionadas à funcionalidade contábil podem ser agrupadas em um único microsserviço. O objetivo é que cada componente no diagrama de

5 Refatoração (do inglês *refactoring*) é o processo de modificar um sistema de *software* para melhorar a estrutura interna do código sem alterar o seu comportamento externo.

componentes da aplicação possa ser mapeado para um microsserviço separado.

O desafio está em derivar um diagrama de componentes a partir de um diagrama de classes UML, o que não é uma tarefa simples. A componentização pode ser vista como um processo de reduzir o número de nós em um diagrama de classes. Em um aplicativo baseado em microsserviços, cada nó no diagrama pode representar um microsserviço, e as arestas representam a interface de passagem de mensagens entre os microsserviços. (MISHRA; KUNDE; NAMBIAR, 2018).

- **Infraestrutura**
  - **API Gateway:** componente fundamental da arquitetura de microsserviços, atuando como uma camada intermediária entre clientes e os diferentes serviços disponíveis. Ele recebe solicitações dos clientes e as redireciona para o microsserviço correto, atuando como um proxy para os diferentes serviços (HAUGELAND et al., 2021).

    Por exemplo, imagine uma loja virtual que oferece diferentes serviços aos seus clientes, como carrinho de compras, pagamento e rastreamento de pedidos. Cada um desses serviços pode estar localizado em servidores diferentes e com diferentes tecnologias. Com o uso do API Gateway, os clientes não precisam conhecer a localização ou detalhes técnicos de cada serviço, podendo realizar todas as suas operações de maneira transparente e unificada.

    Além disso, o API Gateway pode ser responsável por tarefas como autenticação, autorização e controle de acesso. Por exemplo, ao acessar a loja virtual, o cliente pode ser autenticado pelo gateway, que verifica suas credenciais e autoriza seu acesso aos diferentes serviços da aplicação (HAUGELAND et al., 2021).

    Essa abordagem melhora a segurança e a confiabilidade da arquitetura de microsserviços, pois todas as solicitações passam pelo gateway, que pode aplicar políticas de segurança consistentes em toda a aplicação (HAUGELAND et al., 2021).
  - **Circuit Breaker:** o *design* de microsserviços deve ser tolerante a falhas, especialmente quando a organização tem muitos microsserviços em funcionamento. Com tantos serviços distribuídos, é

possível que um serviço fique sobrecarregado e não possa responder em tempo hábil, ou até mesmo ficar inativo. Nesses casos, o padrão Circuit Breaker pode ser útil, monitorando as falhas e interrompendo as chamadas subsequentes para a dependência quando há falhas suficientes. Em vez de adicionar mais carga à dependência, o Circuit Breaker retorna imediatamente um erro ao usuário, dando à dependência tempo para se recuperar da carga (KALSKE; MÄKITALO; MIKKONEN, 2018).

- **Docker:** permite a criação, o gerenciamento e o isolamento de contêineres, que são ambientes virtuais leves e portáteis, que podem ser implantados em qualquer sistema operacional que suporte o Docker (DORRELL, 2017).

  Uma das principais vantagens do Docker é sua eficiência na criação de ambientes de teste para a validação do correto funcionamento de um aplicativo em um ambiente distribuído (ACEVEDO; GÓMEZ Y JORGE; PATIÑO, 2017). Ao utilizar o Docker, os desenvolvedores podem facilmente criar e gerenciar contêineres para cada um dos microsserviços que compõem o aplicativo, permitindo uma simulação realista do ambiente de produção.

  Além disso, o uso do Docker facilita a correção de possíveis erros de comunicação entre os microsserviços. Como os contêineres são isolados uns dos outros, é possível identificar e corrigir problemas de comunicação sem afetar o funcionamento dos demais serviços (HAJ-YOUSSEF et al., 2020).

  Outra vantagem do Docker é sua portabilidade. Uma vez que os contêineres são independentes do sistema operacional, eles podem ser facilmente implantados em qualquer infraestrutura compatível com o Docker, incluindo servidores locais ou na nuvem (KANE; MATTHIAS, 2018).

- **Service Discovery:** em uma arquitetura de microsserviços, é comum haver diversos serviços com múltiplas instâncias, o que torna a tarefa de monitorar e controlar o endereço e o número de porta de cada um deles bastante complicada. Para lidar com esse desafio, é necessário utilizar um componente de Service Discovery, que permite obter

informações sobre as instâncias disponíveis de cada serviço (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016). Com a descoberta de serviço, os microsserviços podem ser registrados e tornarem-se endereçáveis, o que possibilita a comunicação entre eles utilizando um IP e uma porta específica (FAN; MA, 2017).

4.5. DESENVOLVIMENTO DO INSTRUMENTO DE PESQUISA DO TIPO QUESTIONÁRIO

A RSL permitiu a construção de um questionário baseado nas orientações para migração identificadas nos artigos selecionados. Esse questionário é um instrumento decisivo para validar tais orientações. Adaptado de Lucato et al. (2012), o questionário utilizado contém sete orientações relevantes, com quatro afirmações sobre cada tópico. O questionário passará por duas rodadas de validação.

Na primeira rodada, chamada de piloto, a validação dos itens que comporão o questionário será obtida por meio do teste de face, a partir da percepção de cinco especialistas em TI sobre migração de arquitetura monolítica para arquitetura de microsserviços. O questionário contém as orientações classificadas como relevantes e a escala Likert para que os especialistas avaliem tais orientações no apoio às MEs de *software*.

Na segunda rodada, o questionário revisto, completo e validado na primeira rodada será publicado e disponibilizado para no máximo 21 especialistas em TI (método Delphi), os quais responderão cada afirmação (orientação classificada) por meio da escala Likert, junto às suas respectivas percepções. As afirmações (orientações) de cada requisito são mensuradas com uma escala de cinco pontos, indicando: “discordo totalmente”, “discordo”, “indiferente”, “concordo” e “concordo totalmente”. O questionário desenvolvido para esta pesquisa é semelhante ao apresentado por Creazza et al. (2021), que também utiliza uma escala Likert de cinco pontos, mas com outras nomenclaturas adaptadas pelo pesquisador.

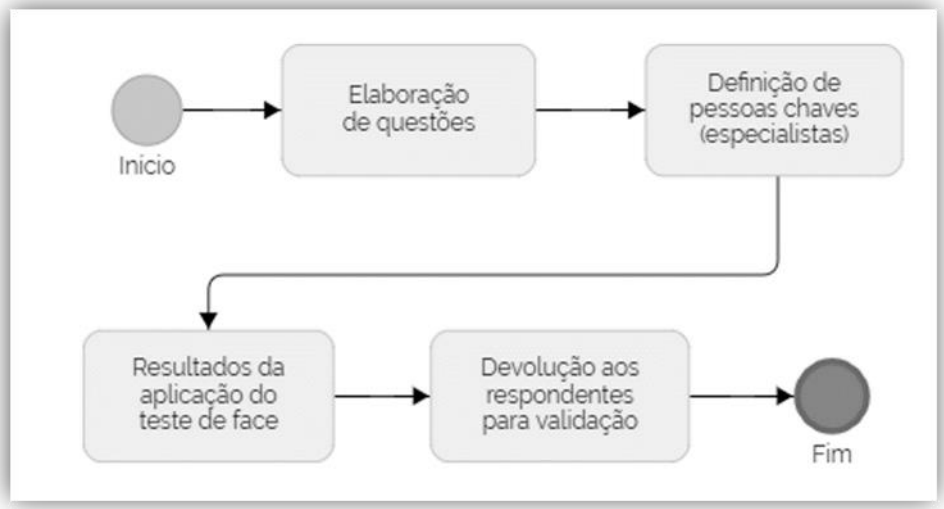
4.6. TESTE PILOTO

O teste piloto é uma etapa fundamental para validar o instrumento de pesquisa composto das questões criadas a partir da pesquisa bibliográfica realizada na RSL. Essa validação foi realizada por meio do teste de face, que permite verificar a qualidade dos dados apurados na pesquisa bibliográfica e fazer ajustes antes da aplicação do instrumento em campo (FORZA, 2002).

Conforme destacado por Bowling (1997), a validação do teste piloto é uma forma de validação de conteúdo que visa verificar se os itens selecionados para o estudo estão medindo efetivamente o que se deseja medir, além de avaliar se o significado e a relevância do indicador estão evidentes por si só. A aplicação de um teste de face busca, então, atestar a validade aparente do instrumento, por meio da validação por especialistas no assunto, verificando se as questões são apropriadas para aplicação na prática organizacional (PASQUALI, 2007).

As atividades para a realização do teste de face estão ilustradas na Figura 13.

Figura 13 – Fluxo do processo para teste de face



Fonte: autor.

Na realização do teste de face foram apresentados: o convite aos cinco especialistas (Apêndice A), as questões sobre a qualidade do questionário (Apêndice B), os conceitos contidos nas orientações relevantes (Apêndice C) e o questionário (Apêndice D) com as orientações classificadas como relevantes, levantadas na literatura. O teste de face tem o objetivo de verificar a aderência

desse instrumento de pesquisa que será utilizado na construção de um Survey com método Delphi.

4.6.1. Revisão dos comentários dos respondentes

O teste piloto foi realizado entre os dias 07 e 20 de dezembro de 2022 por meio de um Formulário Google. O formulário foi enviado por e-mail, incluindo uma carta-convite explicando os objetivos de pesquisa para os seguintes profissionais selecionados:

- Primeiro especialista: profissional com 17 anos de experiência na área de desenvolvimento de *software*; nos últimos 6 anos, ocupa o cargo de arquiteto de *software*.
- Segundo especialista: profissional com 15 anos de experiência na área de desenvolvimento de *software*, trabalhou em várias empresas nacionais e atualmente ocupa o cargo de arquiteto de *software*.
- Terceiro especialista: profissional com 25 anos de experiência em empresas multinacionais e atualmente exerce o cargo de engenheiro de *software*.
- Quarto especialista: profissional com 12 anos de experiência na área de desenvolvimento de *software* e atualmente exerce o cargo de engenheiro de *software*.
- Quinto especialista: profissional com 17 anos de experiência na área de desenvolvimento de *software* e atualmente exerce o cargo de engenheiro de *software*.

A principal característica do questionário para o teste piloto foi a inserção de um campo obrigatório para comentários e para validação de cada questão. Isso permitiu a percepção dos entrevistados sobre as orientações, e não as respostas específicas para cada peso da escala Likert. O objetivo foi ter uma percepção sobre os textos e as opções da escala Likert. Isso deu visibilidade à validade do questionário e sua aderência à realidade das empresas.

No Quadro 11 são apresentadas uma revisão dos comentários dos entrevistados e a validação de cada sugestão.

Quadro 11 – Considerações/comentários dos respondentes do teste de face (piloto)

Respondente	Questão avaliada	Considerações do respondente
2	Q1 – O que motivou a empresa a migrar da arquitetura monolítica para arquitetura de Microsserviços em nuvem?	Questão muito relevante, pois muitas migrações são realizadas para satisfação do profissional e não por necessidade do negócio, trazendo complexidades desnecessárias ao projeto.
2	Q5 – A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura de nuvem e também dos Microsserviços (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016)	Usufruir da estrutura de nuvem não está diretamente relacionado, pois é possível validar MVP's gastando pouco na nuvem, o que já representa uma forma de ganho usufruindo da estrutura de nuvem.
2	Q10 – É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, suportar a migração (SILVA; CARNEIRO; MONTEIRO, 2019)	Para o respondente, a forma como a questão foi elaborada não deixa claro seu objetivo.
2 e 3	Q24 – Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021)	A questão 24 e 25 estão duplicadas.
5	Q13 – Uma vantagem dessa arquitetura é que cada Microsserviço pode possuir seu banco de dados que pode inclusive ser NoSQL (FURDA et al., 2018)	Para o respondente é uma vantagem, porém um trabalho que deve ser muito bem pensado.
5	Q16 – Uma grande desvantagem é a separação de dados, pois, além de possíveis inconsistências, as soluções podem trazer sobrecarga de comunicação adicionada da rede ou chamadas externas também podem introduzir latência (GRAVANIS; KAKARONTZAS; GEROGIANNIS, 2021)	A questão é importante para fazer um contraponto às anteriores.

5	Migrar um sistema de *software* de uma arquitetura monolítica para uma arquitetura de Microsserviços não é uma tarefa trivial. Uma abordagem gradual (iterativa) da migração é a mais aconselhável (WOLFART et al., 2021)	A resposta para essa questão é muito óbvia e não deveria constar no questionário.

Fonte: autor.

4.6.2. Análise das respostas dos especialistas

A partir da revisão dos comentários dos respondentes do teste piloto, é possível afirmar que o questionário apresentou resultado satisfatório e superou as expectativas. Embora a revisão sistemática tenha fornecido bases sólidas para a identificação das práticas, havia uma preocupação com a semelhança de algumas orientações e como isso poderia causar contradições para os respondentes.

O fato positivo é que as contradições não se concretizaram, e, segundo os comentários, os respondentes diferenciaram as orientações. Isso revela que já existe um consenso quanto aos termos com os profissionais consultados, mas isso poderá ser confirmado na pesquisa mais abrangente a ser realizada.

Por um lado, os profissionais não encontraram problemas com as questões colocadas no documento e, portanto, o questionário está aprovado para uma busca mais abrangente no mercado corporativo. Entende-se que o questionário é viável para ser submetido a um público mais amplo, o que permitirá uma análise quantitativa com estatísticas básicas nas próximas etapas.

Todavia, os itens a seguir devem ser aprimorados para uma submissão mais ampla.

- **Item 1:** sobre a questão 5 — “A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura de nuvem e também dos Microsserviços (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016).” —, o respondente afirma que usufruir da estrutura de nuvem não está diretamente relacionado, pois é possível validar MVP’s gastando pouco na nuvem, o que já representa uma forma de ganho usufruindo da estrutura de nuvem.

Após avaliação, a questão foi atualizada para: “A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura dos Microsserviços em nuvem. (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016).”.

- **Item 2:** sobre a questão 10 — “É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, suportar a migração (SILVA; CARNEIRO; MONTEIRO, 2019).” —, o respondente comenta que a forma como a questão foi elaborada não deixa claro o seu objetivo.

Após avaliação, a questão foi atualizada para: “É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, identificar os serviços candidatos e suportar a migração (SILVA; CARNEIRO; MONTEIRO, 2019).”.

- **Item 3:** sobre a questão 13 — “Uma vantagem dessa arquitetura é que cada Microsserviço pode possuir seu banco de dados que pode inclusive ser NoSQL (FURDA et al., 2018).” —, o respondente afirma que é uma vantagem, porém um trabalho que deve ser muito bem pensado.

Este item não gerou alteração na questão, pois o respondente apenas destacou a importância da questão.

- **Item 4:** sobre a questão 16 — “Uma grande desvantagem é a separação de dados, pois, além de possíveis inconsistências, as soluções podem trazer sobrecarga de comunicação adicionada da rede ou chamadas externas também podem introduzir latência (GRAVANIS; KAKARONTZAS e GEROGIANNIS, 2021).” —, o respondente comenta que a questão é importante para fazer um contraponto às anteriores.

Este item não gerou alteração na questão, pois o respondente apenas destacou a importância da questão.

- **Item 5:** sobre a questão 23 — “Migrar um sistema de *software* de uma arquitetura monolítica para uma arquitetura de Microsserviços não é uma tarefa trivial. Uma abordagem gradual (iterativa) da migração é a mais aconselhável (WOLFART et al., 2021).” —, o respondente afirma que a resposta é muito óbvia.

Este item não foi atendido. Decidiu-se manter a questão como estava, pois a argumentação do respondente não foi suficiente.

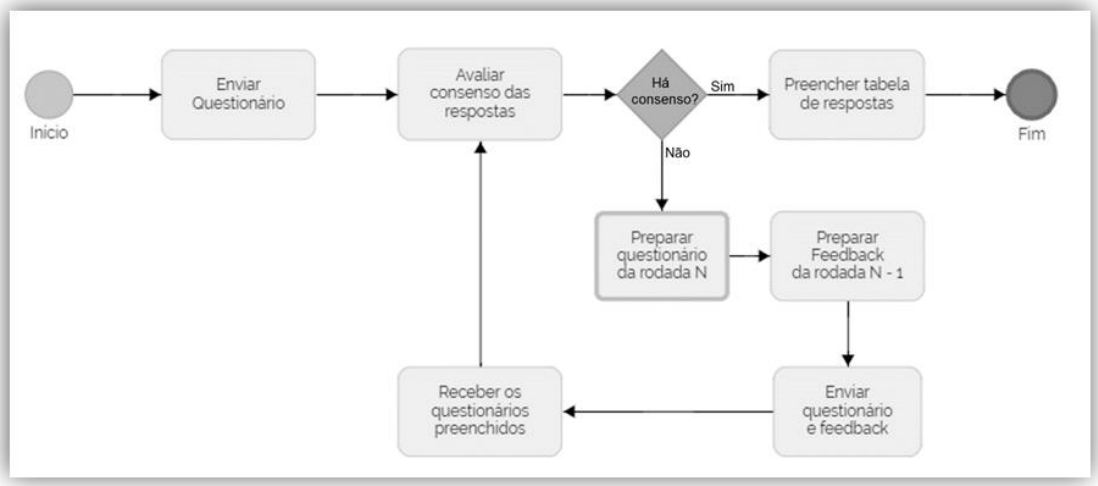
- **Item 6:** sobre as questões 24 e 25 — “Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021).” —, o respondente comenta que elas estão duplicadas.

  Este item foi atendido**,** e a questão em duplicidade foi removida.

4.7. MÉTODO DELPHI (PESQUISA DO TIPO SURVEY)

Após a validação do instrumento de pesquisa (teste de face, teste piloto), o pesquisador iniciou o desenvolvimento do processo de envio de perguntas para a aplicação do Survey controlado, utilizando o método Delphi e a avaliação das respostas e realimentação controlada, conforme o fluxo apresentado na Figura 14.

Figura 14 – Fluxo do método Delphi



Fonte: autor.

O uso do método Delphi visa obter um consenso, por meio de consultas a especialistas de TI de modo intuitivo e iterativo, e, ao final de algumas rodadas de pesquisa com realimentações controladas, obter-se-á um ponto de vista da maioria dos participantes da pesquisa (DALKEY; HELMER, 1963).

Conforme o fluxo demonstrado na Figura 14, o questionário foi enviado aos entrevistados para a primeira rodada. Os entrevistados responderam apenas

perguntas de múltipla escolha. Na segunda rodada, as perguntas foram acompanhadas de uma realimentação da rodada anterior, e as respostas, de um comentário de cada respondente. A descrição da aplicação do método Delphi está detalhada na seção 5 desta dissertação.

O resultado final da aplicação do método Delphi foi utilizado para a montagem do guia de orientações relevantes para uso das MEs de *software* brasileiras em sua caminhada quanto a migração de sistemas monolíticos para sistemas em microsserviços em nuvem.

5. EXECUÇÃO DO MÉTODO DELPHI

A escolha do método Delphi foi baseada em sua comprovada eficácia e reconhecimento na comunidade acadêmica. Além disso, mesmo diante da escassez de especialistas com conhecimento específico nas questões de pesquisa, os requisitos de tamanho do painel Delphi são acessíveis, permitindo a formação de até quatro painéis, cada um composto de um mínimo de dez membros (PALIWODA, 1983).

De acordo com Dalkey e Helmer (1963), um aspecto relevante é a flexibilidade do método Delphi em relação ao seu *design*, o que possibilita a realização de entrevistas de acompanhamento. Essa flexibilidade contribui para a coleta de dados mais abrangentes e proporciona uma compreensão mais profunda das questões de pesquisa fundamentais.

No contexto deste estudo, seguindo o fluxo apresentado na Figura 14 (método Delphi), o questionário, que já incorporou as alterações sugeridas durante o piloto (teste de face), foi publicado e respondido por 21 especialistas por meio do Google Form, no período de 12 a 30 de janeiro de 2023.

O perfil completo desses profissionais entrevistados encontra-se no Apêndice G. Para a execução das rodadas na busca de consenso foram enviados 102 convites por meio de mensagens privadas no LinkedIn, incluindo uma carta-convite (disponível no Apêndice A) na qual são descritos os objetivos da pesquisa.

5.1. DECLARAÇÕES DA LITERATURA

As afirmativas do questionário publicado e enviado aos especialistas sofreram alterações para atender aos resultados do teste de face. Os itens afetados e alterados foram apontados a seguir.

As pontuações da escala Likert foram acomodadas da seguinte forma:

1. Discordo totalmente
2. Discordo
3. Indiferente
4. Concordo
5. Concordo totalmente

O questionário foi desenvolvido com as seguintes afirmações para cada requisito identificado:

Perfil do respondente

• Função na organização.

• Quantos anos de experiência você possui em sua função?

• Quantos anos de experiência como arquiteto de software ou desenvolvedor de software você possui?

• Quantos anos de experiência atuando com o conceito de serviços você possui?

• Quantos anos de experiência atuando com Microsserviços você possui?

• Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação”.?

Motivações para a migração

• Por que sua empresa decidiu migrar?

Informações/métricas de migração

• Antes da migração, quais as informações/métricas foram consideradas?

• Quais informações/métricas foram consideradas após a migração?

• Os objetivos que motivaram a migração foram alcançados em sua totalidade?

O questionário foi acomodado com as seguintes afirmações para cada orientação identificada:

1. **DevOps (*Continuous Integration/Continuous Delivery*)**
    a. A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura dos Microsserviços em nuvem (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016).
    b. Para a equipe ter produtividade atuando com Microsserviços é necessário ter entrega e implantação contínua dos artefatos de *software* até sua liberação para produção (ACEVEDO; GÓMEZ Y JORGE; PATIÑO, 2017).

c. Sem a cultura DevOps, fica muito difícil lidar com os múltiplos serviços, suas implantações e validar as ações do serviço (KALSKE; MÄKITALO; MIKKONEN, 2018).
d. Para atuar com Microsserviços, a automatização do gerenciamento de configuração deve ser implantada (WOLFART et al., 2021).

2. *Domain Driven Design* (DDD) para identificar os microsserviços candidatos

a. O DDD é um método que pode ser utilizado para facilitar a extração de serviços com baixo acoplamento e propõe a definição de contextos compostos por componentes do modelo de negócio que sejam consistentes entre si (FAN; MA, 2017).
b. É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, identificar os serviços candidatos e suportar a migração (SILVA; CARNEIRO; MONTEIRO, 2019).
c. É recomendado utilizar o DDD para encontrar candidatos a Microsserviços no sistema original. Os resultados da análise de contexto limitado são uma ferramenta chave para identificar candidatos a Microsserviços (KAZANAVIČIUS; MAŽEIKA, 2020).
d. Um dos métodos mais eficazes para identificar os serviços candidatos é identificando os domínios e subdomínios com contexto e limites claros (WOLFART et al., 2021).

3. Segregação de Banco de Dados

a. Uma vantagem dessa arquitetura é que cada microsserviço pode possuir seu banco de dados que pode inclusive ser NoSQL (FURDA et al., 2018).
b. Dividir os dados em bancos de dados separados pode causar inconsistência de dados especialmente quando estiver em um contexto de transação (KAZANAVIČIUS; MAŽEIKA, 2020).
c. Os procedimentos para resolver essas inconsistências nos repositórios pode afetar o desempenho do aplicativo (MISHRA; KUNDE; NAMBIAR, 2018).
d. Uma grande desvantagem é a separação de dados, pois, além de possíveis inconsistências, as soluções podem trazer sobrecarga de

comunicação adicionada da rede ou chamadas externas também podem introduzir latência (GRAVANIS; KAKARONTZAS; GEROGIANNIS, 2021).

4. **Perfil da Equipe para realizar a migração**
    a. Quando a organização adota o estilo de arquitetura de Microsserviços, as equipes devem ter mais liberdade e responsabilidade, mas menos processos. Isso significa que as equipes podem implantar seu serviço na produção quando precisarem, em vez de esperar pela aprovação de outra pessoa (KALSKE; MÄKITALO; MIKKONEN, 2018).
    b. A equipe deve possuir uma ótima qualificação para fazer a extração correta de Microsserviços do sistema monolítico. É uma tarefa muito difícil e crucial para uma migração bem-sucedida (KAZANAVIČIUS; MAŽEIKA, 2020).
    c. Com a adoção de Microsserviços é necessário destacar um indivíduo para monitorar os Microsserviços/infraestrutura para garantir a disponibilidade do serviço e monitorar gargalos, desempenho e uso da infraestrutura e recursos (WOLFART et al., 2021).

5. **Por Onde Começar a migração**
    a. É melhor começar pelos serviços mais fáceis e óbvios e quando a organização tiver mais conhecimento sobre arquitetura de Microsserviços então os serviços podem se tornar mais refinados (KALSKE; MÄKITALO; MIKKONEN, 2018).
    b. Iniciar com a funcionalidade que tem o menor impacto quando comparada às demais. Esse processo facilita a validação dos limites estabelecidos entre os recursos com o menor risco de efeitos colaterais para validar o mapeamento de contexto (SILVA; CARNEIRO; MONTEIRO, 2019).
    c. Migrar um sistema de software de uma arquitetura monolítica para uma arquitetura de Microsserviços não é uma tarefa trivial. Uma abordagem gradual (iterativa) da migração é a mais aconselhável (WOLFART et al., 2021).
    d. Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021).

6. **UML para identificar os microsserviços candidatos**
    a. Contextos limitados podem ser separados por meio da decomposição por verbos (casos de uso) ou por substantivos (recursos) e dessa forma é possível identificar os Microsserviços candidatos (FRITZSCH et al., 2019).
    b. O primeiro passo é a transformação do monolito na representação gráfica. No grafo, cada vértice representa a classe do monolito e as arestas não direcionadas representam seu acoplamento com outras classes do monolito. Ao cortar o gráfico em componentes é possível observar os candidatos a Microsserviços (SILVA; CARNEIRO; MONTEIRO, 2019).
    c. Através de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML. Em seguida listar todas as funcionalidades para as quais o sistema foi projetado e para identificar os Microsserviços candidatos, deve fazer a identificação dos artefatos de implementação de cada recurso (WOLFART et al., 2021).

7. **Infraestrutura**
    a. O Circuit Breaker monitora as falhas e, quando houver falhas suficientes, as chamadas subsequentes para a dependência não serão feitas e, em vez disso, um erro será retornado, em vez de adicionar mais carga à dependência. Quando a organização tem mais do que alguns Microsserviços, também deve levar em consideração a possibilidade de um serviço não responder (KALSKE; MÄKITALO; MIKKONEN, 2018).
    b. O API gateway é parte integrante da arquitetura de Microsserviços. O gateway serve como uma camada de conexão para clientes, redirecionando as solicitações para o microsserviço correto, servindo como proxy para os diferentes serviços (HAUGELAND et al., 2021).
    c. O Service Discovery deve ser implantado quando se utilizar arquitetura de Microsserviços, pois, para manter o controle dos serviços implantados e seu endereço exato e número de porta é uma tarefa complicada e com essa solução é possível obter as instâncias disponíveis de cada serviço (MISHRA; KUNDE; NAMBIAR, 2018).
    d. O Docker deve ser implantado, pois torna o ambiente facilmente portátil e isolado. Não há conflitos de dependências ou a necessidade de configurar cada ambiente (WOLFART et al., 2021).

5.2. VALIDAÇÃO DO GUIA DE ORIENTAÇÕES

Esta subseção apresenta uma revisão das respostas para cada afirmação no questionário e valida as orientações com base nos resultados da pontuação atribuída pelos respondentes conforme a escala Likert. Acrescentam-se comentários, que são textos livres produzidos pelos respondentes e que são consolidados, comentados e trazem perspectivas relevantes sobre as afirmações da literatura. Eles são de grande valor, pois trazem as percepções de especialistas sobre o assunto que atuam fortemente no mercado.

Vale destacar que o guia de orientações foi construído a partir de uma RSL e que a validação por especialistas garante que esteja aderente à realidade das empresas. Essa validação proporciona uma contribuição mútua entre a academia e o mercado. Por um lado, a academia contribui com os resultados apresentados pelos autores em uma RSL, que identifica e classifica as orientações mais relevantes sobre microsserviços em nuvem. Por outro lado, os especialistas de mercado contribuem para a percepção de aderência dessas orientações à realidade das empresas, ou seja, validam um guia de orientações como contribuição para as empresas na tomada de decisão sobre a utilização de microsserviços.

O questionário com a aplicação da escala Likert forneceu dados que permitem identificar quantos respondentes escolheram um valor na escala específica para cada afirmação, por exemplo, quantos escolheram concordar ou discordar das afirmações. A partir desses dados, é possível avaliar de forma quantitativa a aderência desse guia à realidade do mercado.

Como critério de pesquisa, assumiu-se que “concordo” e “concordo totalmente” definem essa aderência, portanto foi adotada a média simples de cada resposta. Dessa forma, toma-se a soma dos percentuais de cada escala e calcula-se a média simples tanto para “concordo” quanto para “concordo totalmente” (CAZORLA; SANTANA; UTSUMI, 2019).

Ainda segundo os autores, quando há uma porção de dados brutos, apresentados em tabelas ou gráficos, os valores são somados e depois divididos pelo número de dados. Isso determina uma média simples. Dessa forma, são obtidas duas médias simples, e assume-se que a média final considerada para a validação é a soma das médias dessas duas escalas (“concordo” e “concordo totalmente”). Como critério deste trabalho, foi assumido pelo pesquisador que se a

média final for acima de 72%, a orientação foi aprovada por especialistas. Isso significa que as orientações têm aderência à realidade das empresas. Essas orientações foram encontradas durante a RSL e consolidadas por meio da Curva ABC, que pode ser observada no Quadro 10.

A referência à Curva ABC é importante para a validação das orientações classificadas como relevantes; essas orientações compõem 72% do universo de citações em cada um dos oito tópicos identificados na RSL, para assegurar a uniformidade na análise.

5.2.1. Perfil das empresas que podem utilizar microsserviços em nuvem

O tema “Microsserviços” ganhou popularidade nos últimos anos em razão das facilidades que a Computação em Nuvem trouxe para as empresas. Os respondentes foram questionados se qualquer empresa que desenvolve *softwares* (independentemente de seu porte) pode implantar microsserviços em nuvem, e 76% dos respondentes afirmaram que sim. A Tabela 1 mostra como os respondentes preencheram os questionários.

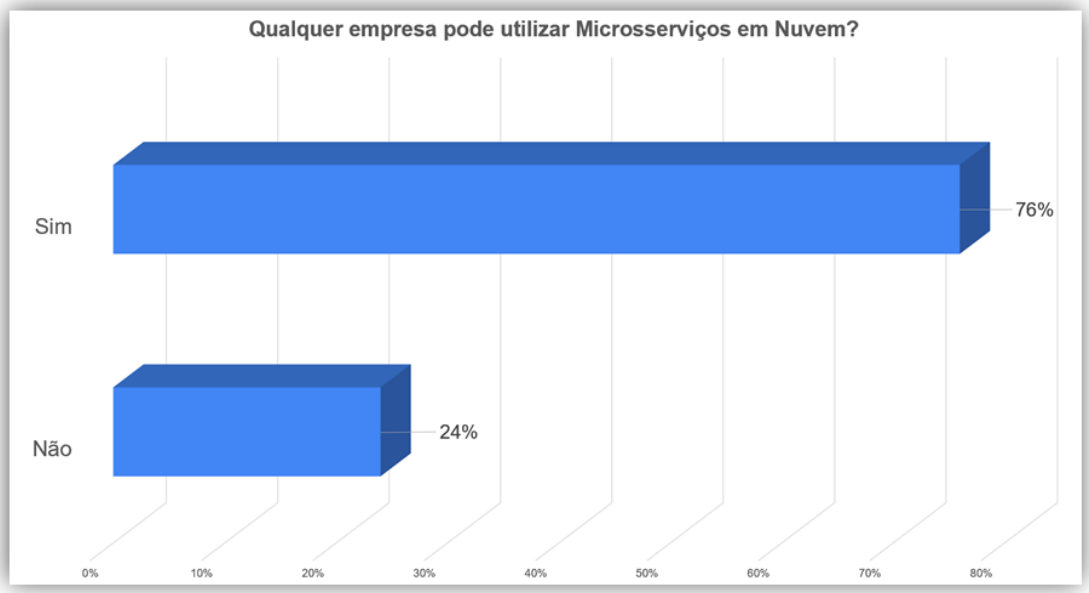
Tabela 1 – Respostas sobre o perfil das empresas que podem utilizar microsserviços em nuvem

Respondente	Concordam?	
	SIM	NÃO
1	X	
2	X	
3		X
4	X	
5	X	
6	X	
7	X	
8	X	
9		X
10	X	
11		X
12	X	
13	X	
14	X	
15		X
16	X	
17	X	
18	X	
19	X	
20	X	
21	X	
Total	17 (76,4%)	4 (23,63%)

Fonte: autor.

Segundo os especialistas consultados, qualquer empresa que desenvolve *softwares* (independentemente de seu porte) pode utilizar microsserviços em nuvem. Para efeito dos gráficos a seguir, o autor arredondou o valor calculado de 76,47% para 76% e o valor calculado de 23,53% para 24%. O gráfico da Figura 15 apresenta as respostas de acordo com os especialistas.

Figura 15 – Qualquer empresa pode utilizar microsserviços em nuvem independentemente de seu porte?



Fonte: autor.

Conforme o gráfico da Figura 15, 76% dos especialistas afirmam que qualquer empresa pode utilizar microsserviços em nuvem em sua operação. Dessa forma, esse guia se torna mais relevante para as MEs de *software*, pois há uma série de desafios que devem ser considerados ao adotar essa arquitetura.

5.2.2. Motivação para migrar da arquitetura monolítica para microsserviços em nuvem

Os respondentes destacam as motivações para as empresas migrarem da arquitetura monolítica para microsserviços em nuvem. A Tabela 2 apresenta como os respondentes preencheram os questionários.

Tabela 2 – Respostas sobre a motivação para as empresas migrarem seus *softwares* para arquitetura de microsserviços em nuvem

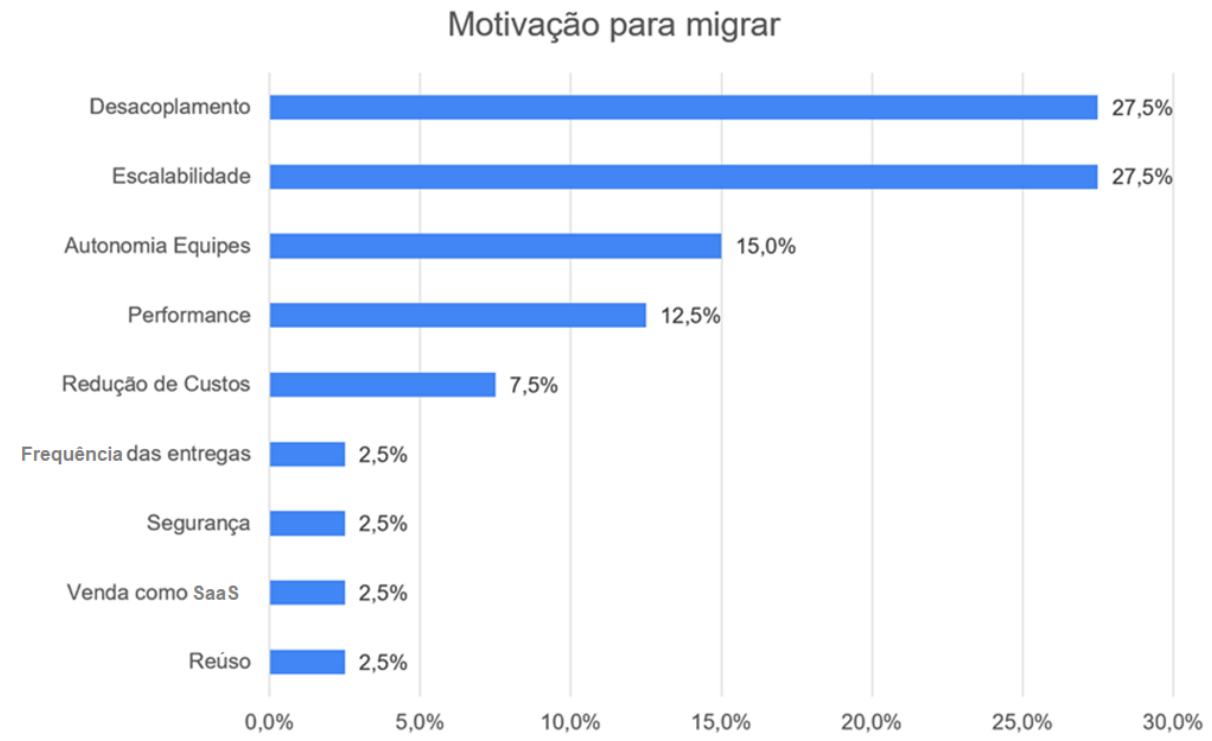
Respondente	**Não participou dessa etapa do projeto**	**Desacoplamento**	**Escalabilidade**	**Autonomia das equipes**	***Performance***	**Redução custos**	**Aumentar a frequência das entregas**	**Reúso**	**Venda como SaaS**	**Segurança**
1	X									
2	X									
3		X	X		X					
4	X									
5		X		X				X		
6		X	X	X		X				
7		X	X	X						
8		X		X						
9							X		X	
10			X	X	X					
11		X								
12			X							X
13		X			X					
14	X									
15			X			X				
16		X	X							
17			X							
18		X			X					
19		X	X			X				
20		X	X	X						
21			X		X					
Total		**11 (27,5%)**	**11 (27,5%)**	**6 (15%)**	**5 (12,5%)**	**3 (7,5%)**	**1 (2,5%)**	**1 (2,5%)**	**1 (2,5%)**	**1 (2,5%)**

SaaS: *Software as a Service.*

Fonte: autor.

As duas maiores motivações das empresas para realizarem a migração de seus *softwares* para microsserviços em nuvem é reduzir o acoplamento (27,5%) e melhorar a escalabilidade (27,5%). O gráfico da Figura 16 apresenta as respostas de acordo com os especialistas.

Figura 16 – Motivação para migrar da arquitetura monolítica para microsserviços em nuvem.



SaaS: *Software as a Service.*

Fonte: autor.

Conforme o gráfico da Figura 16, as maiores motivações são reduzir o acoplamento e a escalabilidade, com 27,5% cada. Aumento da autonomia das equipes foi indicado por 15%; melhoria de *performance*, por 12,5%; redução de custos, por 7,5%; seguidos por melhorar a frequência das entregas, melhorar a segurança, vender módulos do sistema como SaaS e aumentar o reúso, todos com 2,5%.

5.2.3. Métricas utilizadas para realizar a migração

Os respondentes citam as métricas utilizadas para tomar a decisão para realizar a migração da arquitetura monolítica para microsserviços em nuvem. A Tabela 3 apresenta como os respondentes preencheram os questionários.

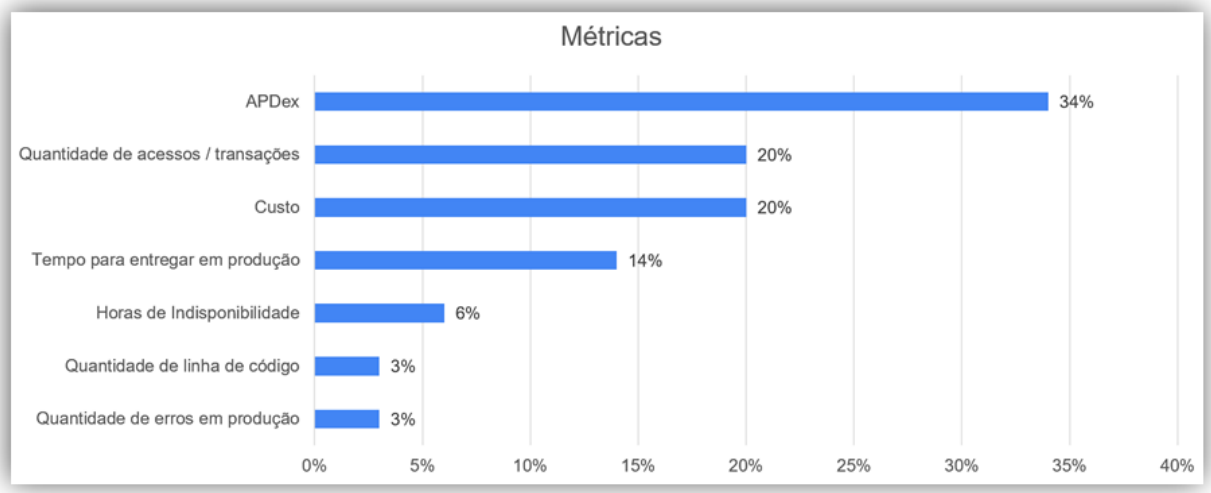
Tabela 3 – Respostas sobre as métricas utilizadas antes e após a migração

Respondente	Não participou dessa etapa do projeto	Apdex	Custo	Tempo para entregar em produção	Quantidade de acessos/ transações	Quantidade de linha de código	Horas de indisponibilidade	Quantidade de erros em produção
1	X							
2	X							
3		X			X			
4	X							
5		X		X				
6		X	X		X			
7			X	X				
8			X		X			
9				X			X	X
10				X	X			
11				X	X	X		
12		X	X					
13		X					X	
14	X							
15		X						
16		X						
17		X	X		X			
18		X	X		X			
19		X						
20		X						
21		X	X					
Total		**12 (34%)**	**7 (20%)**	**5 (14%)**	**7 (20%)**	**1 (3%)**	**2 (6%)**	**1 (3%)**

Fonte: autor.

A métrica mais utilizada foi o Apdex (34%), seguido por Custo e Quantidade de acessos/transações, ambos com 20%. O gráfico da Figura 17 apresenta as respostas de acordo com os especialistas.

Figura 17 – Métricas utilizadas antes e após a migração



Fonte: autor.

Como apresenta o gráfico da Figura 17, a métrica mais utilizada é o Apdex, que é um padrão para medir a satisfação dos usuários, com 34%, seguido por Custo e Quantidade de acessos/transações, ambos com 20%; Tempo para entrega em produção foi indicado por 14%, seguido por Horas de indisponibilidade, com 6%, Quantidade de linha de código e Quantidade de erros em produção, ambos com 3%.

5.2.4. Cumprimento dos objetivos iniciais

Os respondentes citam se os objetivos iniciais foram atendidos. A Tabela 4 apresenta como os respondentes preencheram os questionários.

Tabela 4 – Respostas sobre o cumprimento dos objetivos iniciais

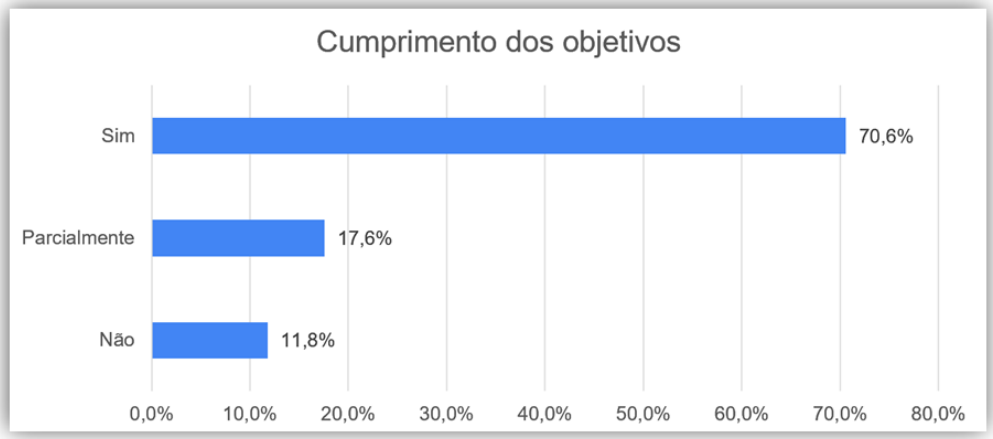
Respondente	Não participou dessa etapa do projeto	Atendidos integralmente	Não atendidos	Atendidos parcialmente
1	X			
2	X			
3		X		
4	X			
5			X	

6		X		
7				X
8		X		
9		X		
10		X		
11				X
12		X		
13		X		
14	X			
15			X	
16		X		
17		X		
18		X		
19		X		
20				X
21		X		
Total		**12 (70,6%)**	**2 (11,8%)**	**3 (17,6%)**

Fonte: autor.

A maioria dos especialistas (71%) afirmou que os objetivos foram cumpridos plenamente. O gráfico da Figura 18 apresenta as respostas de acordo com os especialistas.

Figura 18 – Os objetivos foram alcançados integralmente?



Fonte: autor.

Conforme o gráfico da Figura 18, 70,6% dos especialistas afirmam que os objetivos foram alcançados em sua totalidade; 17,6%, que os objetivos foram cumpridos parcialmente; e apenas 11,8%, que os objetivos não foram alcançados.

5.2.5. DevOps (*Continuous Integration/Continuous Delivery*)

Este tópico foi o mais citado durante a etapa de aplicação da Curva ABC, e as seguintes afirmações foram submetidas aos especialistas:

- A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura dos Microsserviços em nuvem (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016).
- Para a equipe ter produtividade atuando com Microsserviços é necessário ter entrega e implantação contínua dos artefatos de *software* até sua liberação para produção (ACEVEDO; GÓMEZ Y JORGE; PATIÑO, 2017).
- Sem a cultura DevOps, fica muito difícil lidar com os múltiplos serviços, suas implantações e validar as ações do serviço (KALSKE; MÄKITALO; MIKKONEN, 2018).
- Para atuar com Microsserviços, a automatização do gerenciamento de configuração deve ser implantada (WOLFART et al., 2021).

O gráfico da Figura 19 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 19 – Escala Likert DevOps (CI/CD)

Fonte: autor.

Como apresenta o gráfico da Figura 19, na afirmativa 1, 86% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 48% e 38%, respectivamente; 14% responderam “Discordo”; e não houve respostas para “Indiferente” e “Discordo totalmente”. A afirmativa 2 apresentou 95% das respostas entre “Concordo” e “Concordo totalmente”; 5% dos especialistas responderam “Discordo”; e não houve respostas para “Indiferente” e “Discordo totalmente”. Na afirmativa 3, 43%, 52% e 5% dos especialistas responderam “Concordo”, “Concordo totalmente” e “Indiferente”, respectivamente. Na afirmativa 4, 86% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 62% e 29%, respectivamente; 5% responderam “Discordo”; 5%, “Indiferente”; e não houve respostas para “Discordo totalmente”.

A escala “concordo” tem média de 55%, e “concordo totalmente”, 36%. A soma dessas duas médias resulta em 90%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 90% de concordância com as orientações. Assim, as orientações foram validadas pelos especialistas.

5.2.6. *Domain Driven Design* para identificar os microsserviços candidatos

Este tópico ficou em segundo lugar entre os mais citados durante a etapa de aplicação da Curva ABC, e as seguintes afirmações foram submetidas aos especialistas:

- Um dos métodos mais eficazes para identificar os serviços candidatos é identificando os domínios e subdomínios com contexto e limites claros (WOLFART et al., 2021).
- É recomendado utilizar o DDD para encontrar candidatos a Microsserviços no sistema original. Os resultados da análise de contexto limitado são uma ferramenta chave para identificar candidatos a Microsserviços (KAZANAVIČIUS; MAŽEIKA, 2020).
- É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, identificar os serviços candidatos e suportar a migração (SILVA; CARNEIRO; MONTEIRO, 2019).
- O DDD é um método que pode ser utilizado para facilitar a extração de serviços com baixo acoplamento e propõe a definição de contextos

compostos por componentes do modelo de negócio que sejam consistentes entre si (FAN; MA, 2017).

O gráfico da Figura 20 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 20 – Escala Likert *Domain Driven Design*

Fonte: autor.

Como apresenta o gráfico da Figura 20, na afirmativa 1, 86% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 67% e 19%, respectivamente; 14% responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”. A afirmativa 2 apresentou 91% das respostas entre “Concordo” (62%) e “Concordo totalmente” (29%); 5% dos especialistas responderam “Indiferente”; 5%, “Discordo”; e não houve respostas para “Discordo totalmente”. Na afirmativa 3, 62%, 29% e 10% dos especialistas responderam “Concordo”, “Concordo totalmente” e “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”. Na afirmativa 4, 91% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 43% e 48%,

respectivamente; 10% responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”.

A escala “Concordo” tem média de 58%, e “Concordo totalmente”, 31%. A soma dessas duas médias resulta em 89%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 89% de concordância com as orientações. Dessa forma, as orientações foram validadas junto aos especialistas.

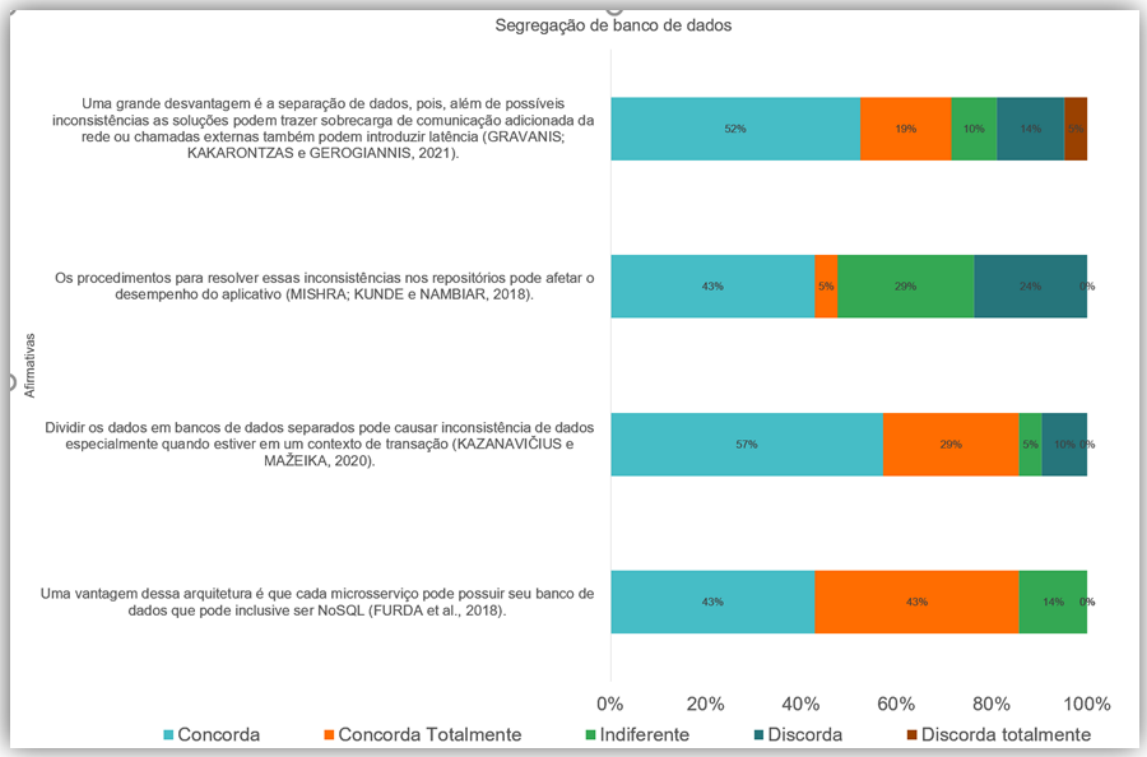
5.2.7. Segregação de Banco de Dados

Este tópico ficou em terceiro lugar entre os mais citados durante a etapa de aplicação da Curva ABC. As seguintes afirmações foram submetidas aos especialistas:

- Uma vantagem dessa arquitetura é que cada microsserviço pode possuir seu banco de dados que pode inclusive ser NoSQL (FURDA et al., 2018).
- Dividir os dados em bancos de dados separados pode causar inconsistência de dados especialmente quando estiver em um contexto de transação (KAZANAVIČIUS; MAŽEIKA, 2020).
- Os procedimentos para resolver essas inconsistências nos repositórios pode afetar o desempenho do aplicativo (MISHRA; KUNDE; NAMBIAR, 2018).
- Uma grande desvantagem é a separação de dados, pois, além de possíveis inconsistências, as soluções podem trazer sobrecarga de comunicação adicionada da rede ou chamadas externas também podem introduzir latência (GRAVANIS; KAKARONTZAS; GEROGIANNIS, 2021).

O gráfico da Figura 21 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 21 – Escala Likert Segregação de Banco de Dados



Fonte: autor.

Como apresenta o gráfico da Figura 21, na afirmativa 1, 71% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 52% e 19%, respectivamente; 10% responderam “Indiferente”; 14%, “Discordo”; e 5%, “Discordo totalmente”. A afirmativa 2 apresentou 43%, 5%, 29% e 24% das respostas para “Concordo”, “Concordo totalmente”, “Indiferente” e “Discordo”, respectivamente; e não houve respostas para “Discordo totalmente”. Na afirmativa 3, 86% das respostas foram entre “Concordo” (57%) e “Concordo totalmente” (29%); 5% dos especialistas responderam “Indiferente”; 10%, “Discordo”; e não houve respostas para “Discordo totalmente”. Na afirmativa 4, 86% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 43% e 43%, respectivamente; 14% responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”.

A escala “Concordo” tem média de 49%, e “Concordo totalmente”, 24%. A soma dessas duas médias resulta em 73%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 73% de concordância com as orientações. Dessa forma, as orientações foram validadas junto aos especialistas.

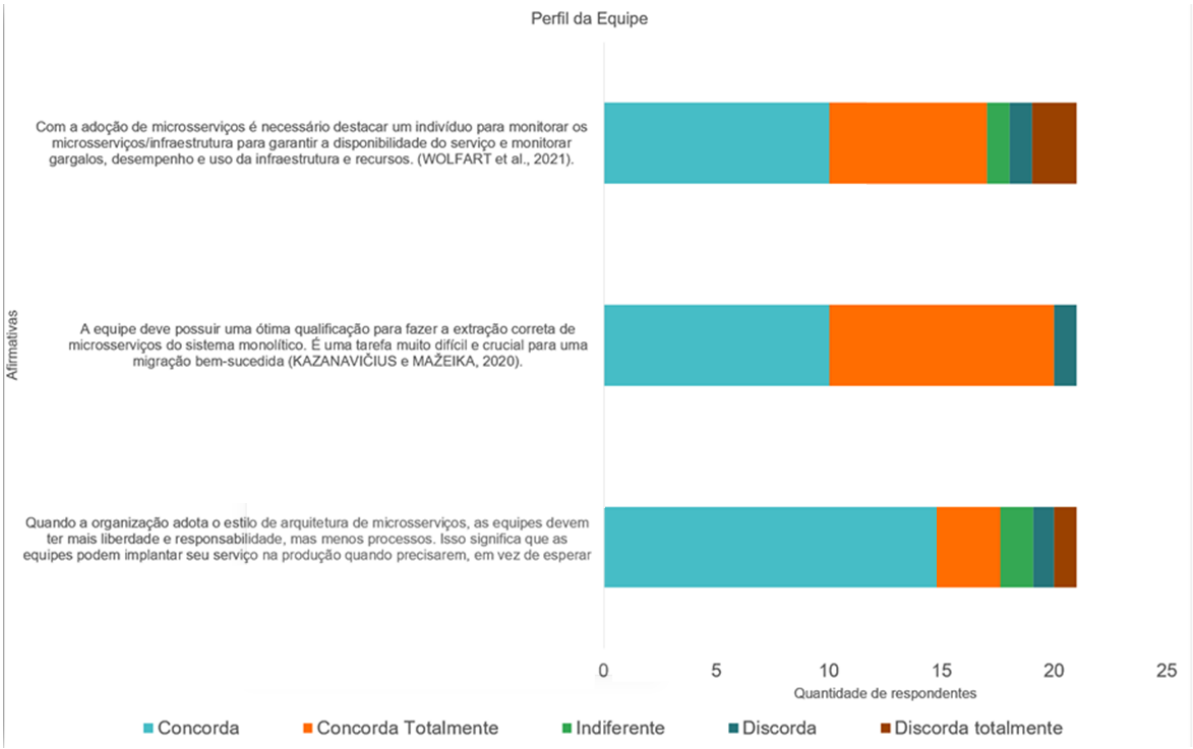
5.2.8. Perfil da Equipe para realizar a migração

Este tópico foi o que gerou mais divergências nas respostas. As seguintes afirmações foram submetidas aos especialistas:

- Quando a organização adota o estilo de arquitetura de Microsserviços, as equipes devem ter mais liberdade e responsabilidade, mas menos processos. Isso significa que as equipes podem implantar seu serviço na produção quando precisarem, em vez de esperar pela aprovação de outra pessoa (KALSKE; MÄKITALO; MIKKONEN, 2018).
- A equipe deve possuir uma ótima qualificação para fazer a extração correta de Microsserviços do sistema monolítico. É uma tarefa muito difícil e crucial para uma migração bem-sucedida (KAZANAVIČIUS; MAŽEIKA, 2020).
- Com a adoção de Microsserviços é necessário destacar um indivíduo para monitorar os Microsserviços/infraestrutura para garantir a disponibilidade do serviço e monitorar gargalos, desempenho e uso da infraestrutura e recursos (WOLFART et al., 2021).

O gráfico da Figura 22 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 22 – Escala Likert Perfil da Equipe



Fonte: autor.

Como apresenta o gráfico da Figura 22, na afirmativa 1, 48% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 33% e 15%, respectivamente; 30% responderam “Discordo”; 10%, “Indiferente”; e 5%, “Discordo totalmente”. A afirmativa 2 apresentou 95% das respostas entre “Concordo” (48%) e “Concordo totalmente” (47%); 5% dos especialistas responderam “Discordo”; e não houve respostas para “Indiferente” e “Discordo totalmente”. Na afirmativa 3, 62%, 19%, 10%, 5% e 5% dos especialistas responderam “Concordo”, “Concordo totalmente”, “Discordo”, “Discordo totalmente” e “Indiferente”, respectivamente.

A escala “Concordo” tem média de 48%, e “Concordo totalmente”, 27%. A soma dessas duas médias resulta em 75%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 75% de concordância com as orientações. Dessa forma, as orientações foram validadas junto aos especialistas.

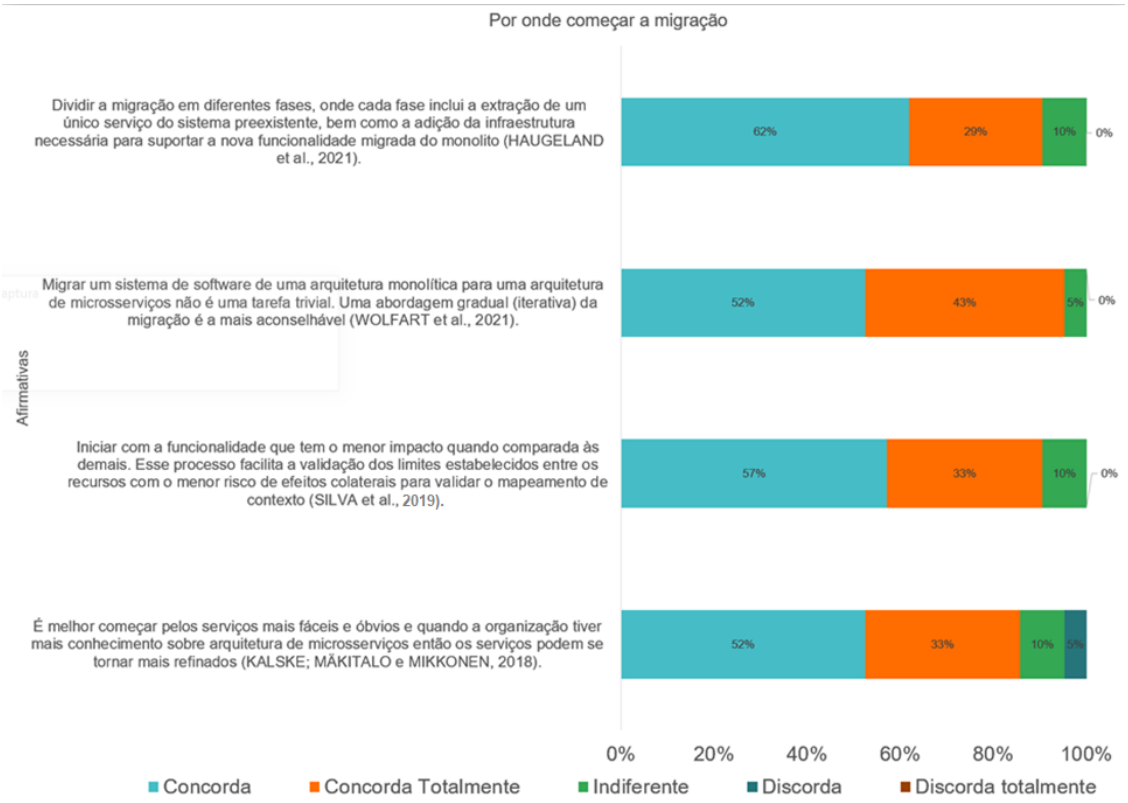
5.2.9. Por Onde Começar a migração

Este tópico aborda por onde a migração deve iniciar. As seguintes afirmações foram submetidas aos especialistas:

- Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021).
- Migrar um sistema de software de uma arquitetura monolítica para uma arquitetura de Microsserviços não é uma tarefa trivial. Uma abordagem gradual (iterativa) da migração é a mais aconselhável (WOLFART et al., 2021).
- Iniciar com a funcionalidade que tem o menor impacto quando comparada às demais. Esse processo facilita a validação dos limites estabelecidos entre os recursos com o menor risco de efeitos colaterais para validar o mapeamento de contexto (SILVA; CARNEIRO; MONTEIRO, 2019).
- É melhor começar pelos serviços mais fáceis e óbvios e quando a organização tiver mais conhecimento sobre arquitetura de Microsserviços então os serviços podem se tornar mais refinados (KALSKE; MÄKITALO; MIKKONEN, 2018).

O gráfico da Figura 23 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 23 – Escala Likert Por Onde Começar a migração



Fonte: autor.

Como apresenta o gráfico da Figura 23, na afirmativa 1, 91% dos especialistas responderam entre “Concordo” (62%) e “Concordo totalmente” (29%); 10% responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”. A afirmativa 2 apresentou 95% das respostas entre “Concordo” (52%) e “Concordo totalmente” (43%); 5% dos especialistas responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”. Na afirmativa 3, os especialistas responderam “Concordo” (57%), “Concordo totalmente” (33%) e “Indiferente” (10%), não havendo respostas para “Discordo” e “Discordo totalmente”. Na afirmativa 4, 85% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 52% e 33%, respectivamente; 10% responderam “Indiferente”; 5%, “Discordo”; e não houve respostas para “Discordo totalmente”.

A escala “Concordo” tem média de 56%, e “Concordo totalmente”, 35%. A soma dessas duas médias resulta em 90%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 90% de concordância com as orientações. Dessa forma, as orientações foram validadas junto aos especialistas.

5.2.10. UML para identificar os microsserviços candidatos

A importância de identificar os candidatos a microsserviços é tão grande que este tópico aparece duas vezes nesta pesquisa. Inicialmente foi abordado o método DDD e agora UML. As seguintes afirmações foram submetidas aos especialistas:

- Contextos limitados podem ser separados por meio da decomposição por verbos (casos de uso) ou por substantivos (recursos) e dessa forma é possível identificar os Microsserviços candidatos (FRITZSCH et al., 2019).
- O primeiro passo é a transformação do monolito na representação gráfica. No grafo, cada vértice representa a classe do monolito e as arestas não direcionadas representam seu acoplamento com outras classes do monolito. Ao cortar o gráfico em componentes é possível observar os candidatos a Microsserviços (SILVA; CARNEIRO; MONTEIRO, 2019).
- Através de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML. Em seguida listar todas as funcionalidades para as quais o sistema foi projetado e para identificar os Microsserviços candidatos, deve fazer a identificação dos artefatos de implementação de cada recurso (WOLFART et al., 2021).

O gráfico da Figura 24 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 24 – Escala Likert UML para identificar os microsserviços candidatos

Fonte: autor.

Como apresenta o gráfico da Figura 24, na afirmativa 1, 76% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 52% e 24%, respectivamente; 24% responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”. A afirmativa 2 apresentou 62% das respostas entre “Concordo” (52%) e “Concordo totalmente” (10%); 33% dos especialistas responderam “Indiferente”; 5%, “Discordo”; e não houve respostas para “Discordo totalmente”. A afirmativa 3 apresentou todas as respostas: “Concordo” (43%), “Concordo totalmente” (14%), “Discordo” (5%), “Discordo totalmente” (5%) e “Indiferente” (33%).

A escala “Concordo” tem média de 49%, e “Concordo totalmente”, 14%. A soma dessas duas médias resulta em 63%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 63% de concordância com as orientações. Dessa forma, essas questões serão aprimoradas e uma nova rodada será aplicada.

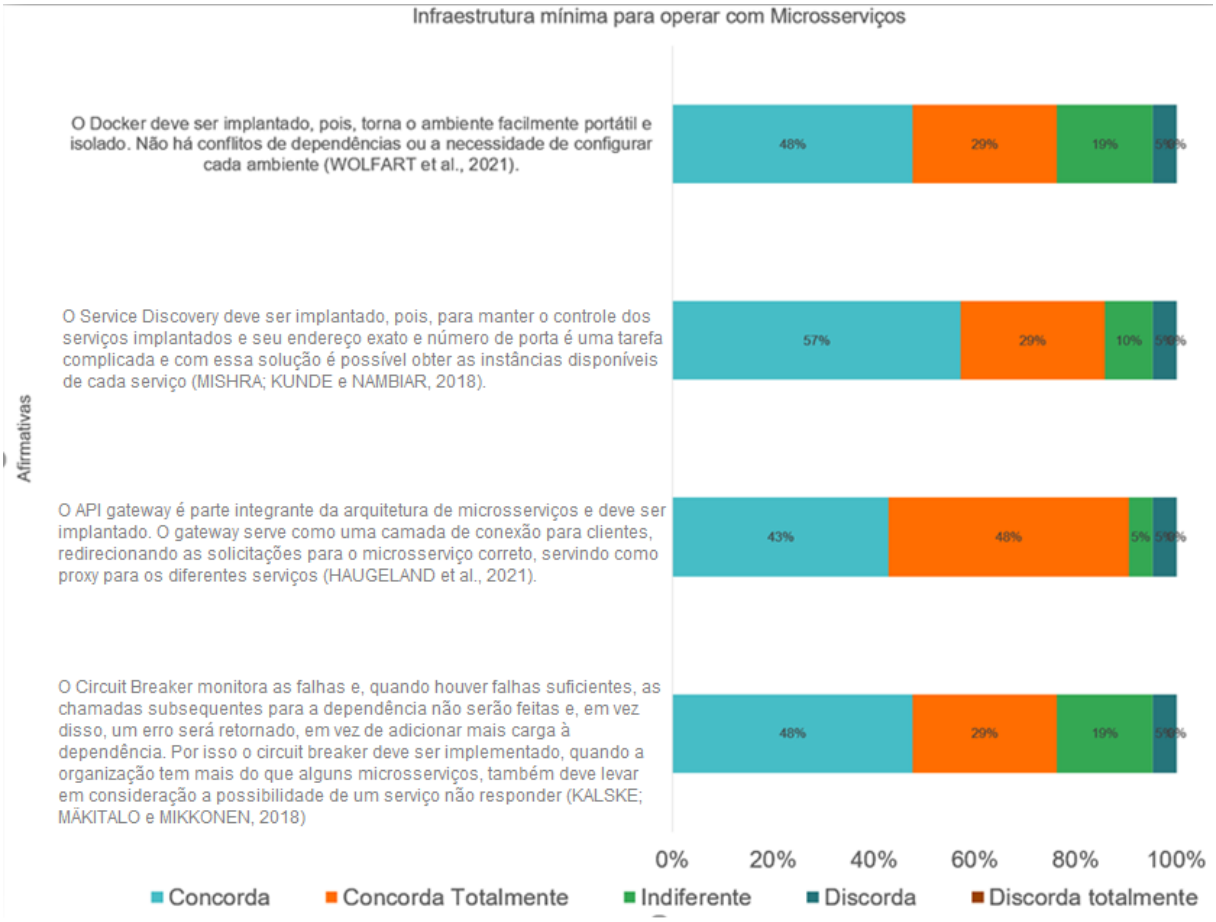
5.2.11. Infraestrutura mínima para utilizar microsserviços

Este tópico aborda a infraestrutura para a empresa suportar uma operação utilizando microsserviços. As seguintes afirmações foram submetidas aos especialistas:

- O Docker deve ser implantado, pois torna o ambiente facilmente portátil e isolado. Não há conflitos de dependências ou a necessidade de configurar cada ambiente (WOLFART et al., 2021).
- O Service Discovery deve ser implantado, pois, para manter o controle dos serviços implantados e seu endereço exato e número de porta é uma tarefa complicada e com essa solução é possível obter as instâncias disponíveis de cada serviço (MISHRA; KUNDE; NAMBIAR, 2018).
- O API gateway é parte integrante da arquitetura de Microsserviços e deve ser implantado. O gateway serve como uma camada de conexão para clientes, redirecionando as solicitações para o microsserviço correto, servindo como proxy para os diferentes serviços (HAUGELAND et al., 2021).
- O Circuit Breaker monitora as falhas e, quando houver falhas suficientes, as chamadas subsequentes para a dependência não serão feitas e, em vez disso, um erro será retornado, em vez de adicionar mais carga à dependência. Por isso o circuit breaker deve ser implementado, quando a organização tem mais do que alguns Microsserviços, também deve levar em consideração a possibilidade de um serviço não responder (KALSKE; MÄKITALO; MIKKONEN, 2018).

O gráfico da Figura 25 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 25 – Escala Likert Infraestrutura para operar com microsserviços



Fonte: autor.

Como apresenta o gráfico da Figura 25, na afirmativa 1, 77% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 48% e 29%, respectivamente; 19% responderam “Indiferente”; 5%, “Discordo”; e não houve respostas para “Discordo totalmente”. A afirmativa 2 apresentou 86% das respostas entre “Concordo” (57%) e “Concordo totalmente” (29%); 10% dos especialistas responderam “Indiferente”; 5%, “Discordo”; e não houve respostas para “Discordo totalmente”. A afirmativa 3 apresentou as respostas “Concordo” (43%), “Concordo totalmente” (48%), “Discordo” (5%) e “Indiferente” (5%), não havendo respostas para “Discordo totalmente”. Na afirmativa 4, 77% dos especialistas responderam entre “Concordo” e “Concordo totalmente” — 48% e 29%, respectivamente; 19% responderam “Indiferente”; 5%, “Discordo”; e não houve respostas para “Discordo totalmente”.

A escala “Concordo” tem média de 49%, e “Concordo totalmente”, 35%. A soma dessas duas médias resulta em 84%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 84% de concordância com as orientações. Dessa forma, as orientações foram validadas junto aos especialistas.

5.2.12. Revisão da rodada 1

A rodada 1 não alcançou o consenso somente no item 5.2.10 – UML para identificar os microsserviços candidatos. Dessa forma, uma nova rodada foi realizada, com o item submetido novamente aos especialistas, agora com novas explicações e ajustes no questionário para obter um *feedback* mais preciso.

5.2.13. Execução do método Delphi (rodada 2)

De acordo com o fluxo da Figura 14 (método Delphi), as orientações relacionadas ao tópico “UML para identificar os microsserviços candidatos” foram revisadas e uma nova rodada do questionário foi publicada e respondida por 20 especialistas entre os dias 17 de abril e 05 de maio de 2023, por meio do Google Form.

O perfil completo desses profissionais entrevistados encontra-se no Apêndice H. Para a execução das rodadas, na busca de consenso, foram enviados 67 convites por meio de mensagens privadas no LinkedIn, incluindo uma carta-convite (disponível no Apêndice A) na qual são descritos os objetivos da pesquisa.

5.2.14. Declarações da literatura

As afirmativas do questionário publicado e enviado aos especialistas sofreram alterações para atender aos resultados do teste de face. Os itens afetados e alterados serão apontados a seguir.

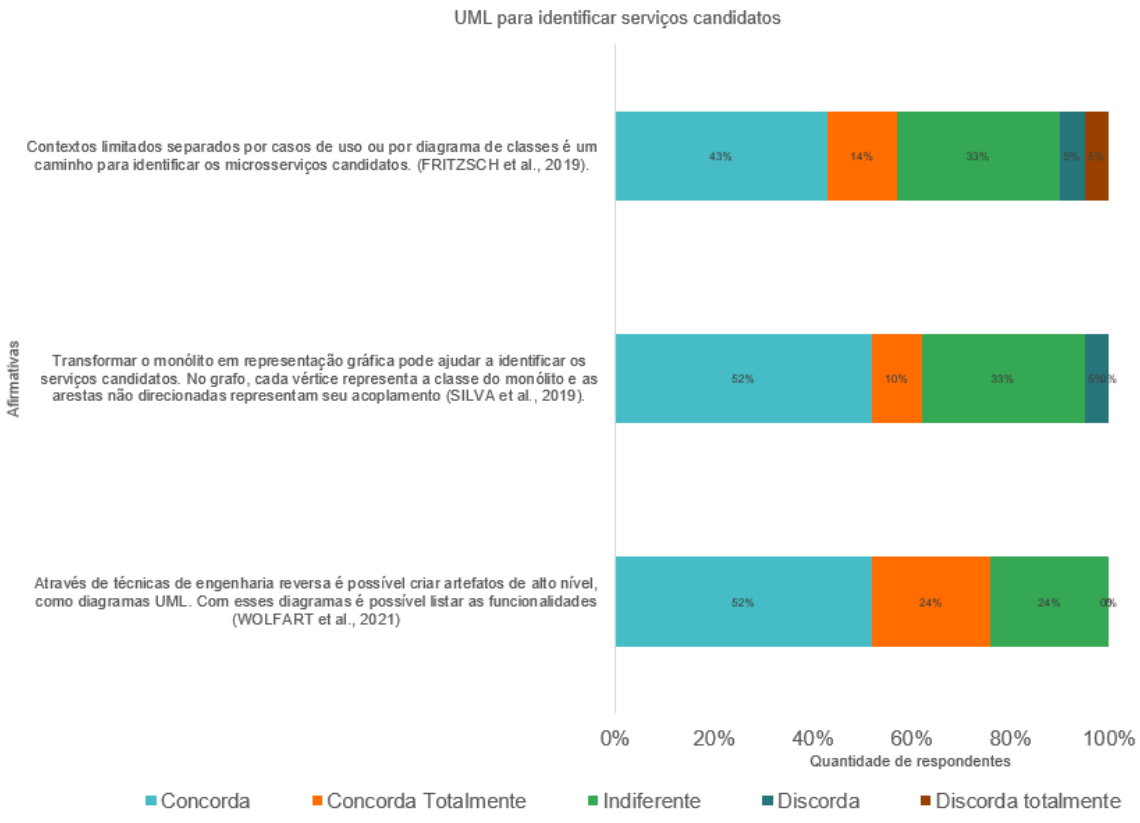
5.2.15. UML para identificar os microsserviços candidatos (revisão)

As orientações relacionadas a este tópico foram revisadas e uma nova rodada de avaliação foi submetida aos especialistas, com as afirmações a seguir.

- Contextos limitados separados por casos de uso ou por diagrama de classes é um caminho para identificar os microsserviços candidatos (FRITZSCH et al., 2019).
- Transformar o monolito em representação gráfica pode ajudar a identificar os serviços candidatos. No grafo, cada vértice representa a classe do monolito e as arestas não direcionadas representam seu acoplamento com outras classes do monolito. Ao cortar o grafo em componentes é possível observar os candidatos a microsserviços (SILVA; CARNEIRO; MONTEIRO, 2019).
- Através de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML. Com esses diagramas é possível listar as funcionalidades para as quais o sistema foi projetado e isso ajudará a identificar os Microsserviços candidatos (WOLFART et al., 2021).

O gráfico da Figura 26 apresenta as respostas de acordo com a escala Likert das afirmativas anteriores.

Figura 26 – Escala Likert UML para identificar microsserviços candidatos (rodada 2)



Fonte: autor.

Como apresenta o gráfico da Figura 26, na afirmativa 1, 90% dos especialistas responderam entre “Concordo” e “Concordo totalmente”; 10% responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”. A afirmativa 2 apresentou 90% das respostas entre “Concordo” (80%) e “Concordo totalmente” (10%); 10% dos especialistas responderam “Indiferente”; e não houve respostas para “Discordo” e “Discordo totalmente”. A afirmativa 3 apresentou as respostas “Concordo” (65%), “Concordo totalmente” (10%), “Indiferente” (15%) e “Discordo” (10%), não havendo respostas para “Discordo totalmente”.

A escala “Concordo” tem média de 78%, e “Concordo totalmente”, 7%. A soma dessas duas médias resulta em 85%. Na Curva ABC, os tópicos foram indicados como relevantes com 72%, e na escala Likert, os respondentes indicam 85% de concordância com as orientações. Dessa forma, o item foi validado junto aos especialistas.

5.2.16. Revisão da rodada 2

A Rodada 2 alcançou o consenso no item 5.2.10 – UML para identificar os microsserviços candidatos. Dessa forma, não foi necessário realizar uma rodada adicional.

5.3. ANÁLISE DAS RESPOSTAS APÓS O CONSENSO OBTIDO (MÉTODO DELPHI)

O guia de orientações construído e disponibilizado foi baseado em uma RSL, que permitiu identificar e classificar as orientações mais relevantes na adoção de microsserviços em nuvem. A RSL também forneceu uma visão acadêmica sobre o assunto, a partir de artigos científicos publicados.

Além disso, a fim de contribuir para o mercado, a pesquisa foi realizada no campo por meio de um questionário para capturar a percepção de especialistas em desenvolvimento de *software*. Essa consulta permitiu validar o guia de orientações e proporcionou uma perspectiva mais pragmática sobre o assunto, conectando-se com o que acontece nas empresas e as percepções dos especialistas.

Vinte e uma orientações foram identificadas, das quais oito foram categorizadas como mais relevantes pela Curva ABC (Quadro 10). Essas oito categorias representam 72% das citações observadas nos trabalhos acadêmicos resultantes da RSL. Com base nessa revisão, cada categoria contou com três ou, em alguns casos, quatro afirmativas que foram incluídas em um questionário de escala Likert, apresentando também uma área para comentários livres.

O guia de orientações foi validado de acordo com as expectativas dos objetivos da pesquisa, já que a maioria das orientações foi validada com sucesso. Como critério deste trabalho, considerou-se que uma orientação é válida quando a soma das médias das respostas “Concordo totalmente” e “Concordo” é superior a 72%, o mesmo critério utilizado na aplicação da Curva ABC; isso aconteceu com os oito tópicos.

Embora a RSL tenha proporcionado bases sólidas para a identificação das orientações mais relevantes na adoção de microsserviços em nuvem, havia uma preocupação sobre a aplicabilidade em MEs. No entanto, o fato positivo é que 76% dos respondentes afirmam que qualquer empresa pode adotar microsserviços em nuvem em sua operação, como pode ser observado na Tabela 1 da seção 5.2.1. Essa aprovação torna este guia ainda mais relevante para as MEs de *software*.

6. DESENVOLVIMENTO DO GUIA DE ORIENTAÇÕES

Esta seção apresenta um conjunto de orientações direcionadas às MEs que desejam migrar de sistemas com arquitetura monolítica para arquitetura de microsserviços em nuvem. O guia visa auxiliar as empresas na avaliação do seu estado atual em relação aos requisitos necessários para o processo de migração e, ao mesmo tempo, possibilitar a implementação de estratégias para tornar a migração mais eficiente.

As orientações apresentadas foram cuidadosamente analisadas e extraídas da literatura, além de terem sido validadas por especialistas em desenvolvimento de *software* do mercado brasileiro.

6.1. VISÃO DO GUIA DE ORIENTAÇÕES

As diretrizes que orientam as empresas a implementarem o guia contempla oito tópicos extraídos da literatura, selecionados após a aplicação da Curva ABC e mostrados na Figura 27.

Figura 27 – Visão geral das orientações do guia

Fonte: autor.

Os tópicos em questão foram aprovados por especialistas por meio da aplicação do método de pesquisa Delphi. O perfil desses especialistas foi dividido em três áreas de conhecimento relacionadas ao desenvolvimento de *software*, sendo 43% especialistas em engenharia de *software*, 33% especialistas em programação e 24% especialistas em arquitetura de *software*. A maioria dos especialistas (57%) tem mais de 10 anos de experiência, 33% têm entre 6 e 10 anos de experiência e apenas 10% têm entre 3 e 5 anos de experiência.

6.2. DESCRIÇÃO DAS ORIENTAÇÕES DO GUIA

A seguir são apresentadas as descrições das orientações do guia.

6.2.1. Orientação 1: Tópicos Gerais

Embora tenham sido identificados vários assuntos importantes na RSL, nem todos foram abordados no guia de orientações. Para selecionar os tópicos a serem detalhados, foi utilizada a Curva ABC, que apontou os mais relevantes (ver Quadro 10, no item 4.3). Isso permitiu uma simplificação de acordo com essa métrica, mas é importante destacar que outros aspectos devem ser avaliados antes de iniciar a migração. Esses aspectos foram agrupados na Orientação 1 e apresentados na Figura 28.

Figura 28 – Orientação 1: Tópicos Gerais

Fonte: autor.

Detalhamento dos subtópicos:

A) Granularidade do microsserviço

- Esse é um aspecto crítico no processo de migração, já que uma equipe sem maturidade adequada poderá transportar os problemas existentes nos códigos monolíticos para o ambiente dos microsserviços, caso não sejam adequadamente definidos os níveis de coesão e autonomia desses componentes. É essencial, portanto, que a equipe esteja plenamente consciente da importância dessa fase do processo e da necessidade de atenção aos detalhes para assegurar a consistência da migração.

B) Testes integrados

- Ao adotar testes integrados durante o processo de migração para microsserviços, há uma tendência de melhoria na qualidade das entregas em produção. No entanto, é essencial que a equipe tenha conhecimento prévio para construir adequadamente essa solução na esteira de entregas do método DevOps. Dessa forma, é possível garantir a eficácia da implementação dos testes integrados, reduzindo a ocorrência de problemas e permitindo uma transição mais suave para o ambiente de microsserviços.

C) Cultura da empresa

- A implementação do processo de integração e entrega contínuas (CI/CD) dentro do método DevOps pode proporcionar mais autonomia às equipes para a disponibilização dos serviços em produção. No entanto, em empresas nas quais é necessária aprovação em cada etapa, essa abordagem pode se tornar um gargalo para as entregas, demandando uma mudança na cultura gerencial da organização.
- Para garantir o desempenho e a disponibilidade dos serviços em um ambiente de nuvem, é fundamental que a empresa disponha de recursos humanos dedicados ao monitoramento constante desse ambiente. É importante que esses profissionais tenham conhecimento técnico adequado para identificar e solucionar eventuais problemas que possam surgir. Além disso, é recomendável investir em ferramentas de monitoramento automatizado, que possam auxiliar na detecção e na correção de falhas de maneira mais rápida e eficiente.

D) Log centralizado

- É uma atividade adicional na qual a equipe de desenvolvimento deve se atentar ao registro de informações precisas no log de erros. Embora a Nuvem já forneça uma série de informações e ferramentas, em muitos casos, essas informações são distribuídas em diferentes ferramentas e sistemas, o que pode dificultar a visualização e a correlação dos dados em caso de problemas ou erros que afetem o funcionamento do sistema como um todo.
- Por essa razão, muitas equipes optam por construir um log centralizado, que permite que todas as informações relevantes para a operação do sistema sejam armazenadas em um único local, o que facilita a identificação e a solução de problemas, além de permitir a análise histórica dos dados para fins de otimização de *performance* e melhoria contínua.
- Além disso, um log centralizado pode ser configurado para atender às necessidades específicas da equipe de infraestrutura e desenvolvimento, incluindo a definição de níveis de detalhamento, formatos de registro e políticas de retenção de dados. Dessa forma, um log centralizado é uma ferramenta importante para garantir a efetividade e a eficiência da monitoração e da solução de problemas em sistemas em nuvem.

E) Cuidados no desenvolvimento

- A equipe responsável pelo desenvolvimento de microsserviços em ambiente de nuvem deve estar capacitada para lidar com os desafios de sistemas distribuídos, que incluem questões como transações, sobrecarga de rede, segurança e falhas na invocação de serviços, entre outras.
- É fundamental que os profissionais envolvidos tenham conhecimento aprofundado sobre esses aspectos e estejam aptos a aplicar as melhores práticas no desenvolvimento de soluções que apresentem alto desempenho, escalabilidade e disponibilidade em ambientes de nuvem.

F) Motivação para migrar

- Segundo os especialistas consultados, as cinco maiores motivações para realizar a migração são:
  - Reduzir o acoplamento (27,5%);
  - Melhoria na escalabilidade (27,5%);
  - Aumento da autonomia das equipes (15%);

- Melhoria de *performance* (12,5%);
- Redução de custos (7,5%).

- Em relação aos objetivos iniciais da migração para arquitetura baseada em microsserviços, 70,8% dos especialistas entrevistados relatam que os objetivos foram atingidos em sua totalidade; 17,6%, que os objetivos foram atingidos parcialmente; e 11,8%, que os objetivos não foram atingidos.
- Esses dados evidenciam que, embora a maioria das equipes de desenvolvimento tenha conseguido atingir seus objetivos iniciais com sucesso, ainda há uma parcela significativa que enfrentou desafios ao longo do processo de migração. Portanto, é importante que as equipes se preparem adequadamente e adotem as melhores práticas para garantir o sucesso da migração.

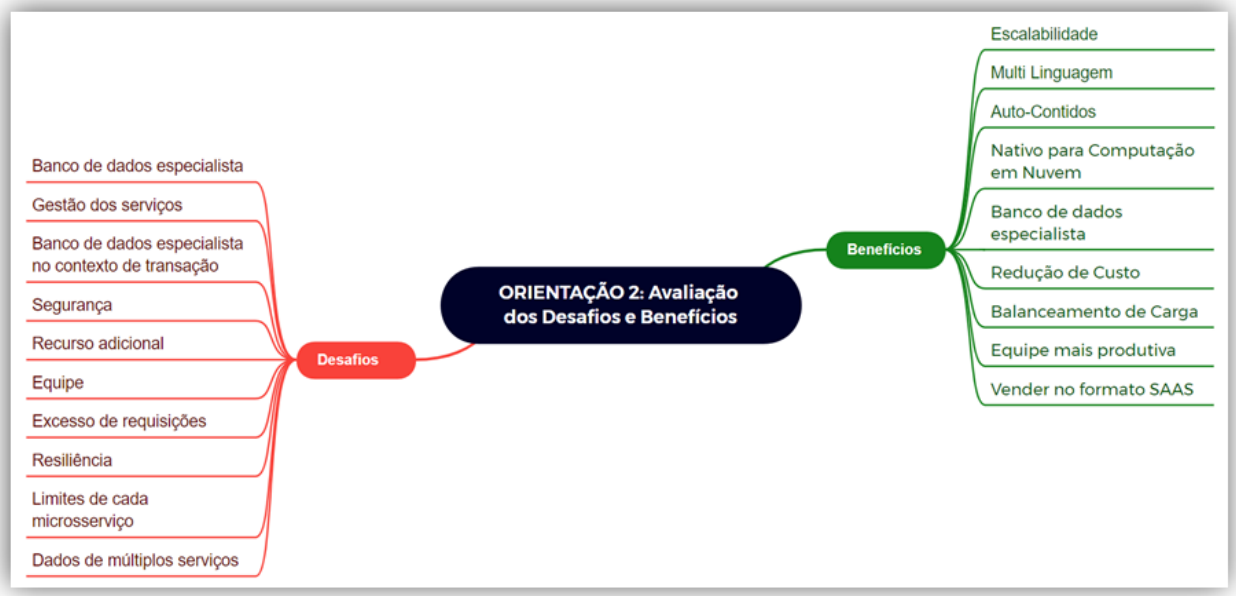
G) Visão Geral: Métricas mais utilizadas

- Segundo os especialistas consultados, as cinco métricas mais utilizadas para realizar a migração e medir o resultado após a migração são:
  - Apdex (34%);
  - Quantidade de acessos/transações (20%);
  - Custo (20%);
  - Tempo para entregar *software* em produção (13%);
  - Tempo de disponibilidade (8%).
- A métrica mais utilizada é o Apdex, representando 34% das respostas dos entrevistados. A quantidade de acessos/transações e o custo foram mencionados em segundo lugar, ambos com 20%. Em seguida, o tempo para entrega em produção foi mencionado por 14% dos entrevistados, enquanto o tempo de disponibilidade foi citado por apenas 6%. Essas métricas são essenciais para avaliar a qualidade e a eficiência de um sistema.

6.2.2. Orientação 2: Avaliação dos Desafios e Benefícios

A seguir serão descritos os desafios e os benefícios na implantação de microsserviços em nuvem, apresentados na Figura 29.

Figura 29 – Orientação 2: Avaliação dos Desafios e Benefícios



SaaS: *Software as a Service*.

Fonte: autor.

Detalhamento dos subtópicos:

Desafios

A) Banco de dados especialista (ou microbanco de dados)

- A necessidade de gerenciar uma grande quantidade de instâncias de bancos de dados pode ser complexa e exigir muito trabalho manual, além de aumentar o custo operacional. Também é necessário garantir a escalabilidade e a disponibilidade do banco de dados, bem como a segurança e a privacidade dos dados armazenados em cada instância.
- Além disso, a heterogeneidade de tecnologias e plataformas de bancos de dados usadas pelos diferentes microsserviços pode dificultar a implementação de uma estratégia de gerenciamento de dados unificada.

B) Banco de dados especialista no contexto de transação

- Para garantir a integridade dos dados em um ambiente distribuído baseado em microsserviços, é fundamental implementar técnicas de controle de concorrência e transações distribuídas, além de monitorar o desempenho do banco de dados para garantir a disponibilidade. A segurança e a privacidade dos dados também devem ser garantidas por meio de mecanismos de autenticação e autorização robustos.

- No contexto de bancos de dados distribuídos, uma das técnicas de controle de concorrência mais utilizadas é o bloqueio otimista, que permite que vários usuários possam acessar simultaneamente o mesmo dado, desde que não haja conflito de escrita. Além disso, as transações distribuídas são gerenciadas por meio de coordenadores distribuídos, que garantem a integridade transacional e a consistência dos dados em todo o sistema de microsserviços.

C) Segurança

- Nos sistemas monolíticos, não havia a preocupação com a segurança entre os módulos, pois estavam todos compilados em um mesmo pacote. Porém, na arquitetura baseada em microsserviços, a preocupação com a segurança das informações deve estar sempre presente.
- Os serviços devem ser invocados somente após a autenticação, e o ideal seria que os níveis de privilégio fossem definidos de acordo com cada requisição, para garantir que apenas usuários autorizados possam acessar as informações sensíveis. Dessa forma, é possível proteger os dados e minimizar os riscos de invasões ou violações de segurança.

D) Recurso adicional

- De acordo com especialistas, a adoção de microsserviços requer uma abordagem mais proativa em relação ao monitoramento da infraestrutura, com 62% deles afirmando que é necessário destacar um indivíduo para essa tarefa. A disponibilidade do serviço, assim como a identificação de gargalos, desempenho e uso de recursos, são aspectos cruciais que devem ser monitorados constantemente.
- Essa responsabilidade requer habilidades e conhecimentos técnicos específicos para garantir que os problemas e as falhas no sistema sejam identificados e corrigidos em tempo hábil, assegurando que a infraestrutura opere de maneira otimizada e atenda às necessidades dos usuários.

E) Equipe

- Sistemas distribuídos são mais complexos e, por isso, é necessário ter uma equipe qualificada para lidar com os desafios que essa arquitetura apresenta. A equipe precisa ter conhecimentos sólidos em diversas áreas, como programação, bancos de dados, redes, segurança, entre outras. Além

disso, é importante que os membros da equipe estejam sempre atualizados sobre as novas tecnologias e tendências em sistemas distribuídos, para que possam implementar as soluções mais adequadas e eficientes.

- Ter uma equipe qualificada é essencial para garantir o sucesso da arquitetura baseada em microsserviços e para garantir que o sistema esteja sempre funcionando de forma eficiente e segura.

F) Gestão dos serviços

- É necessário adotar um protocolo mínimo para a gestão dos serviços, pois uma alteração mal planejada na interface pode afetar diversos outros serviços. Alguns dos itens importantes que devem ser considerados incluem versionamento, relacionamentos, disponibilidade, segurança, escalabilidade e monitoramento do uso do serviço. O versionamento é importante para garantir a compatibilidade entre diferentes versões do serviço, permitindo que os usuários possam continuar utilizando o serviço sem problemas.
- Os relacionamentos devem ser bem definidos para evitar dependências desnecessárias entre os serviços. A disponibilidade do serviço deve ser garantida por meio do monitoramento constante e da aplicação de práticas de redundância. A segurança é essencial para proteger as informações e garantir a integridade dos dados.
- A escalabilidade deve ser planejada desde o início, para que o serviço possa crescer conforme a demanda. E, finalmente, o monitoramento do uso do serviço é importante para garantir que o sistema esteja funcionando de forma otimizada e para identificar possíveis problemas antes que se tornem críticos.

G) Excesso de requisições

- É importante realizar um planejamento prévio, apoiado por testes de carga e testes de estresse, para mapear o comportamento do serviço quando houver excesso de requisições. Os testes de carga ajudam a medir o desempenho sob uma carga esperada, enquanto os testes de estresse sobrecarregam o serviço para encontrar o ponto de ruptura.
- Esses testes são essenciais para identificar possíveis gargalos ou problemas de desempenho que possam afetar o serviço quando houver um aumento repentino de requisições. Com base nos resultados desses testes,

é possível ajustar a arquitetura do sistema, otimizar o desempenho e garantir que o serviço esteja preparado para lidar com qualquer quantidade de requisições.

- Dessa forma, é possível garantir a disponibilidade e a confiabilidade do serviço, mesmo em situações de alta demanda.

H) Resiliência

- É importante considerar que o serviço, a rede e o banco de dados podem falhar a qualquer momento. Por isso, é necessário adotar certas implementações para garantir a resiliência do sistema. Algumas dessas implementações incluem o uso de Health Endpoint, padrão Retry, padrão Circuit Breaker e padrão Bulkhead. O Health Endpoint é utilizado para monitorar a disponibilidade do serviço e verificar se ele está respondendo corretamente às requisições.
- O padrão Retry é utilizado para tentar novamente uma operação que falhou, com o objetivo de evitar a interrupção do serviço. O padrão Circuit Breaker é utilizado para evitar que um erro em um serviço se propague para o sistema inteiro, isolando o serviço em questão até que o problema seja resolvido. O padrão Bulkhead é utilizado para separar diferentes partes do sistema em compartimentos isolados, de forma que um problema em uma parte do sistema não afete as outras partes.
- Com a adoção dessas implementações, é possível garantir que o sistema esteja preparado para lidar com falhas e interrupções, garantindo a disponibilidade e a confiabilidade do serviço em todas as situações.

I) Limites de cada microsserviço

- Definir os limites de cada microsserviço não é uma tarefa trivial e pode trazer sérias consequências se não for realizada adequadamente. Uma decisão mal planejada pode dificultar a escalabilidade, a manutenção e a evolução do sistema, além de impedir que o sistema possa se beneficiar das facilidades buscadas com a adoção da arquitetura baseada em microsserviços.
- É importante avaliar cuidadosamente a complexidade de cada serviço e definir seus limites com base nas necessidades específicas do negócio e dos usuários. Dessa forma, é possível garantir que cada microsserviço

esteja cumprindo sua função de maneira eficiente, sem criar dependências desnecessárias entre os serviços e sem prejudicar a arquitetura geral do sistema.

- É importante lembrar que os limites de cada microsserviço podem mudar ao longo do tempo, e é necessário estar sempre atento às mudanças no ambiente para realizar ajustes e melhorias quando necessário.

J) Dados de múltiplos serviços

- Recuperar dados de vários microsserviços pode ser um desafio, especialmente quando é necessário criar consultas que envolvem dados de diferentes serviços. É necessário definir uma estratégia para lidar com esses casos, considerando as implicações de segurança, desempenho e escalabilidade.
- Uma opção é utilizar um serviço intermediário, como uma API de agregação de dados, que pode consultar múltiplos serviços e retornar os dados em um formato consolidado. Outra opção é utilizar técnicas de integração de dados, como o mapeamento de dados, que permite mapear os dados de diferentes fontes para um formato comum, facilitando a sua análise.
- Independentemente da estratégia escolhida, é importante garantir que a arquitetura do sistema esteja preparada para lidar com consultas complexas que envolvem dados de múltiplos microsserviços, de forma que seja possível manter a consistência e a integridade dos dados.
- Com uma equipe qualificada, é possível lidar com os desafios da arquitetura baseada em microsserviços e extrair o máximo de benefícios dessa abordagem.

Benefícios

K) Escalabilidade

- Uma das principais vantagens da arquitetura baseada em microsserviços é a escalabilidade, que permite que cada serviço seja escalado de forma independente para atender à demanda do aplicativo. Isso significa que é possível dimensionar a infraestrutura de forma mais eficiente, tanto de forma horizontal (adicionando mais instâncias do serviço) quanto de forma vertical (aumentando os recursos das instâncias existentes).

- Com a escalabilidade, é possível garantir que o sistema esteja sempre preparado para lidar com picos de demanda e que seja possível manter a disponibilidade e a confiabilidade do serviço em todas as situações. Além disso, a escalabilidade permite que o sistema seja mais flexível e adaptável às necessidades do negócio, facilitando a sua evolução e expansão.
- Porém, é importante lembrar que a escalabilidade requer um planejamento cuidadoso, que leve em consideração os recursos disponíveis, as limitações do sistema e as necessidades específicas do negócio.

L) Autocontidos

- A arquitetura baseada em microsserviços permite a decomposição de grandes estruturas de *software* em serviços independentes e autocontidos, oferecendo maior flexibilidade e controle de custos nas soluções de *software*. Com essa abordagem, cada serviço pode ser gerenciado de forma independente e escalonado de acordo com a demanda do aplicativo, o que facilita a manutenção e a evolução do sistema, e permite que cada serviço seja atualizado de forma isolada, sem afetar os demais serviços.
- Ao adotar a arquitetura baseada em microsserviços, é possível criar soluções de *software* mais flexíveis, escaláveis e adaptáveis às necessidades do negócio, garantindo maior agilidade e eficiência no desenvolvimento e na entrega de *software*. A possibilidade de gerenciar cada serviço de forma independente e escaloná-lo conforme a demanda oferece maior controle e eficiência na gestão de custos e recursos, garantindo que o sistema esteja sempre preparado para lidar com as demandas do mercado e capaz de evoluir e se adaptar às mudanças do negócio.

M) Nativo para Computação em Nuvem

- Os microsserviços foram projetados para ambientes em nuvem, em que a escalabilidade horizontal e a disponibilidade são fundamentais. Distribuídos e executados em contêineres, como Kubernetes e Docker, são fáceis de gerenciar e implantar em plataformas em nuvem, como AWS, Google Cloud e Microsoft Azure. Essas tecnologias permitem a implantação rápida de novas instâncias de microsserviços para lidar com a demanda, tornando a escalabilidade horizontal mais fácil de ser alcançada.

- Por exemplo, uma aplicação de comércio eletrônico pode ser dividida em microsserviços independentes, como gerenciamento de pedidos, gerenciamento de produtos e processamento de pagamentos, cada um executado em seu próprio contêiner. Dessa forma, a aplicação pode ser facilmente dimensionada horizontalmente, respondendo à demanda em tempo real e garantindo a disponibilidade e a eficiência do sistema em nuvem.

N) Multilinguagem

- Uma das vantagens da arquitetura de microsserviços é a possibilidade de desenvolver diferentes serviços com tecnologias diferentes do projeto principal, de acordo com a demanda. Por exemplo, um serviço de processamento de imagem pode ser desenvolvido em Python, enquanto um serviço de envio de e-mails pode ser desenvolvido em Java.
- Essa abordagem permite escolher a melhor tecnologia para cada serviço, de acordo com suas necessidades, sem afetar o projeto principal. Além disso, essa abordagem também permite utilizar diferentes tipos de banco de dados para cada serviço, como bancos de dados não relacionais, que são ideais para serviços que demandam alta escalabilidade e flexibilidade de estruturação de dados, como o MongoDB.
- Essa abordagem de desenvolvimento permite maior flexibilidade e agilidade no desenvolvimento e na manutenção de cada serviço, além de permitir que os serviços possam ser atualizados de forma independente, sem impactar outros serviços ou o projeto principal.

O) Banco de dados especialista

- Com a separação das bases de dados, é possível utilizar bancos de dados especialistas para cada serviço. Por exemplo, um serviço de autenticação pode utilizar um banco de dados especialista em autenticação, enquanto um serviço de vendas pode utilizar um banco de dados especialista em transações financeiras.
- Essa abordagem permite que os bancos de dados sejam pequenos e descentralizados, reduzindo a complexidade nas interfaces de comunicação. Além disso, cada serviço pode ter sua própria base de dados, o que proporciona maior privacidade e segurança nos dados e permite uma gestão

de conectividade mais eficiente. Essa abordagem de banco de dados especialista em microsserviços também permite escalabilidade e disponibilidade independentes para cada serviço.

- Por exemplo, um serviço de gerenciamento de estoque pode precisar de mais capacidade de processamento durante a alta temporada, enquanto um serviço de atendimento ao cliente pode precisar de mais disponibilidade durante o horário comercial.
- Cada serviço pode ser escalado de forma independente, sem afetar outros serviços ou a infraestrutura como um todo.

P) Redução de custo

- O uso de microsserviços em ambientes de nuvem pode trazer benefícios significativos para as empresas, incluindo a redução de custos de infraestrutura. De acordo com o estudo *Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures*, conduzido por Villamizar et al. (2017), as empresas podem obter uma redução média de até 77,08% nos custos de infraestrutura ao adotar microsserviços em nuvem. Isso acontece porque a escalabilidade horizontal e a disponibilidade dos microsserviços permitem que as empresas paguem apenas pelos recursos que estão sendo utilizados em tempo real, o que pode gerar uma economia significativa em relação à infraestrutura tradicional.
- Além disso, a implantação de microsserviços em contêineres torna mais fácil e rápida a manutenção e a atualização do sistema, reduzindo os custos com tempo de inatividade e erros.
- Dessa forma, o uso de microsserviços em nuvem pode ser uma solução viável para empresas que buscam reduzir seus custos e aumentar a eficiência do sistema.

Q) Escalar sob demanda

- Permite escalar rapidamente e sob demanda, aumentando a disponibilidade e diminuindo o tempo médio de resposta do sistema. Distribuindo a carga entre os serviços disponíveis, é possível alcançar maior confiabilidade por meio de redundância. Além disso, o uso de balanceadores

de carga pode aumentar a probabilidade de receber uma resposta bem-sucedida.

- A capacidade de escalar sob demanda torna essa arquitetura uma solução atraente para empresas que precisam lidar com picos de tráfego e demanda, permitindo adicionar ou remover instâncias de serviços rapidamente sem comprometer a disponibilidade ou a qualidade do sistema.

R) Equipe mais produtiva

- É possível reduzir a dívida técnica das equipes a longo prazo, tornando-as mais produtivas. Isso porque a modularização do sistema permite que cada equipe se concentre em um único serviço, tornando mais fácil e ágil a implementação de novas funcionalidades e correções de erros.
- Dessa forma, as equipes podem trabalhar de forma mais independente e eficiente, sem a necessidade de lidar com a complexidade e dependências de um sistema monolítico.
- Isso resulta em maior agilidade e velocidade de entrega de valor para o negócio.

S) Vender no formato SAAS

- A arquitetura baseada em microsserviços permite que os serviços sejam autônomos, autocontidos e tenham implementação independente. Essas características possibilitam a monetização do sistema no formato SaaS, gerando uma nova fonte de receita para as empresas.
- Por exemplo, no contexto de uma financeira, um serviço de gerenciamento de empréstimos poderia ser dividido em microsserviços independentes, tais como verificação de crédito, avaliação de risco, cálculo de juros e geração de contratos. Cada microsserviço pode ser escalado de forma independente para atender à demanda, tornando o sistema altamente disponível e tolerante a falhas.
- Adicionalmente, a comercialização do sistema no formato SaaS poderia oferecer a outras instituições financeiras uma plataforma de serviços eficiente e flexível.

6.2.3. Orientação 3: Perfil dos profissionais necessários

A seguir serão descritos os perfis dos profissionais necessários para realizar a migração, conforme consta na Figura 30.

Figura 30 – Orientação 3: Perfil dos profissionais necessários

Fonte: autor.

Detalhamento dos subtópicos:

A) Experiência com sistemas distribuídos

- Para implementar uma arquitetura de microsserviços com sucesso, é importante que a equipe de desenvolvimento tenha experiência em sistemas distribuídos e esteja familiarizada com os desafios que envolvem esse tipo de solução. A equipe deve estar ciente de questões como transação, sobrecarga de rede, segurança, falha ao invocar serviços, entre outras.
- É fundamental ter um conhecimento sólido em programação orientada a serviços e arquiteturas distribuídas para desenvolver e manter microsserviços robustos e escaláveis.
- Ter uma equipe experiente e especializada nessa área pode garantir que a arquitetura de microsserviços seja implementada com sucesso, proporcionando um sistema eficiente, confiável e de alta qualidade.

B) Capacidade de planejamento

- Compreender e planejar a identificação dos microsserviços é fundamental para o sucesso de uma arquitetura baseada em microsserviços. A equipe de desenvolvimento deve ser capaz de identificar quais serviços são

necessários, como serão desenvolvidos, quais interfaces serão expostas e como serão conectados com outros microsserviços.

- É importante considerar também a capacidade de escalar cada serviço de forma independente para atender à demanda. Um planejamento cuidadoso pode garantir que os microsserviços sejam coesos, autocontidos e capazes de operar de forma independente, garantindo a escalabilidade e a tolerância a falhas do sistema.
- Além disso, o planejamento deve levar em conta a comunicação entre os microsserviços e como eles serão gerenciados no ambiente de nuvem.

C) Alta qualificação

- É essencial que a equipe envolvida na migração de sistemas para microsserviços tenha uma boa qualificação, de acordo com 95% dos especialistas consultados. Isso se deve aos enormes desafios envolvidos na identificação e extração dos serviços, bem como na implementação de questões como segurança, latência, segregação de dados, entre outras.
- Se esses desafios não forem muito bem planejados e estruturados, há grandes riscos de comprometimento do sucesso da migração.

D) Equipes autônomas

- Quando a organização adota a arquitetura de microsserviços, as equipes devem ser capazes de trabalhar de forma autônoma, tendo mais liberdade e responsabilidade com menos processos burocráticos.
- A autonomia das equipes na implantação de serviços na produção, defendida pelo manifesto ágil, torna a entrega de serviços mais ágil e eficiente.
- No entanto, é importante ressaltar que a autonomia não deve ser confundida com falta de colaboração e comunicação entre as equipes.
- Para garantir o sucesso do projeto de microsserviços, é fundamental um trabalho integrado entre as equipes de desenvolvimento e operações (DevOps) e infraestrutura.

6.2.4. Orientação 4: Por Onde Começar a migração

A seguir serão descritas as etapas por onde começar a migração. É importante destacar que elas foram aprovadas por mais de 90% dos especialistas e são mostradas na Figura 31.

Figura 31 – Orientação 4: Por Onde Começar a migração

Fonte: autor.

Detalhamento dos subtópicos:

A) 1º: Inicie pelos serviços mais fáceis de serem identificados

- Comece pelos serviços mais fáceis e óbvios de serem identificados e extraídos. Dessa forma, a equipe pode ganhar mais conhecimento sobre a arquitetura e o processo de migração, tornando mais fácil a identificação e a extração dos próximos serviços.
- Conforme a equipe ganha mais experiência, os serviços podem ser refinados e melhorados para atender às necessidades específicas da arquitetura de microsserviços.
- É importante ressaltar que esse processo de identificação e extração dos serviços deve ser feito de forma estratégica, considerando as interdependências dos serviços e evitando impactos negativos na funcionalidade do sistema como um todo.

B) 2º: Menor impacto para o negócio

- Escolha a funcionalidade que tem o menor impacto para o negócio e para a equipe, quando comparada às demais.
- Isso permite validar os limites estabelecidos entre os recursos com o menor risco de efeitos colaterais para validar o mapeamento.
- Além disso, essa estratégia permite à equipe adquirir mais conhecimento e experiência na construção de microsserviços, aumentando a eficiência e minimizando os riscos em migrações futuras.

C) 3º: Defina as métricas

- Para avaliar a efetividade da migração para uma arquitetura de microsserviços, é essencial definir e monitorar métricas que possam gerar indicadores do sistema monolítico atual.
- Entre as métricas mais utilizadas pelos especialistas, estão o Apdex, que mede a satisfação do usuário com o desempenho do sistema, a quantidade de acessos ou transações realizadas e o custo envolvido na manutenção do sistema.
- O uso dessas métricas permite avaliar a eficiência da arquitetura atual, identificar oportunidades de melhoria e garantir que a migração esteja alcançando seus objetivos de desempenho e custo.

D) 4º: Utilize abordagem gradual

- Utilize uma abordagem gradual e iterativa. Isso permitirá que a equipe possa começar com os serviços mais fáceis de serem identificados e extraídos, tendo mais conhecimento sobre a arquitetura de microsserviços, para então avançar para serviços mais complexos e refinados.
- É importante ressaltar que a migração para essa arquitetura não é uma tarefa trivial e requer uma equipe qualificada.
- Além disso, é fundamental que as métricas a serem utilizadas para avaliar a efetividade da migração sejam bem definidas e monitoradas regularmente.

E) 5º: Divida em fases

- Divida a migração em diferentes fases. Em cada fase, é possível extrair um único serviço do monolítico e criar a infraestrutura necessária para suportar a nova funcionalidade.
- Essa abordagem em fases ajuda a reduzir os riscos e a complexidade do processo de migração, além de permitir que a equipe de desenvolvimento se concentre em um serviço de cada vez.
- É importante que cada fase seja cuidadosamente planejada e executada com rigor para garantir que a nova arquitetura de microsserviços atenda aos requisitos de desempenho, segurança e escalabilidade.
- Além disso, a equipe de desenvolvimento deve garantir que a comunicação e a colaboração com outras equipes envolvidas na migração sejam mantidas em cada fase.

- Ao seguir essa abordagem iterativa, a migração para uma arquitetura de microsserviços pode ser feita com sucesso e de forma eficiente.

F) 6º: Avalie o desempenho da equipe

- Avalie o desempenho da equipe tendo em mente que o tempo de desenvolvimento de um único serviço será muito superior em relação ao desenvolvimento no monolítico (FOWLER, 2015).
- No entanto, de acordo com Fowler (2015), em sistemas menos complexos, a produtividade é maior no monolítico, mas à medida que a complexidade aumenta, a produtividade cai rapidamente.
- Com o baixo acoplamento dos microsserviços, mesmo em sistemas grandes e complexos, a produtividade permanece equilibrada, como mostrado na Figura 8.
- É importante monitorar o desempenho da equipe em cada fase da migração e garantir que eles tenham os recursos necessários para enfrentar os desafios e cumprir os prazos estabelecidos.

6.2.5. Orientação 5: DevOps (*Continuous Integration/Continuous Delivery*)

A seguir serão descritas as orientações sobre DevOps (CI/CD). Este tópico foi o mais citado durante a etapa de RSL. As orientações foram aprovadas por mais de 90% dos especialistas e são mostradas na Figura 32.

Figura 32 – Orientação 5: DevOps (*Continuous Integration/Continuous Delivery*)

Fonte: autor.

Detalhamento dos subtópicos:

A) Testes integrados

- A implementação de testes integrados pode ser uma forma eficaz de agregar qualidade às entregas e melhorar o desempenho das equipes de desenvolvimento no processo de migração para uma arquitetura de microsserviços.
- Apesar desse tema não ter sido abordado no guia devido à aplicação da Curva ABC, é importante considerar a implementação desses testes e avaliar como eles podem contribuir para a eficiência e a confiabilidade do sistema.
- Os testes integrados permitem garantir que todos os serviços funcionem corretamente em conjunto, prevenindo problemas de integração e garantindo a confiabilidade do sistema como um todo.
- É importante avaliar a viabilidade e a importância dessa prática para a equipe, a fim de aprimorar a qualidade das entregas.

B) *Continuous Integration* e *Continuous Delivery*

- É fundamental para a produtividade da equipe no contexto de microsserviços em nuvem implantar as práticas de CI e CD. Elas permitem a integração e teste contínuos do código, garantindo que as mudanças sejam integradas ao sistema de forma eficiente e eficaz.
- A CD possibilita a implantação de mudanças no sistema de forma automática e rápida, sem interrupção nos fluxos de trabalho. Isso aumenta a eficiência do processo de desenvolvimento e reduz o tempo de espera entre a implementação e a entrega.
- A implementação da CI como primeira atividade é essencial para aproveitar ao máximo a estrutura de microsserviços em nuvem e garantir uma entrega contínua eficaz. Com isso, a equipe pode garantir a qualidade do código e a segurança do sistema.
- Portanto, é importante que a equipe de desenvolvimento defina e implemente um processo de CI bem definido e adaptado às necessidades da organização e dos projetos de microsserviços em nuvem.

C) Cultura

- Embora a cultura DevOps seja recomendada para lidar com múltiplos serviços e suas implantações em microsserviços, ela pode não ser fundamental em MEs com poucas áreas envolvidas no processo de desenvolvimento e implantação.
- No entanto, é importante que a equipe tenha uma comunicação eficiente e colaborativa, com um alinhamento claro dos objetivos e das prioridades. Isso ajuda a garantir que todos estejam cientes do processo de migração para uma arquitetura de microsserviços e suas respectivas responsabilidades.
- Além disso, a equipe deve estar aberta a aprender e se adaptar às novas práticas e tecnologias, para garantir que a migração seja realizada com sucesso e eficiência.

D) Gerenciamento de configuração

- Em uma ME na qual os recursos são escassos, a automatização do gerenciamento de configuração pode ser ainda mais importante, pois permite que a equipe de desenvolvimento se concentre em tarefas mais estratégicas e menos em atividades manuais repetitivas.
- A implementação de ferramentas de gerenciamento de configuração pode ajudar a garantir a consistência e a integridade do sistema, além de permitir que a equipe de desenvolvimento possa rastrear alterações e recuperar versões anteriores do código facilmente.
- Mesmo em um ambiente com poucos recursos, é possível encontrar ferramentas de código aberto ou de baixo custo que podem ajudar na automatização do gerenciamento de configuração.
- É importante avaliar as necessidades específicas da organização e escolher as ferramentas mais adequadas para o contexto da equipe e dos projetos de microsserviços

6.2.6. Orientação 6: Identificar os Microsserviços Candidatos

Essa é uma atividade muito difícil, custosa e complexa. Durante a etapa de RSL, surgiram dois tópicos tratando do mesmo assunto. O primeiro e mais citado

durante a etapa de aplicação da Curva ABC é a abordagem DDD; e o segundo é a UML (casos de uso e diagrama de classes).

As orientações sobre DDD foram aprovadas por 89% dos especialistas, e sobre UML, por 85% dos respondentes. Essas orientações são mostradas na Figura 33.

Figura 33 – Orientação 6: Identificar os Microsserviços Candidatos

Fonte: autor.

Detalhamento dos subtópicos:

***Domain Driven Design* (DDD)**

A) Extração de serviços

- Pode ser utilizado como um método para facilitar o processo. O DDD propõe a definição de contextos que são compostos de componentes do modelo de negócio que sejam consistentes entre si e tenham baixo acoplamento.
- Essa abordagem ágil ajuda a garantir que os serviços extraídos sejam coesos, sem depender de outros serviços ou componentes do sistema monolítico.
- Dessa forma, é possível extrair serviços que possam ser facilmente integrados em uma arquitetura de microsserviços, com maior eficiência e menor risco de interrupções ou problemas de integração.

B) Análise de contexto limitado

- A análise de contexto limitado é uma prática importante para a identificação de microsserviços dentro de um sistema. Essa técnica envolve a delimitação explícita de um modelo de domínio específico, que representa um

subconjunto do domínio geral do sistema, e a análise das interações desse modelo com outras áreas do sistema.

- Ao definir um modelo de domínio limitado, é possível identificar as entidades e as funcionalidades que estão intimamente relacionadas e que podem ser separadas do restante do sistema.
- Isso permite a criação de microsserviços independentes e autônomos, que são mais fáceis de gerenciar e manter. Vale ressaltar que a análise de contexto limitado deve ser combinada com outras práticas de DDD, como a definição de *bounded contexts* e agregados, para garantir uma arquitetura coesa e escalável.

C) Domínio e subdomínio

- A identificação de funcionalidades em termos de domínios e subdomínios é fundamental para a criação de microsserviços. O domínio representa uma área específica do negócio, enquanto o subdomínio é uma subdivisão desse domínio.
- Ao analisar cada subdomínio, é possível identificar as funcionalidades que são autônomas e independentes o suficiente para serem separadas em um microsserviço próprio.
- Dessa forma, é possível criar serviços menores e mais coesos, que são mais fáceis de gerenciar e manter.
- A definição clara dos limites de cada serviço ajuda a evitar conflitos e duplicação de esforços entre as equipes de desenvolvimento, além de garantir que cada microsserviço seja responsável por uma única área do negócio.

D) Validação final

- A identificação correta dos *softwares* candidatos a microsserviços é um passo fundamental na arquitetura de microsserviços.
- É essencial mapear claramente os limites de cada subdomínio e agrupar as funcionalidades autônomas em microsserviços correspondentes para implementar os serviços com segurança e confiança.
- Caso esse trabalho não seja bem feito, pode haver uma sobreposição ou duplicação de funcionalidades entre os serviços, resultando em problemas de manutenção e escalabilidade no futuro.

- Além disso, pode haver conflitos e duplicação de esforços entre as equipes de desenvolvimento, o que pode comprometer o sucesso da arquitetura de microsserviços como um todo.
- Portanto, a identificação correta dos candidatos a microsserviços é fundamental para evitar esses problemas e alcançar uma arquitetura escalável e resiliente.

UML

A) Contextos limitados

- A utilização de diagramas de casos de uso ou diagramas de classes na linguagem UML é uma abordagem altamente eficiente para identificar microsserviços candidatos e definir contextos limitados.
- Os diagramas de casos de uso permitem identificar as funcionalidades de alto nível que o sistema deve oferecer, possibilitando a definição de limites de contexto e a criação de microsserviços que atendam a esses requisitos.
- Já os diagramas de classes permitem modelar as entidades e relacionamentos do domínio de negócio, auxiliando na identificação de funcionalidades específicas que podem ser separadas em microsserviços independentes.
- Em conjunto, essas abordagens criam uma arquitetura de microsserviços escalável, eficiente e fácil de gerenciar, contribuindo para a evolução contínua do sistema.

B) Representação gráfica

- Transformar o monolito em uma representação gráfica pode ser uma abordagem altamente eficiente para identificar os serviços candidatos utilizando UML.
- O grafo resultante, em que cada vértice representa uma classe do sistema e as arestas não direcionadas representam o acoplamento entre as classes, permite identificar os componentes que podem ser isolados e executados de forma independente.
- Essa representação gráfica é uma ferramenta valiosa para entender o sistema e identificar os pontos de acoplamento e dependência entre as

classes, contribuindo para uma migração cuidadosa e eficiente para uma arquitetura de microsserviços.

C) Diagramas

- Por meio de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML, que podem ajudar na identificação de microsserviços candidatos.
- Com esses diagramas é possível listar as funcionalidades para as quais o sistema foi projetado e identificar as classes que implementam essas funcionalidades, auxiliando na separação dos serviços em microsserviços independentes.
- Além disso, os diagramas UML podem ajudar a entender as dependências e os relacionamentos entre as classes do sistema, permitindo a identificação de pontos críticos que devem ser isolados em microsserviços independentes.
- Essa abordagem é uma ferramenta valiosa para a criação de uma arquitetura de microsserviços escalável e fácil de gerenciar, adaptável às necessidades do negócio e preparada para enfrentar os desafios da tecnologia.

6.2.7. Orientação 7: Segregação de Banco de Dados

A segregação de banco de dados é um tema polêmico, com opiniões divergentes entre os autores sobre sua característica positiva ou negativa.

Uma das principais diferenças entre as abordagens de microsserviços e monolíticas é que os monolitos estão presos a um único tipo de banco de dados, enquanto nas arquiteturas distribuídas cada aplicação pode ter o tipo de banco de dados mais adequado ao cenário, o que permite aos sistemas distribuídos utilizar soluções de bases de dados não relacionais, como NoSQL, com mais facilidade.

A implementação de microsserviços com a separação de bancos de dados traz um novo conceito de solução: o microbanco de dados. Além de pequenos e descentralizados, tanto os serviços quanto os bancos de dados oferecem três grandes benefícios: redução da complexidade nas interfaces de comunicação, maior privacidade e segurança nos dados e melhor gestão de conectividade.

Essa abordagem contribui para uma arquitetura escalável, fácil de gerenciar e adaptável às necessidades do negócio, além de oferecer um ambiente mais seguro e eficiente para lidar com dados. Com microsserviços, é possível criar um ambiente mais resiliente e preparado para as mudanças constantes do mercado, com uma arquitetura distribuída capaz de atender às demandas de diferentes cenários de negócios.

As orientações sobre segregação de banco de dados foram aprovadas por 73% dos especialistas consultados e são mostradas na Figura 34.

Figura 34 – Orientação 7: Segregação de Banco de Dados

Fonte: autor.

Detalhamento dos subtópicos:

Vantagens

A) Escalabilidade

- Ao utilizar um microbanco de dados para cada serviço, é possível dimensionar cada serviço separadamente, o que permite aumentar a capacidade do banco de dados de um serviço específico sem afetar o desempenho dos outros serviços.
- Além disso, a segregação de banco de dados permite que cada serviço tenha seu próprio esquema de banco de dados, que pode ser otimizado para atender às necessidades específicas do serviço, aumentando ainda mais a escalabilidade e o desempenho do sistema como um todo (SATHISHKUMAR; KAVITHA, 2020).

B) Flexibilidade

- Com microbancos de dados, os desenvolvedores têm mais liberdade para escolher o tipo de banco de dados que melhor se adapta às necessidades de cada serviço. Por exemplo, um serviço pode precisar de um banco de dados relacional, enquanto outro pode se beneficiar mais de um banco de dados NoSQL.
- Essa flexibilidade permite que os desenvolvedores escolham a melhor tecnologia de banco de dados para cada serviço, em vez de ficarem limitados a um único tipo de banco de dados. Além disso, essa abordagem simplifica a manutenção e o gerenciamento dos bancos de dados, tornando o processo mais eficiente e adaptável às necessidades do negócio.
- Essa escolha cuidadosa de tecnologias de bancos de dados é importante para melhorar o desempenho e reduzir os custos do sistema (NARAYANAN; VITHANI, 2020).

C) Independência

- Ao utilizar um microbanco de dados para cada serviço, é mais fácil manter a independência entre os serviços. Isso significa que, se houver um problema com um banco de dados específico, ele não afetará outros serviços.
- Essa independência também permite que os desenvolvedores alterem o esquema de banco de dados de um serviço sem afetar outros serviços, o que é especialmente útil em sistemas complexos com muitos serviços interdependentes.
- Com microbancos de dados, cada serviço pode evoluir independentemente, facilitando a implementação de novos recursos e melhorias no sistema como um todo (NAMIOT; SNEPS-SNEPPE, 2020).

D) Facilidade de gerenciamento

- Gerenciar vários microbancos de dados pode ser mais fácil do que gerenciar um único banco de dados grande que é usado por todos os serviços.
- Isso ocorre porque os microbancos de dados são menores e mais simples de gerenciar, e é possível automatizar tarefas de gerenciamento, como *backups* e restaurações.

- Além disso, com microbancos de dados, é possível ter diferentes níveis de segurança e acesso para cada serviço, o que simplifica a gestão de privilégios de usuário e garante maior segurança dos dados.
- Essa abordagem também torna mais fácil a implementação de políticas de governança de dados e de conformidade regulatória, já que cada serviço pode ter seu próprio conjunto de regras e diretrizes específicas (MAZHAR; REHMANI; REHMANI, 2021).

E) Performance

- Com microbancos de dados, é possível otimizar o desempenho de cada serviço individualmente, em vez de ter que otimizar um único banco de dados grande para atender a todos os serviços.
- Isso pode resultar em tempos de resposta mais rápidos e melhor desempenho geral do sistema. Além disso, como os microbancos de dados são menores, as consultas e as transações são processadas mais rapidamente, o que pode reduzir a latência e melhorar a experiência do usuário.
- A escalabilidade horizontal também é facilitada pela utilização de microbancos de dados, uma vez que novos bancos de dados podem ser facilmente adicionados para suportar o aumento de tráfego em serviços específicos.
- Isso permite que o sistema seja dimensionado de forma mais eficiente e flexível, de acordo com as necessidades do negócio (YU et al., 2019).

Desvantagens

A) Sobrecarga de rede

- A segregação de banco de dados em microsserviços pode introduzir sobrecarga de rede, pois cada serviço precisa fazer chamadas externas para acessar os dados necessários. Isso pode resultar em latência adicional e sobrecarga na rede, o que pode afetar negativamente o desempenho do sistema como um todo.
- Além disso, a complexidade das interfaces de comunicação entre os serviços pode aumentar, o que pode dificultar o desenvolvimento e a manutenção do sistema (GRAVANIS; KAKARONTZAS; GEROGIANNIS, 2021).

B) Inconsistência de dados

- A segregação de banco de dados em microsserviços pode levar a inconsistências de dados, especialmente em contextos de transações distribuídas.
- Quando os dados são divididos em bancos de dados separados, pode ser mais difícil manter a consistência entre eles, o que pode resultar em erros e problemas de integridade de dados.
- Isso pode ser especialmente problemático em sistemas que precisam manter um alto nível de consistência entre os dados, como sistemas financeiros ou de processamento de transações (KAZANAVIČIUS; MAŽEIKA, 2020).

C) Contornar os problemas

- Para resolver possíveis inconsistências de dados causadas pela segregação de bancos de dados, é necessário implementar procedimentos específicos para garantir a integridade dos dados.
- No entanto, esses procedimentos podem afetar o desempenho do aplicativo.
- É importante considerar cuidadosamente a implementação desses procedimentos para minimizar o impacto no desempenho do sistema (MISHRA; KUNDE; NAMBIAR, 2018).

D) Complexidade

- É necessário gerenciar múltiplos bancos de dados, cada um com suas próprias características e requisitos, tornando o sistema mais complexo. Isso pode aumentar a carga de trabalho da equipe de desenvolvimento e dificultar a manutenção do sistema.
- Além disso, a complexidade pode aumentar ainda mais se os dados precisarem ser compartilhados entre serviços, exigindo a implementação de mecanismos de sincronização e comunicação entre os bancos de dados (LIU et al., 2021).

E) Custo

- A segregação de banco de dados em microsserviços pode aumentar o custo do sistema, uma vez que é necessário contratar um novo recurso para

cada banco de dados, além de configurar e gerenciar múltiplos bancos de dados em vez de um único banco de dados grande.

- Esse aumento de custo pode ser um fator limitante para a adoção da arquitetura de microsserviços em algumas empresas, principalmente as menores, que podem não ter recursos suficientes para suportar essa abordagem.
- No entanto, é importante avaliar cuidadosamente os custos e os benefícios antes de tomar uma decisão, pois, embora possa haver um aumento nos custos iniciais, a arquitetura de microsserviços pode trazer benefícios significativos a longo prazo, como maior flexibilidade, escalabilidade e desempenho (CLÉMENT, 2020).

6.2.8. Orientação 8: Infraestrutura Mínima para Utilizar Microsserviços

Essa é a infraestrutura mínima recomendada para suportar operações com microsserviços em nuvem, de acordo com a aprovação de 84% dos especialistas consultados. As orientações a seguir, mostradas na Figura 35, podem ser seguidas para garantir a efetividade da arquitetura de microsserviços em nuvem.

Figura 35 – Orientação 8: Infraestrutura Mínima para Utilizar Microsserviços

Fonte: autor.

Detalhamento dos subtópicos:

A) Circuit Breaker

- O Circuit Breaker é uma ferramenta utilizada em arquiteturas de microsserviços para monitorar a comunicação entre os serviços e detectar falhas.

- Ele é responsável por interromper a comunicação com um serviço que está com problemas, evitando que o erro se propague para outros serviços e aumente a carga no sistema. Quando há falhas suficientes, o Circuit Breaker desliga a conexão com o serviço com problemas e retorna um erro para o cliente em vez de continuar a enviar solicitações que podem agravar o problema.
- Exemplos de ferramentas de Circuit Breaker incluem o Hystrix da Netflix, o Istio da Google, o Polly da AWS, o Resilience4j e o Circuit Breaker da Spring Cloud. Na Amazon Web Services (AWS), o serviço Elastic Load Balancer (ELB) inclui recursos de Circuit Breaker para ajudar a gerenciar a comunicação entre serviços na nuvem.
- Na Microsoft Azure, o serviço Azure Service Fabric inclui um mecanismo de Circuit Breaker para ajudar a detectar e gerenciar falhas em serviços de microsserviços.

B) API Gateway

- O API Gateway é uma peça fundamental na arquitetura de microsserviços, atuando como ponto de entrada para todas as chamadas dos clientes e redirecionando as solicitações para o microsserviço correto.
- Grandes provedores de nuvem suportam o Gateway de API, como o Amazon API Gateway na AWS, o Azure API Management na plataforma da Microsoft Azure e o Google Apigee da Google.
- Essas ferramentas oferecem recursos como balanceamento de carga, autenticação, autorização, transformação de mensagens e monitoramento para APIs em larga escala, facilitando a criação, a publicação, a manutenção e a proteção de APIs.
- Com essas soluções, as organizações podem implementar um API Gateway robusto e escalável em suas arquiteturas de microsserviços.

C) Docker e Kubernetes

- O Docker é uma plataforma de conteinerização que oferece diversos benefícios para a gestão de infraestrutura de uma empresa, incluindo a portabilidade e o isolamento do ambiente.
- Além disso, o Kubernetes, uma tecnologia para *containers*, otimiza a gestão de espaço de dados e a velocidade de processos.

- Essas ferramentas são particularmente relevantes para micro e pequenas empresas, que, muitas vezes, precisam lidar com recursos limitados e demandas crescentes. Entre os principais benefícios do Docker e do Kubernetes, destacam-se:
  - Orquestração: com o Docker e o Kubernetes, é possível desenvolver e indicar o armazenamento de dados de acordo com as necessidades da empresa, utilizando armazenamentos locais e estratégias em múltiplos fornecedores de nuvem (*multi-cloud*). Isso permite maior flexibilidade e reduz a dependência de um único fornecedor.
  - Otimização dos recursos: é possível distribuir a memória conforme a necessidade, utilizando os *clusters* de nós para executar tarefas nos *containers*, o que garante uma utilização mais eficiente dos recursos disponíveis. Isso é especialmente importante para MEs que precisam lidar com recursos limitados.
  - Autocorreção: em caso de falha de algum *container*, o Kubernetes o reinicia e, em caso de erro, substitui por outro, garantindo a continuidade do serviço e reduzindo a necessidade de intervenção humana. Isso permite que a equipe de TI da ME possa se concentrar em outras tarefas importantes.
  - Automatização: com o Docker e o Kubernetes, é possível programar a criação de *containers* automaticamente na implantação, além de eliminar *containers* ou concentrar todos os recursos no novo *container*. Isso garante maior eficiência no processo de implantação de aplicações, o que é especialmente relevante para MEs que precisam lidar com limitações de recursos.
  - Segurança da informação: o Docker e o Kubernetes permitem o armazenamento e o gerenciamento de informações sigilosas, utilizando senhas, tokens e chaves, o que aumenta a segurança e mantém determinado *container* em segredo. Isso é importante para MEs que precisam proteger as suas informações confidenciais.

Embora o tema “Kubernetes” tenha ficado fora do guia devido ao corte estabelecido pela Curva ABC, entende-se que essa implementação é importante a médio e longo prazo. A utilização do Docker e do Kubernetes

traz benefícios significativos para a gestão de infraestrutura de empresas, garantindo mais eficiência, segurança e escalabilidade para suas aplicações e serviços, e é especialmente relevante para MEs que precisam lidar com recursos limitados e demandas crescentes.

D) *Service Discovery*

- O *Service Discovery* é uma solução importante para empresas que utilizam a arquitetura de microsserviços em ambientes de nuvem, pois permite o controle dos serviços implantados e seus endereços exatos e números de porta, evitando problemas de escalabilidade e desempenho.
- Grandes fornecedores de nuvem, como a Amazon Web Services, a Microsoft Azure e o Google Cloud Platform, fornecem soluções de *Service Discovery*, como o Amazon Route 53, a Azure Service Fabric e o Google Cloud Endpoints, para um gerenciamento eficiente dos microsserviços em uma arquitetura.

A arquitetura de microsserviços comumente tem diversos serviços em execução simultaneamente, o que pode gerar problemas de comunicação entre eles caso não seja possível manter o controle dos serviços implantados e seus endereços e números de porta.

O *Service Discovery* permite que as aplicações descubram e se conectem com os serviços corretos em tempo real, sem a necessidade de intervenção manual, lidando com a dinamicidade da infraestrutura em nuvem.

As soluções de *Service Discovery* disponibilizadas pelos grandes fornecedores de nuvem permitem que as empresas tenham um gerenciamento mais eficiente dos microsserviços em suas arquiteturas em nuvem.

Além disso, há melhora na eficiência da comunicação entre os serviços, permitindo que as aplicações descubram os serviços disponíveis e se conectem com eles de maneira rápida e eficiente.

Essa melhoria evita problemas de desempenho e garante a alta disponibilidade das aplicações.

Sendo assim, é fundamental que as empresas implantem o *Service Discovery* em suas infraestruturas de microsserviços em nuvem para garantir uma comunicação eficiente e escalável entre seus serviços.

7. CONCLUSÕES, CONTRIBUIÇÕES, LIMITAÇÕES E ESTUDOS FUTUROS

Esta dissertação teve como objetivo analisar a literatura existente sobre a migração da arquitetura monolítica para a arquitetura de microsserviços na Computação em Nuvem, focando nas MEs de *software*. A intenção foi identificar problemas e oportunidades de melhorias na implementação dessa arquitetura, fornecendo conhecimento útil para as microempresas brasileiras.

Para isso, foi realizada uma RSL, permitindo a busca e a análise de trabalhos relevantes em bases científicas. Essa abordagem possibilitou identificar quais práticas são realmente relevantes na implementação dessa arquitetura pelas empresas de mercado. Após a seleção e a classificação das práticas apontadas na literatura, utilizando-se o método Curva ABC (Quadro 10), foi elaborado um guia com orientações que darão suporte às microempresas de *software* na migração de sistemas com arquitetura monolítica para a arquitetura de microsserviços em nuvem. Esse guia oferecerá às MEs brasileiras um suporte valioso nesse processo de transição, que busca soluções de arquiteturas de *softwares* modernas e eficientes.

Para tanto, as orientações apontadas no guia desenvolvido, a partir de uma pesquisa de campo com o uso de questionário, foram submetidas para avaliação junto a especialistas em desenvolvimento de *software* do mercado brasileiro (especialistas com vasta experiência em implementações de microsserviços em nuvem). Portanto, o guia desenvolvido nesta pesquisa pode tornar-se uma ferramenta fundamental para o avanço tecnológico e para o estado da arte em *softwares* com o uso de Computação em Nuvem no mercado brasileiro.

A realização deste projeto de pesquisa permitiu ao pesquisador maior compreensão acerca dos diversos estudos existentes sobre a migração de sistemas de informação do ambiente monolítico para a arquitetura de microsserviços no contexto do desenvolvimento de *software*. No entanto, a maioria desses estudos concentra-se nas grandes corporações, deixando uma lacuna no que diz respeito às MEs, que enfrentam desafios particulares relacionados à limitação de recursos financeiros e humanos em seu dia a dia no mercado.

Nesse sentido, é plausível considerar que a criação desse guia, voltado especificamente às MEs de *software*, que oferece um conjunto de orientações relevantes e detalhadas no processo de migração da arquitetura monolítica para a arquitetura de microsserviços na Computação em Nuvem, poderá ser de grande

importância e valor para o mercado brasileiro, apoiando as empresas a enfrentarem seus desafios de forma mais eficiente, promovendo melhorias na qualidade, na escalabilidade e no desempenho dos serviços prestados.

Além disso, a implementação bem-sucedida de uma arquitetura de microsserviços em MEs de *software* pode trazer benefícios significativos, como maior flexibilidade, facilidade na manutenção e aprimoramento na velocidade de entrega de novas funcionalidades exigidas por seus clientes. Essas melhorias, por sua vez, podem contribuir para a competitividade e o crescimento dessas organizações, permitindo que se destaquem em um cenário cada vez mais competitivo e inovador.

Acredita-se que este projeto de pesquisa tem o potencial de causar um impacto significativo no setor, por meio do compartilhamento de melhores práticas e da promoção de uma maior conscientização sobre as vantagens e as desvantagens da migração dos sistemas de informação tradicional para a arquitetura de microsserviços na Computação em Nuvem. Este estudo pode servir como um recurso valioso para as MEs brasileiras e, possivelmente, para outras empresas de pequeno porte ao redor do mundo.

Em resumo, este trabalho busca preencher uma lacuna importante na literatura, fornecendo diretrizes e recomendações específicas para MEs de *software* que trabalham com o tema no mercado ou que desejam migrar seus sistemas de informação do ambiente monolítico para a arquitetura de microsserviços na Computação em Nuvem.

Acredita-se que a aplicação do guia de recomendações avaliado e aprovado por especialistas do mercado possa trazer benefícios às MEs, uma vez que a transição de uma arquitetura monolítica para uma arquitetura de microsserviços em nuvem pode ser um processo desafiador, devido principalmente às suas características e complexidades apresentadas ao longo deste estudo.

Espera-se que esse guia seja amplamente difundido, tanto no meio acadêmico quanto no mercado carente desse tipo de conhecimento. Dado aos desafios tecnológicos atuais, é fundamental que as empresas estejam preparadas e atualizadas, e a aplicação das orientações desse guia pode trazer benefícios reais, assim como já foi obtido pelas grandes organizações de TI. É hora de aproveitar as oportunidades que a tecnologia nos oferece, e esse guia é uma ferramenta essencial para isso.

7.1. CONTRIBUIÇÕES PARA A ÁREA

A contribuição desta pesquisa pode ser observada em três perspectivas distintas.

A primeira perspectiva consiste em uma contribuição imediata para a academia, expandindo a teoria sobre a migração de *software* monolítico para a arquitetura de microsserviços em nuvem no contexto das MEs de *software*. A pesquisa realizada até o momento não identificou nenhum artigo que abordasse esse tema aplicado especificamente às MEs, o que possibilita a ampliação desse campo de estudo, enfatizando a relevância dessas organizações para a economia e abordando a lacuna existente nas publicações científicas.

A segunda perspectiva é fornecer uma RSL que permitiu explorar o estado da arte sobre o tema da migração de sistemas com arquitetura monolítica para arquitetura de microsserviços em nuvem. Esse recorte específico pode ser comparado com outros estudos e servir de inspiração para que pesquisadores busquem soluções adequadas às realidades das MEs.

A terceira perspectiva é o desenvolvimento do instrumento Guia de Orientações, que pode ser utilizado por empresas durante o processo de migração de seus sistemas com arquitetura monolítica para microsserviços em nuvem. Esse guia inclui todas as referências e fundamentação teórica necessárias para apoiar as MEs na identificação e na avaliação de sua preparação para lidar com essa nova arquitetura.

O conjunto de orientações constantes no guia desenvolvido, que foram validadas por especialistas, permite que a ME compare a realidade de sua organização com os riscos que podem ser mitigados. Essa contribuição oferece aos gestores e equipes de desenvolvimento uma visão abrangente das etapas a serem implementadas para reduzir os riscos na migração de seus sistemas.

7.2. LIMITAÇÕES DA PESQUISA

Neste estudo, é importante considerar algumas limitações significativas e explícitas relacionadas à análise das MEs de *software*. Uma delas é a complexidade em lidar com diferentes arquiteturas de *software*, o que pode ser um obstáculo substancial na implementação das práticas propostas. Essa limitação exige que as

práticas sugeridas sejam cuidadosamente adaptadas ao contexto específico de cada microempresa, levando em conta as particularidades e desafios enfrentados por elas.

Outro aspecto a ser considerado é o risco de interpretações equivocadas das orientações apresentadas no guia, o que pode resultar em implementações inconsistentes e potencialmente prejudiciais para as MEs. Além disso, a confiabilidade dos dados coletados deve ser meticulosamente avaliada, pois a qualidade das informações obtidas é crucial para a aplicação efetiva das práticas sugeridas.

Uma limitação importante e evidente deste estudo é a falta de pesquisas anteriores voltadas especificamente para a migração de arquitetura monolítica para arquitetura de microsserviços em nuvem no contexto das MEs de *software*. Essa lacuna na literatura representa um desafio significativo para a pesquisa e requer a busca por literaturas correlatas e a análise de conceitos e práticas relevantes em contextos semelhantes.

Essas limitações devem ser levadas em consideração ao avaliar os resultados e as contribuições desta pesquisa. Apesar dessas restrições, este estudo ainda pode oferecer *insights* valiosos e servir como ponto de partida para futuras investigações sobre a migração de arquiteturas monolíticas para arquiteturas de microsserviços em nuvem, principalmente no contexto das MEs de *software*.

7.3. TRABALHOS FUTUROS

Propõe-se uma pesquisa que mostre uma implementação e o uso efetivo do guia por MEs do mercado brasileiro e que seja avaliada a qualidade do produto de *software* implementado com microsserviços, bem como a satisfação dos clientes e a diminuição do custo no projeto. Essa mensuração, bem como métricas, podem ser desenvolvidas para que uma quantificação possa ser associada aos ganhos quando da implementação do guia desenvolvido.

Acredita-se que fatores motivacionais e relações interpessoais entre os times de projetos das MEs envolvidas no processo de migração utilizando o guia de recomendações influenciarão na evolução dele. Um estudo envolvendo aspectos emocionais dos profissionais envolvidos no uso desse guia, contemplando as recomendações propostas, também poderia ser realizado.

Outros benefícios identificados que podem ser contemplados em outros trabalhos são:

- Redução de custos em projetos de migração de *software* de ambiente monolítico para ambiente de microsserviços;
- Menos retrabalho e, consequentemente, aumento da eficiência e da competitividade no uso dos sistemas migrados;
- Facilidades na implementação do guia de orientações em MEs do mercado brasileiro;
- Migração voltada para a inovação.

REFERÊNCIAS

ACEVEDO, Cesar Augusto Jaramillo; GÓMEZ Y JORGE, Juan Pablo; PATIÑO, Iván Ríos. Methodology to transform a monolithic software into a microservice architecture. *In*: **2017 6TH INTERNATIONAL CONFERENCE ON SOFTWARE PROCESS IMPROVEMENT (CIMPS)**, 2017, Zacatecas, México, p. 1-6.

ALVARENGA, Antonio Carlos; NOVAES, Antonio Galvão N. **Logística aplicada:** suprimento e distribuição física. 3. ed. São Paulo: Edgar Blücher Ltda., 2000.

AMAZON WEB SERVICES (AWS). O que é uma API? **AWS**, 2023. Disponível em: https://aws.amazon.com/pt/what-is/api. Acesso em: 09 mar. 2023.

_____. O que são microsserviços? **AWS**, 2022. Disponível em: https://aws.amazon.com/pt/microservices. Acesso em: 02 maio 2022.

_____. Prior Version(s) of Amazon EC2 Service Level Agreement - Not Currently In Effect. **AWS**, 2018. Disponível em: https://aws.amazon.com/ec2/sla/historical. Acesso em: 30 mar. 2022.

ARMBRUST, Michael et al. A view of cloud computing. **Communications of the ACM**, v. 53, n. 4, p. 50-58, 2010.

ASSOCIAÇÃO BRASILEIRA DAS EMPRESAS DE SOFTWARE (ABES). Dados do setor. **ABES**, 2023. Disponível em: https://abes.com.br/dados-do-setor/. Acesso em: 10 abr. 2023.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS (ABNT). **ISO/IEC 25000. Engenharia de software – Qualidade de produto**. Associação Brasileira de Normas Técnicas, 2005.

AUER, Florian et al. From monolithic systems to Microservices: An assessment framework. **Information and Software Technology**, v. 137, 106600, 2021.

BAJAJ, D. et al. Partial migration for re-architecting a cloud native monolithic application into microservices and FaaS. *In*: BADICA, C.; LIATSIS, P.; KHARB, L.; CHAHAL, D. (eds). Information, Communication and Computing Technology. ICICCT 2020. **Communications in Computer and Information Science**, v. 1170, p. 111-124, 2020.

BALALAIE, Armin; HEYDARNOORI, Abbas; JAMSHIDI, Pooyan. Migrating to Cloud-Native Architectures Using Microservices: an Experience Report. *In*: CELESTI, A.; LEITNER, P. (eds). Advances in Service-Oriented and Cloud Computing. ESOCC 2015. **Communications in Computer and Information Science**, v. 567, p. 201-215, 2016.

BALLOU, Ronald H. **Logística Empresarial:** Transporte, Administração de Materiais e Distribuição Física. São Paulo: Atlas, 1993.

BALZAMOV, A. V. et al. Development of a methodology for migration of monolithic systems to microservice architecture using cloud technologies. **Journal of Physics: Conference Series**, v. 2091, n. 1, 012036, 2021.

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. **Software Architecture in Practice**. 3. ed. Boston: Addison-Wesley, 2012.

BLINOWSKI, Grzegorz; OJDOWSKA, Anna; PRZYBYLEK, Adam. Monolithic vs. Microservice Architecture: a performance and scalability evaluation. **Ieee Access**, v. 10, p. 20357-20374, 2022.

BOWLING, Ann. **Measuring health:** a review of quality-of-life measurement scales. Milton Keynes: Open University Press, 1997.

BRITO, Miguel; CUNHA, Jácome; SARAIVA, João. Identification of microservices from monolithic applications through topic modelling. *In*: **SAC' 21: Proceedings of the 36th Annual ACM Symposium on Applied Computing**, p. 1409-1418, 2021.

CAMUNDA. The State of Microservices 2021 Report. **Camunda**, 2021. Disponível em: https://camunda.com/learn/reports/microservices-report-2021/. Acesso em: 05 fev. 2023.

CAZORLA, Irene Maurício; SANTANA, Eurivalda Ribeiro dos Santos; UTSUMI, Miriam Cardoso. O campo conceitual da média aritmética: uma primeira aproximação conceitual. **Revista Eletrônica de Educação Matemática**, v. 14, p. 1-21, 2019.

CLÉMENT, A. Microservices: The Good, the bad, and the ugly. **Journal of Object Technology**, 2020.

CLEMENTS, Paul et al. Documenting software architectures: views and beyond. *In*: **INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING**, 25., 2003. IEEE, 2003. p. 740-741.

COSTA, Marco Antonio F. da; COSTA, Maria de Fátima Barrozo da. **Projeto de Pesquisa:** entenda e faça. 5. ed. Rio de Janeiro: Ed. Vozes, 2014.

COULOURIS, George et al. **Distributed Systems:** Concepts and Design. Boston: Addison-Wesley, 2001.

CREAZZA, Alessandro et al. Who cares? Supply chain managers' perceptions regarding cyber supply chain risk management in the digital transformation era. **Supply Chain Management: An International Journal**, v. 27, n. 1, p. 30-53, 2021.

DALKEY, N.; HELMER, O. Delphi Technique: characteristics and sequence model to the use of experts. **Management Science**, v. 9, n. 3, p. 458-467, 1963.

DOMASCHKA, Jörg; HAUSER, Cristopher B.; ERB, Benjamin. Reliability and Availability Properties of Distributed Database Systems. *In:* **IEEE 18TH INTERNATIONAL ENTERPRISE DISTRIBUTED OBJECT COMPUTING CONFERENCE**, 2014, Ulm, Alemanha, p. 226-233.

DORRELL, David. **Docker Deep Dive**. Apress, 2017.

DRAGAN, D.; BARNES, D. Service Oriented Architecture vs. Microservices: A Comparative Analysis. *In:* **2019 IEEE 5TH INTERNATIONAL CONFERENCE ON COLLABORATION AND INTERNET COMPUTING (CIC)**, 2019.

DRAGONI, Nicola et al. Microservices: yesterday, today, and tomorrow. **Present and Ulterior Software Engineering**, p.195-216, 2017.

DRAGONI, N.; SILLITTI, A.; SUCCI, G. Microservices in the Cloud: An Overview of Challenges and Solutions. *In:* **PROCEEDINGS OF THE 4TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING TECHNOLOGIES AND APPLICATIONS**, 2020.

ESSA, M. et al. Service Oriented Architecture and Microservices Architecture: A Comparative Study. *In:* **INTERNATIONAL CONFERENCE ON COMPUTER, CONTROL, ELECTRICAL, AND ELECTRONICS ENGINEERING (ICCCEEE)**, 2019.

FAN, Chen-Yuan; MA, Shang-Pin. Migrating monolithic mobile application to microservice architecture: An experiment report. *In:* **2017 IEEE INTERNATIONAL CONFERENCE ON AI & MOBILE SERVICES (AIMS)**, 2017, Honolulu, HI, Estados Unidos, p. 109-112.

FIORENTINI, Dario; LORENZATO, Sergio. **Investigação em Educação Matemática:** percursos teóricos e metodológicos. Campinas: Autores Associados, 2006.

FLEXERA. The State of the Cloud 2021 Report. **Flexera**, 2021. Disponível em: https://info.flexera.com/SLO-CLOUD-REPORT-Global-Landing-Page.html. Acesso em: 05 fev. 2023.

FORRESTER RESEARCH. **Predictions 2020:** Microservices and Serverless Computing will Drive the Next Wave of Digital Business Transformation. 2019.

FORZA, Cipriano. Survey research in operations management: a process-based perspective. **International Journal of Operations & Production Management**, v. 22, n. 2, p. 152-194, 2002.

FOWLER, Martin. MicroservicePremium. **Martin Fowler***,* 2015. Disponível em: https://Martinfowler.com/bliki/MicroservicePremium.html. Acesso em: 17 maio 2022.

FREIRE, Augusto Flávio A. A. et al. Migrating production monolithic systems to microservices using aspect oriented programming. **Software: Practice and Experience**, v. 51, n. 6, p. 1280-1307, 2021.

FREITAS, E.; YAMASHITA, L. Adopting microservices architecture in a Brazilian software SME: a case study. **Journal of Software Engineering Research and Development**, 2017.

FRITZSCH, J. et al. From monolith to microservices: A classification of refactoring approaches. *In:* **SOFTWARE ENGINEERING ASPECTS OF CONTINUOUS DEVELOPMENT AND NEW PARADIGMS OF SOFTWARE PRODUCTION AND DEPLOYMENT: FIRST INTERNATIONAL WORKSHOP, DEVOPS 2018**, Chateau de Villebrumier, France, March 5-6, 2018, Revised Selected Papers 1 (p. 128-141). Springer International Publishing, 2019.

FURDA, Andrei et al. Migrating enterprise legacy source code to microservices: on multitenancy, statefulness, and data consistency. **IEEE Software**, v. 35, n. 3, p. 63-72, 2018.

GARRIGA, M. et al. Towards microservices security trade-offs: Lessons learned. *In*: **IEEE INTERNATIONAL CONFERENCE ON SOFTWARE ARCHITECTURE WORKSHOPS (ICSAW)**, 2018.

GARTNER. Gartner Forecasts Worldwide Public Cloud End-User Spending to Reach Nearly $500 Billion in 2022. **Gartner**, 2022. Disponível em: https://www.gartner.com/en/newsroom/press-releases/2022-04-19-gartner-forecasts-worldwide-public-cloud-end-user-spending-to-reach-nearly-500-billion-in-2022. Acesso em: 21 maio 2022.

_____. Top 10 Strategic Technology Trends for 2021. **Gartner**, 2021. Disponível em: https://www.gartner.com/smarterwithgartner/top-10-strategic-technology-trends-for-2021/. Acesso em: 10 dez. 2022.

GRAND VIEW RESEARCH. Microservices Market Size, Share & Trends Analysis Report By Component (Platform, Services), By Deployment, By Organization, By Application, By End Use, By Region, And Segment Forecasts, 2021-2028. **Grand View Research**, 2021. Disponível em: https://www.grandviewresearch.com/industry-analysis/microservices-market. Acesso em: 05 fev. 2023.

GRAVANIS, Dimitrios; KAKARONTZAS, George; GEROGIANNIS, Vassilis. You don't need a Microservices Architecture (yet) Monoliths may do the trick. *In:* **ESSE '21: Proceedings of the 2021 European Symposium on Software Engineering**, p. 39-44, 2021.

HAJ-YOUSSEF, M. et al. Microservices Orchestration Frameworks: A Survey. **IEEE Access**, 2020.

HARPER, Eric et al. Microdatabases for the Industrial Internet. **Arxiv**, v. 3, n. 2, p. 33-43, 2016.

HASSAN, S. et al. A systematic mapping study of microservices governance. **Journal of Systems and Software**, 2021.

HAUGELAND, Sindre Grønstøl et al. Migrating Monoliths to Microservices-based Customizable Multi-tenant Cloud-native Apps. *In:* **2021 47TH EUROMICRO CONFERENCE ON SOFTWARE ENGINEERING AND ADVANCED APPLICATIONS (SEAA)**, 2021, Palermo, Itália, p. 170-177.

HEESCH, Uwe Van et al. Decision-Centric Architecture Reviews. **IEEE Software**, v. 31, n. 1, p. 69-76, 2014.

HISRICH, Robert D.; PETERS, Michael P.; SHEPHERD, Dean A. **Entrepreneurship**. McGraw-Hill Education, 2019.

HUANG, C. et al. Rapid software delivery: a case study on using DevOps to deliver software in a startup. **Journal of Systems and Software**, 2021.

JING, Jin; HELAL, Abdelsalam Sumi; ELMAGARMID, Ahmed. Client-server computing in mobile environments. **Acm Computing Surveys**, v. 31, n. 2, p. 117-157, 1999.

KALSKE, Miika; MÄKITALO, Niko; MIKKONEN, Tommi. Challenges When Moving from Monolith to Microservice Architecture. *In:* GARRIGÓS, I.; WIMMER, M. (eds). **Current Trends in Web Engineering**. 2018. p. 32-47.

KANE, Sean P.; MATTHIAS, Karl. **Docker**: Up & Running: Shipping Reliable Containers in Production. O'Reilly Media, Inc., 2018.

KAZANAVIČIUS, Justas; MAŽEIKA, Dailus. Analysis of legacy monolithic software decomposition into microservices. *In:* **CONFERENCE FORUM AND DOCTORAL CONSORTIUM CO-LOCATED WITH THE 14TH INTERNATIONAL BALTIC CONFERENCE ON DATABASES AND INFORMATION SYSTEMS**, 2020, p. 25-32.

KHALIQUE, Muhammad et al. Intellectual capital and organisational performance in Malaysian knowledge-intensive SMEs. **International Journal of Learning and Intellectual Capital**, v. 15, n. 1, p. 20-36, 2018.

LAURETIS, Lorenzo De. From monolithic architecture to microservices architecture. *In:* **2019 IEEE INTERNATIONAL SYMPOSIUM ON SOFTWARE RELIABILITY ENGINEERING WORKSHOPS (ISSREW)**, 2019, Berlim, Alemanha, p. 93-96.

LI, Yan et al. Granularity Decision of Microservice Splitting in View of Maintainability and Its Innovation Effect in Government Data Sharing. **Discrete Dynamics in Nature and Society**, v. 2020, n. 2, p. 1-11, 2020.

LIAO, Yongxin et al. Past, present and future of Industry 4.0: a systematic literature review and research agenda proposal. **International Journal of Production Research**, v. 55, n. 12, 2017.

LIU, C. et al. The design and implementation of microservice-based online education platform. **Journal of Ambient Intelligence and Humanized Computing**, v. 12, n. 7, p. 6117-6130, 2021.

LENARDUZZI, Valentina et al. Does migrating a monolithic system to microservices decrease the technical debt? **Journal of Systems and Software**, v. 169, p. 110710, 2020.

LEUNGWATTANAKIT, Watcharin et al. Modular Software Model Checking for Distributed Systems. **Ieee Transactions on Software Engineering**, v. 40, n. 5, p. 483-501, 2014.

LUCATO, Wagner Cezar et al. Model to measure the degree of competitiveness for auto parts manufacturing companies. **International Journal of Production Research**, v. 50, n. 19, p. 5508-5522, 2012.

MAZHAR, M. I.; REHMANI, M. H.; REHMANI, M. A. Microservices architecture for the internet of things: a survey. **Journal of Ambient Intelligence and Humanized Computing**, v. 9, p. 14792-14813, 2021.

MELL, Peter; GRANCE, Timothy. The NIST Definition of Cloud Computing. **NIST Special Publication**, v. 800, n. 145, p. 1-7, 2011.

MENDONÇA, Nabor C. et al. The monolith strikes back: Why istio migrated from microservices to a monolithic architecture. **IEEE Software**, v. 38, n. 5, p. 17-22, 2021.

MICROSOFT. Escalar verticalmente um aplicativo no Serviço de Aplicativo do Azure. **Microsoft**, 2022a. Disponível em: https://docs.microsoft.com/pt-br/azure/app-service/manage-scale-up. Acesso em: 05 jun. 2022.

_____. Service Level Agreements (SLA) for Online Services. **Microsoft**, 2022b. Disponível em: https://azure.microsoft.com/en-us/support/legal/sla. Acesso em: 04 jun. 2022.

MISHRA, Mayank; KUNDE, Shruti; NAMBIAR, Manoj. Cracking the monolith: challenges in data transitioning to cloud native architectures. *In:* **ECSA '18: PROCEEDINGS OF THE 12TH EUROPEAN CONFERENCE ON SOFTWARE ARCHITECTURE: COMPANION PROCEEDINGS**, 2018, p. 1-4.

MOHER, David et al. Preferred reporting items for systematic reviews and meta-analyses: the PRISMA statement. **International Journal of Surgery**, v. 8, n. 5, p. 336–341, 2010.

MOTIWALLA, Luvai F.; THOMPSON, Jeff. **Enterprise systems for management**. Pearson, 2012.

MUKHERJEE, Shantanu; ROY, Sandip; BOSE, Rajesh. Defining an appropriate trade-off to overcome the challenges and limitations in Software Security Testing. **Journal of Xidian University**, v. 14, n. 7, p. 1471-1479, 2020.

NAIK, N.; PATEL, M. SOA and Microservices: A Comparative Study. *In:* **IEEE INTERNATIONAL CONFERENCE ON COMPUTER COMMUNICATION AND CONTROL (IC4)**, 2017, v. 12, p. 1186-1191.

NAMIOT, D.; SNEPS-SNEPPE, M. Scalability and flexibility of microservices architecture: A systematic mapping study. **Journal of Systems and Software**, v. 169, p. 11073, 2020.

NARAYANAN, A.; VITHANI, N. Microservices Architecture: Advantages and Challenges. *In:* **PROCEEDINGS OF THE 2ND INTERNATIONAL CONFERENCE ON INVENTIVE RESEARCH IN COMPUTING APPLICATIONS**, 2020, v. 132, p. 1073-1077.

NEHME, Antonio et al. Securing Microservices. **It Professional**, 2019.

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

OPPL, Stefan. Subject-Oriented Elicitation of Distributed Business Process Knowledge. *In:* SCHMIDT, W. (eds.). S-BPM ONE - Learning by Doing - Doing by Learning. S-BPM ONE 2011. **Communications in Computer and Information Science**, v. 213, p. 16-33, 2011.

PACCHINI, Athos Paulo Tadeu. **O grau de prontidão das empresas industriais para implantação da indústria 4.0**: um estudo no setor automotivo brasileiro. 197 f. Tese (Mestrado em Engenharia de Produção) – Universidade Nove de Julho, São Paulo, 2019.

PALIWODA, Stanley J. Predicting the future using Delphi. **Management Decision**, v. 21, n. 1, p. 31-38, 1983.

PASQUALI, Luiz. Validade dos testes psicológicos: será possível reencontrar o caminho? **Psicologia: Teoria e Pesquisa**, v. 23, n. especial, p. 99-107, 2007.

PAZAZOGLOU, D.; TRAVERSO, P. Interoperability in service-oriented architectures: A systematic review. **Information and Software Technology**, v. 91, p. 1-15, 2017.

POP, Dragos-Paul; ALTAR, Adam. Designing an MVC Model for Rapid Web Application Development. **Procedia Engineering**, v. 69, p. 1172-1179, 2014.

PRASANDY, Teguh; MURAD, Dina Fitria; DARWIS, Taufik. Migrating application from monolith to microservices. *In:* **2020 INTERNATIONAL CONFERENCE ON INFORMATION MANAGEMENT AND TECHNOLOGY (ICIMTECH)**, 2020, Bandung, Indonesia, p. 726-731.

PRISMA. Welcome to the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) website! **PRISMA**, 2015. Disponível em: http://prisma-statement.org. Acesso em: 22 maio 2022.

RAHMAN, Mazedur; GAO, Jerry. A reusable automated acceptance testing architecture for microservices in behavior-driven development. *In:* **2015 IEEE SYMPOSIUM ON SERVICE-ORIENTED SYSTEM ENGINEERING**, 2015, San Francisco, CA, Estados Unidos, p. 321-325.

RED HAT. O que é API? **Red Hat**, 2023. Disponível em: https://www.redhat.com/pt-br/topics/api/what-are-application-programming-interfaces. Acesso em: 09 mar. 2023.

REED, M.; SHARMA, S.; ISERMANN, R. **Microservices architecture: Practical adoption for the enterprise.** 2016. Disponível em: https://www.ibm.com/downloads/cas/ZBDMJE0W. Acesso em: 09 mar. 2023.

SATHISHKUMAR, P.; KAVITHA, V. An empirical study on microservices architecture. **International Journal of Advanced Science and Technology**, v. 29, n. 8, p. 3262-3270, 2020.

SAVCHENKO, D.; RADCHENKO, Gleb; TAIPALE, Ossi. Microservices validation: Mjolnirr platform case study. *In:* **2015 38TH INTERNATIONAL CONVENTION ON INFORMATION AND COMMUNICATION TECHNOLOGY, ELECTRONICS AND MICROELECTRONICS (MIPRO)**, 2015, Opatija, Croácia, p. 235-240.

SEBRAE. Micro e pequenas empresas geram 27% do PIB do Brasil. **SEBRAE**, 2022. Disponível em: https://www.sebrae.com.br/sites/PortalSebrae/ufs/mt/noticias/micro-e-pequenas-empresas-geram-27-do-pib-do-brasil,ad0fc70646467410VgnVCM2000003c74010aRCRD. Acesso em: 05 jan. 2023.

SILVA, Hugo H. O. S.; CARNEIRO, Glauco de F.; MONTEIRO, Miguel P. Towards a roadmap for the migration of legacy software systems to a microservice based architecture. *In:* **PROCEEDINGS OF THE 9TH INTERNATIONAL CONFERENCE ON CLOUD COMPUTING AND SERVICES SCIENCE CLOSER**, 2019, Heraklion, Crete, Greece, v. 1, p. 37-47.

SINGH, A.; KUMAR, S.; SINGH, V. Challenges and solutions in the implementation of microservices architecture. *In:* **PROCEEDINGS OF THE 2018 INTERNATIONAL**

CONFERENCE ON EMERGING TECHNOLOGIES AND INNOVATIONS, 2018, v. 41, p. 1155-1162.

TAIBI, Davide; LENARDUZZI, Valentina; PAHL, Claus. Microservices Anti-patterns: a taxonomy. *In:* BUCCHIARONE, A. et al. **Microservices**. Springer, Cham, 2019. p. 111-128.

TERZIĆ, Branko et al. Development and evaluation of MicroBuilder: a Model-Driven tool for the specification of REST Microservice Software Architectures. **Enterprise Information Systems**, v. 12, n. 8-9, p. 1034-1057, 2018.

TIINSIDE. Gartner prevê que gastos mundiais de TI crescerão 5,1% em 2022. **tiinside**, 2022. Disponível em: https://tiinside.com.br/19/01/2022/gartner-preve-que-gastos-mundiais-de-ti-crescerao-51-em-2022. Acesso em: 10 maio 2022.

VIENNOT, Nicolas et al. A Microservices Architecture for Heterogeneous-Database Web Application. *In:* **EUROSYS '15: PROCEEDINGS OF THE TENTH EUROPEAN CONFERENCE ON COMPUTER SYSTEMS**, n. 21, p. 1-16, 2015.

VIGGIATO, Markos et al. Microservices in practice: A survey study. **arXiv**, v. 147, p. 92-108, 2018.

VILLAMIZAR, Mario et al. Cost comparison of running web applications in the cloud using monolithic, microservice, and AWS Lambda architectures. **Service Oriented Computing and Applications**, v. 11, n. 2, p. 233-247, 2017.

VILLAMIZAR, M. et al. Evaluation of the Overhead of a Commercial and an Open-Source Cloud Platform. **IEEE Latin America Transactions**, v. 13, n. 3, p. 1066-1072, 2015.

WOLFART, Daniele et al. Modernizing legacy systems with microservices: A roadmap. *In:* **EASE '21: PROCEEDINGS OF THE 25TH INTERNATIONAL CONFERENCE ON EVALUATION AND ASSESSMENT IN SOFTWARE ENGINEERING**, 2021, p. 149-159.

YU, H. et al. Microservice architecture: An empirical study of practitioners' perspectives. **IEEE Transactions on Services Computing**, v. 12, n. 5, p. 658-672, 2019.

APÊNDICE A – Convite para participação na pesquisa e termo de consentimento

Convite para participação na pesquisa “Questões para validar um instrumento de pesquisa do tipo Questionário”

O mestrando em Informática e Gestão do Conhecimento, Wilians Douglas Conde Souza, vinculado à Universidade Nove de Julho (Uninove), convida vossa senhoria para participar do preenchimento das questões para validação de um instrumento de pesquisa desenvolvida sob a orientação do professor doutor Ivanir Costa.

O objetivo desta pesquisa é validar se as questões sobre migração de sistemas de informação do ambiente monolítico para ambiente de Microsserviços na computação em nuvem em microempresas de software podem ser utilizadas como instrumento de pesquisa deste trabalho que visa contribuir para a melhoria da avaliação da competência do profissional do setor privado que pode contribuir para a tomada de decisão das microempresas na implementação da arquitetura de Microsserviços em seus softwares.

A pesquisa é composta por um questionário que levará apenas alguns minutos para ser preenchido.

As respostas individuais serão manuseadas apenas pelo pesquisador e seu orientador. O resultado será amplamente divulgado pela dissertação e periódicos científicos, porém a identidade dos participantes será preservada, com o sigilo das respostas garantido.

Não há despesas pessoais para o participante em qualquer fase do estudo. Também não há compensação financeira relacionada à sua participação.

Link para acessar o instrumento de pesquisa
https://forms.gle/SSe34BoPiXVSCzmz6

Link para acessar as questões que validam o instrumento de pesquisa
https://forms.gle/By1Gs3DsNSe3RUiK8

APÊNDICE B – Validação da qualidade do instrumento de pesquisa (questionário)

1. A quantidade de questões apresentadas é suficiente para as respostas sobre a viabilidade na migração da arquitetura do software em uma microempresa de software?
   SIM ( ) NÃO ( )
   Comente sua resposta:

2. As questões estão formuladas de forma clara, concreta e precisa?
   SIM ( ) NÃO ( )
   Comente sua resposta:

3. O conteúdo das questões constantes no questionário pode ser utilizado para medir a viabilidade na migração da arquitetura do software em uma microempresa de software?
   SIM ( ) NÃO ( )
   Comente sua resposta:

4. As questões deixam claro os requisitos para a migração da arquitetura do software em uma microempresa de software?
   SIM ( ) NÃO ( )
   Comente sua resposta:

5. As questões deixam claro que a pesquisa pretende apresentar os desafios financeiros e humanos para a migração da arquitetura do software em uma microempresa de software?
   SIM ( ) NÃO ( )
   Comente sua resposta:

6. É necessário mudar a composição ou estrutura de alguma questão?
   SIM ( ) NÃO ( )
   Comente sua resposta:

7. Para efeito da medição da relevância das práticas do questionário será utilizada uma escala Likert de cinco pontos com as seguintes opções: discordo totalmente, discordo, indiferente, concordo e concordo totalmente. Você concorda com as opções que serão utilizadas na escala Likert?

   SIM ( ) NÃO ( )

   Comente sua resposta:

APÊNDICE C – Definição sobre arquitetura de software, computação em nuvem e arquitetura monolítica e de Microsserviços para os especialistas avaliarem a qualidade do instrumento de pesquisa (questionário) que está sendo proposto.

Microempresa de software

No Brasil, existem 6.454 empresas que atuam no desenvolvimento e produção de software, sendo que cerca de 95% delas são classificadas como micro e pequenas empresas (até 99 funcionários), das quais 48,2% são consideradas Microempresas (MEs) (com menos do que 10 funcionários), com faturamento estimado em R$ 18,2 bilhões (ABES, 2023).

A entrega rápida de software é crítica para empresas que querem sobreviver e prosperar em um mercado em constante evolução (HUANG et al., 2021). Com isso, há uma grande tendência das empresas sem capital para investir em pesquisa, seguirem o que vem dando certo no mercado.

Arquitetura de software

A origem dos estudos de arquitetura de software foram no período de 1968 a 1970, onde Edsger Dijkstra escreveu sobre a ideia de software estruturado por camadas, e em seguida, Fred Brooks e Ken Iverson trataram de estrutura de computadores no ponto de vista de programação, e em 1970, com uma das principais contribuições para a engenharia de software, David Parnas descreve algumas das premissas e princípios da arquitetura de software e a importância da estrutura de sistemas (BASS; CLEMENTS; KAZMAN, 2012). Ainda segundo esses autores, a arquitetura de software é um conjunto de estruturas que formam o sistema e suas relações internas, com isso, ela é primordial durante o processo de desenvolvimento do sistema, pois é importante notar as consequências decorrentes das decisões, as quais podem gerar vantagens ou desvantagens na resolução de problemas.

Computação em Nuvem

A Computação em Nuvem refere-se tanto a aplicativos entregues como serviços na Internet quanto ao hardware e software de sistemas nos data centers que fornecem esses serviços (ARMBRUST et al., 2010). Ainda segundo os autores,

uma grande vantagem desse modelo é a possibilidade de se pagar pelo uso dos recursos de computação conforme o uso necessário (por exemplo, processadores por hora e armazenamento por dia) e liberá-los assim que possível, com isso essa solução é capaz de oferecer serviços abaixo dos custos de um data center de médio porte e ainda assim gerar um bom lucro.

Arquitetura de Microsserviços e Arquitetura Monolítica

Atualmente, dois paradigmas de Engenharia de Software dominam o desenvolvimento de sistemas corporativos modernos: monolítico e arquitetura baseada em Microsserviços (FOWLER, 2015).

Na arquitetura monolítica a estrutura de código fonte do sistema é centralizada em um único local, ou seja, quando uma aplicação é construída em uma grande e única estrutura, independe da complexidade, tamanho do código fonte ou mesma organizada em componentes, sua estrutura é em apenas uma unidade (SAVCHENKO; RADCHENKO; TAIPALE, 2015).

Microsserviços é um design arquitetural que decompõe suas funcionalidades por responsabilidade em pequenos serviços autônomos que trabalham em conjunto, o que os caracteriza como estruturas reduzidas de software, que possuem interfaces padronizadas para se comunicarem, além da possibilidade que as funcionalidades possam ser escaladas isoladamente (NEWMAN, 2015).

APÊNDICE D – Questionário sobre experiências de especialistas sobre orientações na migração da arquitetura monolítica para a arquitetura de microsserviços

Questões sobre experiências anteriores na migração da arquitetura monolítica para a arquitetura de Microsserviços

1. O que motivou a empresa a migrar da arquitetura monolítica para arquitetura de Microsserviços em nuvem?
   Os respondentes utilizarão um campo aberto para expressar sua resposta.

   Esta questão é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

2. Quais informações/métricas foram consideradas antes da migração?
   Esta questão é relevante para o questionário?
   Os respondentes utilizarão um campo aberto para expressar sua resposta.

   Esta questão é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

3. Quais informações/métricas foram consideradas após a migração?
   Os respondentes utilizarão um campo aberto para expressar sua resposta.

   Esta questão é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
   Os respondentes utilizarão um campo aberto para expressar sua resposta.

   Esta questão é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

Afirmações sobre *Continuous Integration* e *Continuous Delivery* (DevOps) para constarem no questionário

5. A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura de nuvem e também dos Microsserviços (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016).
   Os respondentes marcarão uma das opções abaixo:
   ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

   Esta afirmação é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

6. Para a equipe ter produtividade atuando com Microsserviços é necessário ter entrega e implantação contínua dos artefatos de software até sua liberação para produção (ACEVEDO; GÓMEZ Y JORGE; PATIÑO, 2017).
   Os respondentes marcarão uma das opções abaixo:
   ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

   Esta afirmação é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

7. Sem a cultura DevOps, fica muito difícil lidar com os múltiplos serviços, suas implantações e validar as ações do serviço (KALSKE; MÄKITALO; MIKKONEN, 2018).
   Os respondentes marcarão uma das opções abaixo:
   ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

   Esta afirmação é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

8. Para atuar com Microsserviços, a automatização do gerenciamento de configuração deve ser implantada (WOLFART et al., 2021).
   Os respondentes marcarão uma das opções abaixo:
   ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

   Esta afirmação é relevante para constar no questionário?
   SIM ( ) NÃO ( )
   Comente sua resposta:

Afirmações sobre *Domain Driven Design* (DDD) para identificar os Microsserviços candidatos

9. O DDD é um método que pode ser utilizado para facilitar a extração de serviços com baixo acoplamento e propõe a definição de contextos compostos por componentes do modelo de negócio que sejam consistentes entre si (FAN; MA, 2017).
   Os respondentes marcarão uma das opções abaixo:
   ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?
SIM ( ) NÃO ( )
Comente sua resposta:

10. É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, suportar a migração (SILVA; CARNEIRO; MONTEIRO, 2019).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

11. É recomendado utilizar o DDD para encontrar candidatos a Microsserviços no sistema original. Os resultados da análise de contexto limitado são uma ferramenta chave para identificar candidatos a Microsserviços (KAZANAVIČIUS; MAŽEIKA, 2020).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

12. Um dos métodos mais eficazes para identificar os serviços candidatos é identificando os domínios e subdomínios com contexto e limites claros (WOLFART et al., 2021).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?
SIM ( ) NÃO ( )
Comente sua resposta:

Afirmações sobre Segregação de Banco de dados

13. Uma vantagem desse modelo é que cada microsserviço pode possuir seu banco de dados que pode inclusive ser NoSQL (FURDA et al., 2018).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

14. Dividir os dados em bancos de dados separados pode causar inconsistência de dados especialmente quando estiver em um contexto de transação (KAZANAVIČIUS; MAŽEIKA, 2020).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

15. Os procedimentos para resolver essas inconsistências nos repositórios pode afetar o desempenho do aplicativo (MISHRA; KUNDE; NAMBIAR, 2018).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?
SIM ( ) NÃO ( )
Comente sua resposta:

16. Uma grande desvantagem é a separação de dados, pois, além de possíveis inconsistências, as soluções podem trazer sobrecarga de comunicação adicionada da rede ou chamadas externas também podem introduzir latência (GRAVANIS; KAKARONTZAS; GEROGIANNIS, 2021).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

Afirmações sobre o Perfil necessário da equipe para realizar a migração

17. Para que os benefícios da arquitetura de Microsserviços possa ser desfrutado é necessário maturidade da aplicação, domínio intelectual do código-fonte e uma equipe de desenvolvedores qualificados em aplicações distribuídas (ACEVEDO; GÓMEZ Y JORGE; PATIÑO, 2017).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

18. Quando a organização adota o estilo de arquitetura de Microsserviços, as equipes devem ter mais liberdade e responsabilidade, mas menos processos. Isso significa que as equipes podem implantar seu serviço na produção quando

precisarem, em vez de esperar pela aprovação de outra pessoa (KALSKE; MÄKITALO; MIKKONEN, 2018).

Os respondentes marcarão uma das opções abaixo:

○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?

SIM ( ) NÃO ( )

Comente sua resposta:

19. A equipe deve possuir uma ótima qualificação para fazer a extração correta de Microsserviços do sistema monolítico. É uma tarefa muito difícil e crucial para uma migração bem-sucedida (KAZANAVIČIUS; MAŽEIKA, 2020).

    Os respondentes marcarão uma das opções abaixo:

    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?

    SIM ( ) NÃO ( )

    Comente sua resposta:

20. Com a adoção de Microsserviços é necessário destacar um indivíduo para monitorar os Microsserviços/infraestrutura para garantir a disponibilidade do serviço e monitorar gargalos, desempenho e uso da infraestrutura e recursos. (WOLFART et al., 2021).

    Os respondentes marcarão uma das opções abaixo:

    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?

    SIM ( ) NÃO ( )

    Comente sua resposta:

Afirmações sobre por onde começar a migração

21. É melhor começar pelos serviços mais fáceis e óbvios e quando a organização tiver mais conhecimento sobre arquitetura de Microsserviços então os serviços podem se tornar mais refinados (KALSKE; MÄKITALO; MIKKONEN, 2018).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

22. Iniciar com a funcionalidade que tem o menor impacto quando comparada às demais. Esse processo facilita a validação dos limites estabelecidos entre os recursos com o menor risco de efeitos colaterais para validar o mapeamento de contexto (SILVA; CARNEIRO; MONTEIRO, 2019).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

23. Migrar um sistema de software de uma arquitetura monolítica para uma arquitetura de Microsserviços não é uma tarefa trivial. Uma abordagem gradual (iterativa) da migração é a mais aconselhável (WOLFART et al., 2021).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?
SIM ( ) NÃO ( )
Comente sua resposta:

24. Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021).
Os respondentes marcarão uma das opções abaixo:
○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?
SIM ( ) NÃO ( )
Comente sua resposta:

Afirmações sobre utilizar UML para identificar os Microsserviços candidatos

25. No diagrama de classes, cada classe pode ser representada como uma unidade funcional ou um componente e assim pode ser mapeada para um microsserviço (MISHRA; KUNDE; NAMBIAR, 2018).
Os respondentes marcarão uma das opções abaixo:
○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?
SIM ( ) NÃO ( )
Comente sua resposta:

26. Contextos limitados podem ser separados por meio da decomposição por verbos (casos de uso) ou por substantivos (recursos) e dessa forma é possível identificar os Microsserviços candidatos (FRITZSCH et al., 2019).

Os respondentes marcarão uma das opções abaixo:

○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?

SIM ( ) NÃO ( )

Comente sua resposta:

27. O primeiro passo é a transformação do monolito na representação gráfica. No grafo, cada vértice representa a classe do monolito e as arestas não direcionadas representam seu acoplamento com outras classes do monolito. Ao cortar o gráfico em componentes é possível observar os candidatos a Microsserviços (SILVA; CARNEIRO; MONTEIRO, 2019).

Os respondentes marcarão uma das opções abaixo:

○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?

SIM ( ) NÃO ( )

Comente sua resposta:

28. Através de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML. Em seguida listar todas as funcionalidades para as quais o sistema foi projetado e para identificar os Microsserviços candidatos, deve fazer a identificação dos artefatos de implementação de cada recurso (WOLFART et al., 2021).

Os respondentes marcarão uma das opções abaixo:

○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?

SIM ( ) NÃO ( )

Comente sua resposta:

Afirmações sobre a infraestrutura mínima para implementar Microsserviços

29. O Circuit Breaker monitora as falhas e, quando houver falhas suficientes, as chamadas subsequentes para a dependência não serão feitas e, em vez disso, um erro será retornado, em vez de adicionar mais carga à dependência. Por isso o *circuit breaker* deve ser implementado, quando a organização tem mais do que alguns Microsserviços, também deve levar em consideração a possibilidade de um serviço não responder (KALSKE; MÄKITALO; MIKKONEN, 2018).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

30. O API gateway é parte integrante da arquitetura de Microsserviços e deve ser implantado. O gateway serve como uma camada de conexão para clientes, redirecionando as solicitações para o microsserviço correto, servindo como proxy para os diferentes serviços (HAUGELAND et al., 2021).
    Os respondentes marcarão uma das opções abaixo:
    ○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

    Esta afirmação é relevante para constar no questionário?
    SIM ( ) NÃO ( )
    Comente sua resposta:

31. O Service Discovery deve ser implantado, pois, para manter o controle dos serviços implantados e seu endereço exato e número de porta é uma tarefa complicada e com essa solução é possível obter as instâncias disponíveis de cada serviço (MISHRA; KUNDE; NAMBIAR, 2018).

Os respondentes marcarão uma das opções abaixo:

○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

Esta afirmação é relevante para constar no questionário?

SIM ( ) NÃO ( )

Comente sua resposta:

32. O Docker deve ser implantado, pois torna o ambiente facilmente portátil e isolado. Não há conflitos de dependências ou a necessidade de configurar cada ambiente (WOLFART et al., 2021).

Os respondentes marcarão uma das opções abaixo:

○ Discordo totalmente ○ Discordo ○ Indiferente ○ Concordo ○ Concordo totalmente

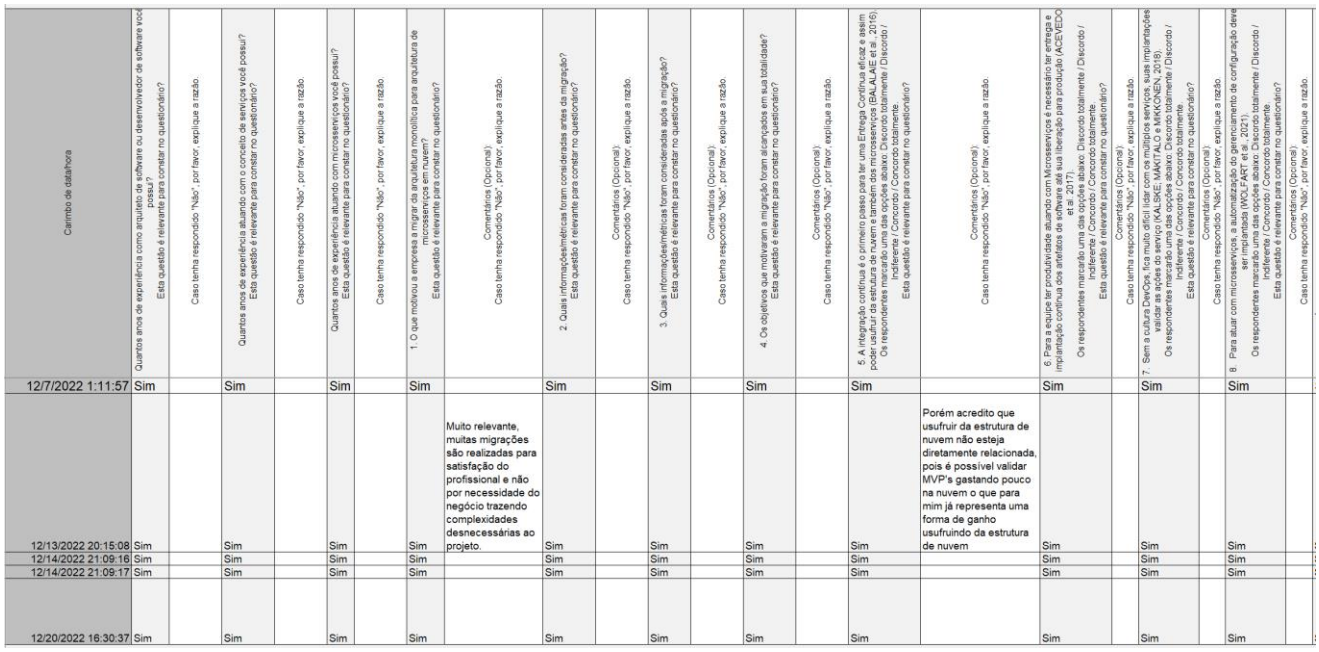
Esta afirmação é relevante para constar no questionário?

SIM ( ) NÃO ( )

Comente sua resposta:

APÊNDICE E – Respostas do piloto referente ao questionário sobre experiências de especialistas sobre orientações na migração da arquitetura monolítica para a arquitetura de microsserviços

Tabela 5 – Respostas do piloto referente ao questionário sobre experiências de especialistas sobre orientações na migração da arquitetura monolítica para a arquitetura de microsserviços



Carimbo de data/hora	Quantos anos de experiência como arquiteto de software ou desenvolvedor de software você possui? Esta questão é relevante para constar no questionário?		Quantos anos de experiência atuando com o conceito de serviços você possui? Esta questão é relevante para constar no questionário?		Quantos anos de experiência atuando com microsserviços você possui? Esta questão é relevante para constar no questionário?		1. O que motivou a empresa a migrar da arquitetura monolítica para arquitetura de microsserviços? Esta questão é relevante para constar no questionário?		2. Quais informações/métricas foram consideradas antes da migração? Esta questão é relevante para constar no questionário?		3. Quais informações/métricas foram consideradas após a migração? Esta questão é relevante para constar no questionário?		4. Os objetivos que motivaram a migração foram alcançados em sua totalidade? Esta questão é relevante para constar no questionário?		5. A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura de ambiente em nuvem para microsserviços (BALALAIE et al., 2016). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?		6. Para a equipe ter produtividade atuando com Microsserviços é necessário ter entrega e implantação contínua dos artefatos de software (até sua liberação para produção) (ACEVEDO et al., 2017). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?		7. Sem a cultura DevOps, fica muito difícil lidar com os múltiplos serviços, suas implantações validar as ações do serviço (KALSKE; MÄKITALO e MIKKONEN, 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?		8. Para atuar com microsserviços, a automatização do gerenciamento de configuração deve ser implantada (WOLFART et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	
12/7/2022 1:11:57	Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	
12/13/2022 20:15:08	Sim		Sim		Sim		Sim	Muito relevante, muitas migrações são realizadas para satisfação do profissional e não por necessidade do negócio trazendo complexidades desnecessárias ao projeto.	Sim		Sim		Sim		Sim	Porém acredito que usufruir da estrutura de nuvem não esteja diretamente relacionada, pois é possível validar MVP's gastando pouco na nuvem o que para mim já representa uma forma de ganho usufruindo da estrutura de nuvem	Sim		Sim		Sim	
12/14/2022 21:09:16	Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	
12/14/2022 21:09:17	Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	
12/20/2022 16:30:37	Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	

9. O DDD É um método que pode ser utilizado para facilitar a extração de serviços com baixo acoplamento e propõe a definição de contextos compostos por componentes do modelo de negócio que sejam consistentes entre si (FAN e MA, 2017). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	10. É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, suportar a migração (SILVA et al., 2019). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	11. É recomendado utilizar o DDD para encontrar candidatos a microsserviços no sistema original. Os resultados da análise de contexto limitado são uma ferramenta chave para identificar candidatos a microsserviços (KALSKE; MÄKITALO e MIKKONEN, 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	12. Um dos métodos mais eficazes para identificar os serviços candidatos é identificando os domínios e subdomínios com contexto e limites claros (WOLFART et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	13. Uma vantagem dessa arquitetura é que cada microserviço pode possuir seu banco de dados que pode incluíve ser NoSQL (FURDA et al., 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	14. Dividir os dados em bancos de dados separados pode causar inconsistência de dados especialmente quando estiver em um contexto de transação (KAZANAVICIUS e MAŽEIKA, 2020). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	15. Os procedimentos para resolver essas inconsistências nos repositórios pode afetar o desempenho do aplicativo (MISHRA; KUNDE e NAMBIAR, 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	16. Uma grande desvantagem é a separação de dados, pois, além de possíveis inconsistências as soluções podem trazer sobrecarga de comunicação adicionada da rede ou chamadas externas também podem introduzir latência (GRAVANIS; KAKARONTZAS e KONSTANTINOU, 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	17. Para que os benefícios da arquitetura de Microsserviços possa ser desfrutados é necessário maturidade da aplicação, domínio intelectual do código-fonte e uma equipe de desenvolvedores qualificados em aplicações distribuídas (ACEVEDO et al., 2017). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.	18. Quando a organização adota o estilo de arquitetura de microsserviços, as equipes devem ter mais liberdade e responsabilidade, mas menos processos. Isso significa que as equipes podem ser mais ágeis e inovadoras, o que pode ser um benefício para atrair pela inovação de outra pessoa (KALSKE; MÄKITALO e MIKKONEN, 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.
Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	
Sim		Não	acredito que a forma como está a questão fica difícil de entender.	Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	
Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	
Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim		Sim	
Sim		Sim		Sim		Sim	uma vantagem e um trabalho que deve ser muito bem pensado	Sim		Sim		Sim		Sim	para fazer um contraponto das anteriores	Sim		Sim		Sim		Sim	

Questão					
19. A equipe deve possuir uma ótima qualificação para fazer a extração correta de microsserviços do sistema monolítico. É uma tarefa muito difícil e crucial para uma migração bem-sucedida (KAZANAVIČIUS e MAŽEIKA, 2020). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
20. Com a adoção de microsserviços é necessário destacar um indivíduo para monitorar os microsserviços/infraestrutura para garantir a disponibilidade do serviço e monitorar gargalos, desempenho e uso da infraestrutura e recursos. (WOLFART et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Não
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
21. É melhor começar pelos serviços mais fáceis e óbvios e quando a organização tiver mais conhecimento sobre arquitetura de microsserviços então os serviços podem se tornar mais refinados (KALSKE; MÄKITALO e MIKKONEN, 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
22. Iniciar com a funcionalidade que tem o menor impacto quando comparada às demais. Esse processo facilita a validação dos limites estabelecidos entre os recursos com o menor risco de efeitos colaterais para validar o mapeamento de contexto (SILVA et al., 2019). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
23. Migrar um sistema de software de uma arquitetura monolítica para uma arquitetura de microsserviços não é uma tarefa trivial. Uma abordagem gradual (iterativa) da migração é a mais aconselhável (WOLFART et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Não
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					redundante
24. Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Não
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.		Acredito que vale a pergunta, mas acredito que passos de maturidade anteriores, deixariam essa pergunta e a de deixar um individuo responsável pelo monitoramento meio sem sentido.			
25. Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Não	Sim	Sim	Não
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.		pergunta duplicada			
26. Contextos limitados podem ser separados por meio da decomposição por verbos (casos de uso) ou por substantivos (recursos) e dessa forma é possível identificar os microsserviços candidatos (FRITZSCH et al., 2019). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
27. O primeiro passo é a transformação do monólito na representação gráfica. No grafo, cada vértice representa a classe do monólito e as arestas não direcionadas representam seu acoplamento com outras classes do monólito. Ao cortar o gráfico em componentes é possível observar os candidatos a microsserviços (SILVA et al., 2019). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.		ansioso para saber o resultado dessa :)			
28. Através de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML. Em seguida listar todas as funcionalidades para as quais o sistema foi projetado e para identificar os Microsserviços candidatos, deve fazer a identificação dos artefatos de implementação de cada recurso (WOLFART et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Não
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					

Questão					
29. O Circuit Breaker monitora as falhas e, quando houver falhas suficientes, as chamadas subsequentes para a dependência não serão feitas e, em vez disso, um erro será retornado, em vez de adicionar mais carga à dependência. Por isso o circuit breaker deve ser implementado, quando a organização tem mais do que alguns microsserviços, também deve levar em consideração a possibilidade de um serviço não responder (KALSKE; MÄKITALO e MIKKONEN, 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
30. O API gateway é parte integrante da arquitetura de microsserviços e deve ser implantado. O gateway serve como uma camada de conexão para clientes, redirecionando as solicitações para o microsserviço correto, servindo como proxy para os diferentes serviços (HAUGELAND et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
31. O Service Discovery deve ser implantado, pois, para manter o controle dos serviços implantados e seu endereço exato e número de porta é uma tarefa complicada e com essa solução é possível obter as instâncias disponíveis de cada serviço (MISHRA; KUNDE e NAMBIAR, 2018). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Sim
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
32. O Docker deve ser implantado, pois, torna o ambiente facilmente portátil e isolado. Não há conflitos de dependências ou a necessidade de configurar cada ambiente (WOLFART et al., 2021). Os respondentes marcarão uma das opções abaixo: Discordo totalmente / Discordo / Indiferente / Concordo / Concordo totalmente. Esta questão é relevante para constar no questionário?	Sim	Sim	Sim	Sim	Não
Comentários (Opcional): Caso tenha respondido "Não", por favor, explique a razão.					
Considerações finais (Opcional):		gostaria de receber o resultado, é uma pesquisa que pode gerar um conteúdo muito interessante.			

APÊNDICE F – Respostas do piloto referente à validação da qualidade do instrumento de pesquisa (questionário)

Tabela 6 – Respostas do piloto referente à validação da qualidade do instrumento de pesquisa (questionário)

Carimbo de data/hora	1. A quantidade de questões apresentadas é suficiente para as respostas sobre a viabilidade na migração da arquitetura do software em uma microempresa de software?	Caso tenha respondido "Não", por favor, explique a razão.	2. As questões estão formuladas de forma clara, concreta e precisa?	Caso tenha respondido "Não", por favor, explique a razão.	3. O conteúdo das questões constantes no questionário pode ser utilizado para medir a viabilidade na migração da arquitetura do software em uma microempresa de software?	Caso tenha respondido "Não", por favor, explique a razão.	4. As questões deixam claro os requisitos para a migração da arquitetura do software em uma microempresa de software?	Caso tenha respondido "Não", por favor, explique a razão.	5. As questões deixam claro que a pesquisa pretende apresentar os desafios financeiros e humanos para a migração da arquitetura do software em uma microempresa de software?	Caso tenha respondido "Não", por favor, explique a razão.	6. É necessária mudar a composição ou estrutura de alguma questão	Caso tenha respondido "Não", por favor, explique a razão.	7. Para efeito da medição da relevância das práticas do questionário será utilizada uma escala Likert de cinco pontos com as opções: discordo totalmente, discordo, indiferente, concordo e concordo totalmente. Você concorda com as opções que serão utilizadas na escala Likert?	Caso tenha respondido "Não", por favor, explique a razão.	8. Por favor, coloque seus comentários sobre qualquer questão que tenha respondido "Não" ou qualquer sugestão que queira fazer.
12/7/2022 1:23:31	Sim		Sim		Sim		Sim		Sim		Não		Sim		
12/13/2022 20:45:28	Sim		Sim		Sim		Sim		Sim		Não		Sim		
12/14/2022 21:14:55	Sim		Sim		Sim		Sim		Sim		Não	Todas as questões estão em pleno acordo.	Sim		
12/14/2022 21:14:56	Sim		Sim		Sim		Sim		Sim		Não		Sim		
12/20/2022 16:32:54	Sim		Sim		Não	pois não trata do cenário em si, nem metricas, nem desafios ...	Não	esclarecem alguns pontos ...	Não		Não		Sim		

APÊNDICE G – Método Delphi (Rodada 1)

Questionário publicado

Perfil do respondente:

• Função na organização.

• Quantos anos de experiência você possui em sua função?

• Quantos anos de experiência como arquiteto de software ou desenvolvedor de software você possui?

• Quantos anos de experiência atuando com o conceito de serviços você possui?

• Quantos anos de experiência atuando com Microsserviços você possui?

• Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação”?

Motivações para a migração

• Por que a empresa decidiu migrar?

Informações/métricas de migração

• Antes da migração, quais as informações/métricas foram consideradas?

• Quais informações/métricas foram consideradas após a migração?

• Os objetivos que motivaram a migração foram alcançados em sua totalidade?

DevOps (*Continuous Integration/Continuous Delivery*)

5. A integração contínua é o primeiro passo para ter uma Entrega Contínua eficaz e assim poder usufruir da estrutura dos Microsserviços em nuvem (BALALAIE; HEYDARNOORI; JAMSHIDI, 2016).

6. Para a equipe ter produtividade atuando com Microsserviços é necessário ter entrega e implantação contínua dos artefatos de software até sua liberação para produção (ACEVEDO; GÓMEZ Y JORGE; PATIÑO, 2017).

7. Sem a cultura DevOps, fica muito difícil lidar com os múltiplos serviços, suas implantações e validar as ações do serviço (KALSKE; MÄKITALO; MIKKONEN, 2018).

8. Para atuar com Microsserviços, a automatização do gerenciamento de configuração deve ser implantada (WOLFART et al., 2021).

Domain Driven Design (DDD) para identificar os microsserviços candidatos

9. O DDD é um método que pode ser utilizado para facilitar a extração de serviços com baixo acoplamento e propõe a definição de contextos compostos por componentes do modelo de negócio que sejam consistentes entre si (FAN; MA, 2017).

10. É possível utilizar o DDD para traduzir funcionalidades em domínio e subdomínio e, assim, identificar os serviços candidatos e suportar a migração (SILVA; CARNEIRO; MONTEIRO, 2019).

11. É recomendado utilizar o DDD para encontrar candidatos a Microsserviços no sistema original. Os resultados da análise de contexto limitado são uma ferramenta chave para identificar candidatos a Microsserviços (KAZANAVIČIUS; MAŽEIKA, 2020).

12. Um dos métodos mais eficazes para identificar os serviços candidatos é identificando os domínios e subdomínios com contexto e limites claros (WOLFART et al., 2021).

Segregação de Banco de dados

13. Uma vantagem dessa arquitetura é que cada microsserviço pode possuir seu banco de dados que pode inclusive ser NoSQL (FURDA et al., 2018).

14. Dividir os dados em bancos de dados separados pode causar inconsistência de dados especialmente quando estiver em um contexto de transação (KAZANAVIČIUS; MAŽEIKA, 2020).

15. Os procedimentos para resolver essas inconsistências nos repositórios pode afetar o desempenho do aplicativo (MISHRA; KUNDE; NAMBIAR, 2018).

16. Uma grande desvantagem é a separação de dados, pois, além de possíveis inconsistências, as soluções podem trazer sobrecarga de comunicação adicionada

da rede ou chamadas externas também podem introduzir latência (GRAVANIS; KAKARONTZAS e GEROGIANNIS, 2021).

Perfil da Equipe para realizar a migração

18. Quando a organização adota o estilo de arquitetura de Microsserviços, as equipes devem ter mais liberdade e responsabilidade, mas menos processos. Isso significa que as equipes podem implantar seu serviço na produção quando precisarem, em vez de esperar pela aprovação de outra pessoa (KALSKE; MÄKITALO; MIKKONEN, 2018).

19. A equipe deve possuir uma ótima qualificação para fazer a extração correta de Microsserviços do sistema monolítico. É uma tarefa muito difícil e crucial para uma migração bem-sucedida (KAZANAVIČIUS; MAŽEIKA, 2020).

20. Com a adoção de Microsserviços é necessário destacar um indivíduo para monitorar os Microsserviços/infraestrutura para garantir a disponibilidade do serviço e monitorar gargalos, desempenho e uso da infraestrutura e recursos (WOLFART et al., 2021).

Por Onde Começar a migração

21. É melhor começar pelos serviços mais fáceis e óbvios e quando a organização tiver mais conhecimento sobre arquitetura de Microsserviços então os serviços podem se tornar mais refinados (KALSKE; MÄKITALO; MIKKONEN, 2018).

22. Iniciar com a funcionalidade que tem o menor impacto quando comparada às demais. Esse processo facilita a validação dos limites estabelecidos entre os recursos com o menor risco de efeitos colaterais para validar o mapeamento de contexto (SILVA; CARNEIRO; MONTEIRO, 2019).

23. Migrar um sistema de software de uma arquitetura monolítica para uma arquitetura de Microsserviços não é uma tarefa trivial. Uma abordagem gradual (iterativa) da migração é a mais aconselhável (WOLFART et al., 2021).

24. Dividir a migração em diferentes fases, onde cada fase inclui a extração de um único serviço do sistema preexistente, bem como a adição da infraestrutura necessária para suportar a nova funcionalidade migrada do monolito (HAUGELAND et al., 2021).

UML para identificar os microsserviços candidatos

25. Contextos limitados podem ser separados por meio da decomposição por verbos (casos de uso) ou por substantivos (recursos) e dessa forma é possível identificar os Microsserviços candidatos (FRITZSCH et al., 2019).

26. O primeiro passo é a transformação do monolito na representação gráfica. No grafo, cada vértice representa a classe do monolito e as arestas não direcionadas representam seu acoplamento com outras classes do monolito. Ao cortar o gráfico em componentes é possível observar os candidatos a Microsserviços (SILVA; CARNEIRO; MONTEIRO, 2019).

27. Através de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML. Em seguida listar todas as funcionalidades para as quais o sistema foi projetado e para identificar os Microsserviços candidatos, deve fazer a identificação dos artefatos de implementação de cada recurso (WOLFART et al., 2021).

Infraestrutura

28. O Circuit Breaker monitora as falhas e, quando houver falhas suficientes, as chamadas subsequentes para a dependência não serão feitas e, em vez disso, um erro será retornado, em vez de adicionar mais carga à dependência. Por isso o circuit breaker deve ser implementado, quando a organização tem mais do que alguns Microsserviços, também deve levar em consideração a possibilidade de um serviço não responder (KALSKE; MÄKITALO; MIKKONEN, 2018).

29. O API gateway é parte integrante da arquitetura de Microsserviços e deve ser implantado. O gateway serve como uma camada de conexão para clientes,

redirecionando as solicitações para o microsserviço correto, servindo como proxy para os diferentes serviços (HAUGELAND et al., 2021).

30. O Service Discovery deve ser implantado, pois, para manter o controle dos serviços implantados e seu endereço exato e número de porta é uma tarefa complicada e com essa solução é possível obter as instâncias disponíveis de cada serviço (MISHRA; KUNDE; NAMBIAR, 2018).

31. O Docker deve ser implantado, pois torna o ambiente facilmente portátil e isolado. Não há conflitos de dependências ou a necessidade de configurar cada ambiente (WOLFART et al., 2021).

Comentários finais

Considerações finais:

Perfil dos respondentes

Os tempos de experiência na Tabela 7 são exibidos em anos.

Tabela 7 – Perfil dos respondentes Delphi (Rodada 1)

#	Resposta	Cargo	Tempo de experiência na área de desenvolvimento de *software*	Tempo de experiência atuando com orientação a serviços	Tempo de experiência atuando com microsserviços
1	1/12/2023 15:47:04	Desenvolvedor Java	12	2	1
2	1/13/2023 13:23:14	Desenvolvedor .NET	6	6	2
3	1/13/2023 20:16:34	Desenvolvedor Java	20	5	3
4	1/15/2023 13:28:33	Analista desenvolvedor PL/SQL	10	10	8

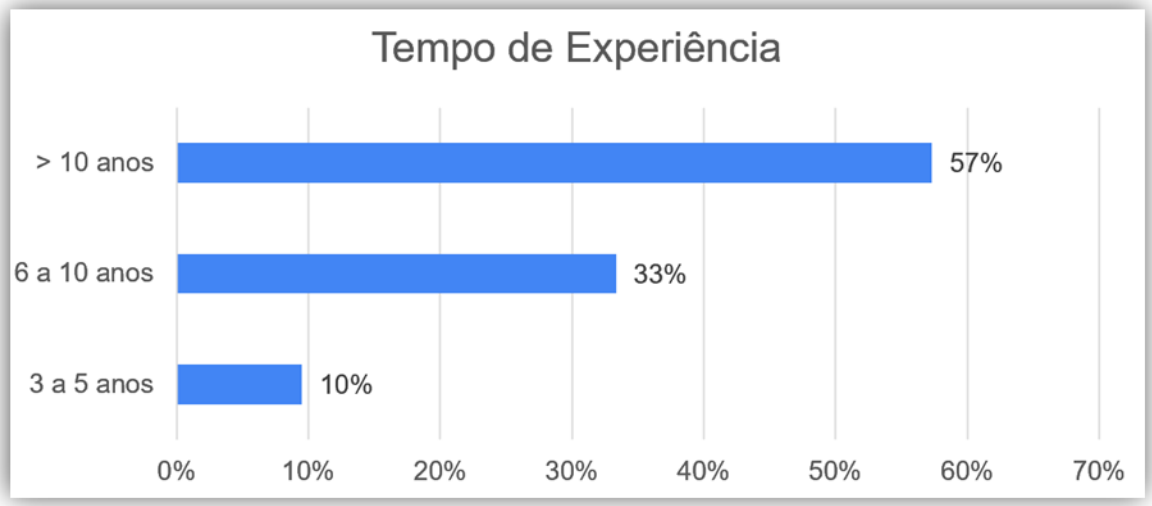
5	1/19/2023 11:42:32	Engenheiro de software	3	2	2
6	1/19/2023 12:48:45	Desenvolvedor	10	5	2
7	1/19/2023 12:56:56	Engenheiro de software	15	6	5
8	1/19/2023 13:41:28	Engenheiro de software	6	5	5
9	1/19/2023 16:09:37	Engenheiro de software	10	4	3
10	1/19/2023 16:48:02	Engenheiro de software	8	6	4
11	1/19/2023 22:31:35	Engenheiro de software	12	4	4
12	1/20/2023 10:01:55	Desenvolvedor	11	6	3
13	1/20/2023 16:58:57	Engenheiro de software	24	15	10
14	1/21/2023 2:04:07	Arquiteto de software	4	5	4
15	1/22/2023 7:48:02	Desenvolvedor	16	10	6
16	1/23/2023 15:25:11	Arquiteto de software	16	8	2
17	1/25/2023 1:17:16	Arquiteto de software	11	4	4
18	1/25/2023 9:20:29	Arquiteto de software	9	7	3
19	1/25/2023 12:17:50	Engenheiro de software	15	10	8

20	1/25/2023 14:43:20	Tech lead	11	11	5
21	1/30/2023 9:33:15	Arquiteto de software	22	18	7

Fonte: autor.

O gráfico da Figura 36 apresenta o tempo de experiência dos profissionais.

Figura 36 – Tempo de experiência dos especialistas Delphi (Rodada 1)

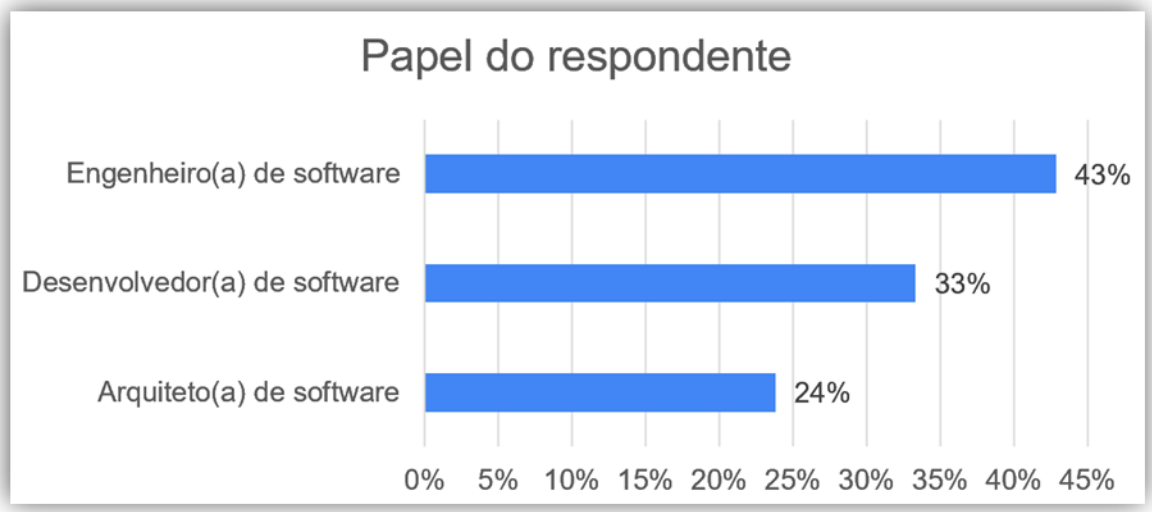


Fonte: autor.

Como apresenta o gráfico da Figura 36, 57% dos respondentes têm mais de 10 anos de experiência atuando no processo de desenvolvimento de *software*, 33% entre 6 e 10 anos, e 10% entre 3 e 5 anos de experiência.

O gráfico da Figura 37 apresenta o papel dos respondentes dentro da área de desenvolvimento de *software*.

Figura 37 – Papel dos respondentes Delphi (Rodada 1)



Fonte: autor.

Como apresenta o gráfico da Figura 37, 43% dos respondentes são engenheiros(as) de *software*, 33% são desenvolvedores de *software* e 24% são arquitetos(as) de *software*.

Respostas dos respondentes

Respondente #1

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Acho que não, pois dependendo do tamanho da empresa e do produto a ser ofertado, pode dar mais trabalho do que a suas vantagens.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Facilitar a manutenção do sistema e diminuir a complexidade de cada parte do sistema, se tornando menores e autocontido.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Foi utilizado o tamanho de linha de código no projeto, tempo para entrega de qualquer manutenção no sistema e complexidade para atualizar parte dele sem prejudicar o resto.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Foi utilizado o tamanho de linha de código no projeto, tempo para entrega de qualquer manutenção no sistema e complexidade para atualizar parte dele sem prejudicar o resto.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Quase todos, por conta de dependência feita na criação do projeto que se tornaria muito cara para serem refeitas.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo
6. Concordo
7. Concordo totalmente
8. Concordo

***Domain Driven Design* (DDD)**
9. Concordo totalmente
10. Concordo
11. Concordo
12. Concordo totalmente

Segregação de Banco de Dados
13. Indiferente
14. Concordo totalmente

15. Concordo
16. Concordo

Perfil da Equipe para realizar a migração

18. Concordo
19. Concordo totalmente
20. Concordo totalmente

Por Onde Começar a migração

21. Concordo totalmente
22. Concordo totalmente
23. Indiferente
24. Concordo totalmente

UML para identificar os microsserviços candidatos

25. Concordo
26. Indiferente
27. Indiferente

Infraestrutura

28. Indiferente
29. Concordo totalmente
30. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #2

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Concordo.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Não participei de nenhuma migração.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Não participei de nenhuma migração.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Não participei de nenhuma migração.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Não participei de nenhuma migração.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo
6. Concordo
7. Concordo totalmente
8. Indiferente

***Domain Driven Design* (DDD)**

9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo totalmente
14. Concordo

15. Concordo
16. Discordo

Perfil da Equipe para realizar a migração

18. Discordo
19. Concordo totalmente
20. Concordo

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo totalmente
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Indiferente

Infraestrutura

28. Concordo
29. Concordo totalmente
30. Indiferente
31. Concordo

Comentários finais

Respondente não deixou comentário.

Respondente #3

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Sim.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Modularização e disponibilização de serviços.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Volumetria inicial de transações diárias.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Volumetria de acessos e disponibilidade aceitável da infraestrutura.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo totalmente
6. Concordo totalmente
7. Concordo totalmente
8. Concordo totalmente

***Domain Driven Design* (DDD)**
9. Concordo totalmente
10. Concordo totalmente
11. Concordo totalmente
12. Concordo

Segregação de Banco de Dados

13. Concordo totalmente
14. Indiferente
15. Concordo
16. Concordo

Perfil da Equipe para realizar a migração

18. Concordo
19. Concordo
20. Concordo

Por Onde Começar a migração

21. Concordo totalmente
22. Concordo totalmente
23. Concordo totalmente
24. Concordo totalmente

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Concordo totalmente

Infraestrutura

28. Concordo totalmente
29. Concordo totalmente
30. Concordo totalmente
31. Concordo totalmente

Comentários finais

Deve-se adotar microsserviços somente quando demanda de acessos as funcionalidades dos domínios tiver previsão de crescer muito a médio e curto prazo, comprometendo assim a disponibilidade dos acessos.

Respondente #4

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Sim.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Fizemos recente uma migração no início do ano de 2022 e o que mais nos motivou na migração foi a segurança seguido de custo pois não precisamos nos preocupar com a depreciação dos servidores sem falar que podemos fazer upgrades com maior facilidade também com a disponibilidade do sistema o que agora com o home office o sistema em nuvem facilitará muito o acesso ao ERP.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Custo da energia gasta no data center local, Custo com ar condicionado, Custo dos links de internet, Custo com Segurança, Depreciação dos equipamentos.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Melhoria no desempenho da Aplicação e Banco, Disponibilidade do Sistema.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Melhoria no desempenho da aplicação e banco de dados.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo totalmente
6. Concordo totalmente
7. Concordo totalmente
8. Concordo totalmente

***Domain Driven Design* (DDD)**

9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo totalmente
14. Concordo totalmente
15. Concordo
16. Concordo

Perfil da Equipe para realizar a migração

18. Indiferente
19. Concordo totalmente
20. Concordo totalmente

Por Onde Começar a migração

21. Concordo totalmente
22. Concordo totalmente
23. Indiferente
24. Concordo totalmente

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Concordo

Infraestrutura

28. Indiferente
29. Concordo totalmente
30. Concordo
31. Indiferente

Considerações finais

Respondente não deixou comentário.

Respondente #5

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Nunca participei, sempre ingressei em empresas que já trabalhavam com microsservicos.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Nunca participei, sempre ingressei em empresas que já trabalhavam com microsservicos.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Nunca participei, sempre ingressei em empresas que já trabalhavam com microsservicos.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Nunca participei, sempre ingressei em empresas que já trabalhavam com microsservicos.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?

Resposta: Nunca participei, sempre ingressei em empresas que já trabalhavam com microsservicos.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo
6. Concordo
7. Concordo
8. Concordo

***Domain Driven Design* (DDD)**

9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo
14. Concordo
15. Concordo
16. Concordo

Perfil da Equipe para realizar a migração

18. Concordo
19. Discordo
20. Concordo

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Concordo

Infraestrutura

28. Concordo
29. Concordo
30. Concordo
31. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #6

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Pode, mas precisa?

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Limite para escalar a aplicação.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Tempo de execução, consumo de recursos.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?

Resposta: Tempo de execução, consumo de recursos, aumento do throughput.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo totalmente
6. Concordo totalmente
7. Concordo totalmente
8. Concordo totalmente

***Domain Driven Design* (DDD)**
9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados
13. Concordo totalmente
14. Concordo totalmente
15. Concordo
16. Concordo totalmente

Perfil da Equipe para realizar a migração
18. Concordo
19. Concordo
20. Discordo

Por Onde Começar a migração
21. Indiferente
22. Indiferente

23. Concordo totalmente
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo totalmente
26. Concordo
27. Concordo

Infraestrutura

28. Concordo totalmente
29. Concordo totalmente
30. Concordo totalmente
31. Concordo totalmente

Considerações finais

Respondente não deixou comentário.

Respondente #7

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Sim.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Facilidade de manutenção e performance individual de cada contexto.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Tempo de resposta a requisições e utilização de recursos como CPU, memória.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Tempo de resposta a requisições e utilização de recursos como CPU, memória.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo
6. Concordo
7. Concordo totalmente
8. Concordo

***Domain Driven Design* (DDD)**
9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados
13. Concordo
14. Concordo
15. Discordo
16. Discordo

Perfil da Equipe para realizar a migração
18. Discordo
19. Concordo
20. Discordo totalmente

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo
24. Concordo

UML para identificar os microsserviços candidatos

25. Indiferente
26. Indiferente
27. Discordo

Infraestrutura

28. Indiferente
29. Concordo
30. Concordo
31. Indiferente

Considerações finais

Respondente não deixou comentário.

Respondente #8

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Acredito que pode sim, porém inicialmente não vejo problema em desenvolver um sistema monolítico.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Não tive experiência com migração, somente trabalhei com as duas arquiteturas em momentos diferentes.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Não tive experiência com migração, somente trabalhei com as duas arquiteturas em momentos diferentes.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Não tive experiência com migração, somente trabalhei com as duas arquiteturas em momentos diferentes.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Não tive experiência com migração, somente trabalhei com as duas arquiteturas em momentos diferentes.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo totalmente
6. Concordo totalmente
7. Concordo totalmente
8. Concordo totalmente

***Domain Driven Design* (DDD)**
9. Concordo
10. Indiferente
11. Indiferente
12. Indiferente

Segregação de Banco de Dados
13. Concordo
14. Concordo totalmente
15. Concordo totalmente
16. Concordo totalmente

Perfil da Equipe para realizar a migração

18. Discordo
19. Concordo totalmente
20. Concordo

Por Onde Começar a migração

21. Concordo totalmente
22. Concordo totalmente
23. Concordo totalmente
24. Concordo totalmente

UML para identificar os microsserviços candidatos

25. Concordo totalmente
26. Concordo
27. Indiferente

Infraestrutura

28. Indiferente
29. Indiferente
30. Concordo
31. Concordo totalmente

Considerações finais

Usar um proxy reverso para comunicação entre as aplicações pode ser uma ótima abordagem também.

Respondente #9

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Concordo totalmente.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Melhoria de performance e de manutenção.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Impacto na plataforma para os consumidores, latência e escalabilidade.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Impacto na plataforma para os consumidores, latência e escalabilidade.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim, 100%.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo
6. Concordo
7. Concordo
8. Concordo

***Domain Driven Design* (DDD)**
9. Concordo totalmente
10. Concordo
11. Concordo totalmente
12. Concordo

Segregação de Banco de Dados

13. Concordo totalmente
14. Discordo
15. Indiferente
16. Discordo

Perfil da Equipe para realizar a migração

18. Concordo totalmente
19. Concordo totalmente
20. Concordo

Por Onde Começar a migração

21. Concordo totalmente
22. Concordo totalmente
23. Concordo totalmente
24. Concordo totalmente

UML para identificar os microsserviços candidatos

25. Indiferente
26. Concordo
27. Concordo totalmente

Infraestrutura

28. Concordo totalmente
29. Concordo totalmente
30. Concordo totalmente
31. Concordo totalmente

Considerações finais

Respondente não deixou comentário.

Respondente #10

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?

Resposta: Nem sempre é vantagem devido o porte do projeto, ou dos custos envolvidos, as vezes um monolítico bem escrito tem um resultado final com um custo menor, o que pode ser um grande fator para o financeiro dependendo do porte da empresa.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: A principal motivação foi a modernização para conseguir ter uma melhor performance e escalabilidade, devido um contrato com um alto trafego de dados.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Tempo médio de resposta, ping, limite de dados, quantidade de acessos simultâneos sem impacto relevante para a aplicação.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Tempo médio de resposta, quantidade de acessos simultâneos sem impacto relevante para a aplicação.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Foram alcançados de maneira gradativa, pois a migração completa demorou para ser concluída.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo
6. Concordo
7. Concordo
8. Concordo

***Domain Driven Design* (DDD)**

9. Concordo totalmente
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo
14. Concordo
15. Indiferente
16. Concordo

Perfil da Equipe para realizar a migração

18. Discordo
19. Concordo
20. Concordo

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo
24. Indiferente

UML para identificar os microsserviços candidatos

25. Indiferente
26. Indiferente
27. Indiferente

Infraestrutura

28. Concordo
29. Concordo

30. Concordo
31. Concordo

Considerações finais
Respondente não deixou comentário.

Respondente #11

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Concordo.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Concordo.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: .

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: .

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: .

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo
6. Concordo
7. Concordo
8. Concordo

***Domain Driven Design* (DDD)**

9. Indiferente
10. Concordo
11. Concordo
12. Indiferente

Segregação de Banco de Dados

13. Concordo totalmente
14. Concordo totalmente
15. Concordo
16. Concordo totalmente

Perfil da Equipe para realizar a migração

18. Discordo
19. Concordo
20. Concordo

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo totalmente
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo totalmente
26. Indiferente
27. Indiferente

Infraestrutura

28. Concordo
29. Concordo totalmente
30. Concordo totalmente
31. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #12

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: A abordagem de micro serviços deve ser muito bem fundamentada, haja vista que uma arquitetura distribuída é complexa para se implementar e manter. Deve se levar em consideração diversos fatores como forma de entrega/distribuição, tamanho da equipe, monitoração…

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Em geral desacoplamentos, descentralização, reuso, autonomia dos times.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Esforço para evolução vs impacto em outras features.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Agilidade nas entregas e única fonte da verdade por domínio.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Em geral, os objetivos passaram ser outros.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo totalmente
6. Concordo totalmente
7. Concordo totalmente
8. Concordo

***Domain Driven Design* (DDD)**

9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo
14. Concordo
15. Indiferente
16. Concordo

Perfil da Equipe para realizar a migração

18. Concordo
19. Concordo totalmente
20. Concordo

Por Onde Começar a migração

21. Concordo totalmente
22. Concordo
23. Concordo
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo
26. Indiferente
27. Concordo

Infraestrutura

28. Concordo
29. Concordo
30. Concordo
31. Indiferente

Considerações finais

Respondente não deixou comentário.

Respondente #13

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Com profissionais experientes no assunto, sim.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: “Decoupling” de serviços possibilitando flexibilidade no desenvolvimento e evolução do software, redução de custo operacional com escalabilidade granular.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: “Decoupling” de serviços possibilitando flexibilidade no desenvolvimento e evolução do software, redução de custo operacional com escalabilidade granular.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Custo operacional, escalabilidade, SLA.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo totalmente
6. Concordo
7. Indiferente
8. Concordo

***Domain Driven Design* (DDD)**

9. Concordo totalmente
10. Concordo totalmente
11. Concordo totalmente
12. Concordo totalmente

Segregação de Banco de Dados

13. Indiferente
14. Concordo totalmente
15. Indiferente
16. Concordo

Perfil da Equipe para realizar a migração

18. Discordo
19. Concordo totalmente
20. Indiferente

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Indiferente

Infraestrutura

28. Concordo
29. Discordo
30. Indiferente
31. Indiferente

Considerações finais

Respondente não deixou comentário.

Respondente #14

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?

Resposta: Depende, entrar em um mundo de microservices pode ser um tiro no pé, pois isso envolve não só a maturidade do time de desenvolvimento, mas tbm a equipe de ops que é jogado em nível hard. Acredito que ir para o mundo de microservices deve ser muito bem avaliado, se você tiver uma equipe capaz de suportar a demanda em cima da arquitetura distribuída e tiver conhecimento suficiente sobre os objetivos do negócio “TALVEZ” microservices faça sentido do contrário vá de monolito que nunca será uma má opção.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Escala de time e escala de aplicação. Manter monolitos grandes na nuvem gera custo, as vezes precisamos escalar a solução como um todo por causa de uma pequena parte do sistema em microservices escalamos o serviço no qual desejamos de fato performance. Quanto ao time chega um momento que colocar 2 ou 10 devs atuando no mesmo artefato não trará grandes benefícios devido ao acoplamento do código.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Produtividade do time e custo.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Produtividade do time e custo.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Parcialmente sim. O trabalho é constante.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Discordo
6. Discordo
7. Concordo
8. Discordo

***Domain Driven Design* (DDD)**
9. Concordo totalmente
10. Concordo totalmente

11. Concordo totalmente
12. Concordo totalmente

Segregação de Banco de Dados

13. Concordo
14. Discordo
15. Discordo
16. Concordo

Perfil da Equipe para realizar a migração

18. Concordo
19. Concordo
20. Concordo totalmente

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo totalmente
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Concordo

Infraestrutura

28. Concordo
29. Concordo
30. Concordo
31. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #15

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Concordo.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Alta disponibilidade, flexibilidade nas implantações e custos foram os fatores mais relevantes.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Manutenção.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Escalabilidade, Confiabilidade, adesão de novas tecnologias.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo totalmente
6. Concordo
7. Concordo
8. Concordo

***Domain Driven Design* (DDD)**

9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo
14. Concordo
15. Discordo
16. Indiferente

Perfil da Equipe para realizar a migração

18. Indiferente
19. Concordo
20. Concordo

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo
24. Concordo

UML para identificar os microsserviços candidatos

25. Indiferente
26. Indiferente
27. Indiferente

Infraestrutura

28. Concordo totalmente
29. Concordo totalmente
30. Concordo
31. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #16

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Concordo parcialmente, acredito que todo produto de software deve obter uma arquitetura evolutiva, um código bem escrito desde seu MVP monolítico. Quando chegar a complexidade necessária, esse tipo de arquitetura pode ser utilizada. Não acredito que um produto novo deva nascer como Microsserviços.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Partes do projeto passou a necessitar de diferentes recursos de máquina e disponibilidade, além de separação de equipe por frentes de negócios que evoluiam em ritmos diferentes.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Velocidade das entregas, Apdex, quantidade erros em produção.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: As mesmas.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Parcialmente, muitos dos objetivos tiveram bons resultados, porém durante o caminho foram encontrados desafios que estavam relacionados diretamente a senioridade da equipe. São necessários o conhecimento de diversos padrões assim como um bom domínio de

técnicas/abordagem e boas práticas de desenvolvimento de software, como por exemplo TDD, DDD, SOLID entre outros. Na época, durante o processo foi indentificado que o conhecimento não era o suficiente e trouxe grandes complexidades por isso.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Discordo
6. Concordo
7. Concordo
8. Concordo

***Domain Driven Design* (DDD)**

9. Concordo totalmente
10. Concordo totalmente
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo
14. Concordo
15. Concordo
16. Concordo

Perfil da Equipe para realizar a migração

18. Concordo
19. Concordo totalmente
20. Concordo

Por Onde Começar a migração

21. Discordo
22. Indiferente
23. Concordo
24. Indiferente

UML para identificar os microsserviços candidatos

25. Indiferente
26. Discordo
27. Concordo

Infraestrutura

28. Concordo totalmente
29. Concordo totalmente
30. Concordo
31. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #17

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação”.?
Resposta: Sim.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Escalabilidade e performance.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Custo e desempenho.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Custo e desempenho.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Concordo
6. Concordo
7. Concordo totalmente
8. Indiferente

***Domain Driven Design* (DDD)**

9. Concordo totalmente
10. Concordo totalmente
11. Concordo totalmente
12. Concordo

Segregação de Banco de Dados

13. Concordo totalmente
14. Concordo
15. Discordo
16. Discordo totalmente

Perfil da Equipe para realizar a migração

18. Concordo
19. Concordo
20. Concordo totalmente

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo
24. Concordo totalmente

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Concordo

Infraestrutura

28. Concordo totalmente
29. Concordo totalmente
30. Concordo totalmente
31. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #18

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Não é algo ligado ao tamanho da empresa, mas sim ao propósito direto do software. Questões ligadas a disponibilidade, performance e grupos de disponibilidade de serviços essenciais são fatores que ditam a necessidade ou não de usar microservicos.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Desacoplamento dos serviços. O objetivo foi tornar os serviços independentes em termos de responsabilidade, gestão e evolução.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Domínio de uso, personas e responsabilidades atreladas ao

serviço, ou seja, foi mapeado os limites de responsabilidade para aquele serviço.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Basicamente consumo de recursos computacionais e capacidade de escala.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Plenamente.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo totalmente
6. Concordo totalmente
7. Concordo totalmente
8. Concordo

***Domain Driven Design* (DDD)**
9. Concordo
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados
13. Concordo totalmente
14. Concordo
15. Indiferente
16. Concordo

Perfil da Equipe para realizar a migração

18. Indiferente
19. Concordo
20. Concordo

Por Onde Começar a migração

21. Concordo
22. Concordo
23. Concordo totalmente
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo
26. Indiferente
27. Concordo

Infraestrutura

28. Concordo
29. Concordo
30. Concordo
31. Concordo

Considerações finais

Respondente não deixou comentário.

Respondente #19

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Acho um pouco arriscado já partir para o Microserviço sem antes fazer uma análise da necessidade.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Agilidade dos deploys e a venda de serviço da plataforma de forma modularizada.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Tempo de paralização dos de todos serviços, mesmo quando a alteração era em algo relativamente simples. Número de deploys realizados com hotfix.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Agilidade nas entregas vs transparência da aplicação.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?

Resposta: Acredito que sim, não permaneci na empresa tempo suficiente após o projeto para ter essa informação.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo
6. Concordo
7. Concordo
8. Concordo

***Domain Driven Design* (DDD)**

9. Indiferente
10. Concordo
11. Concordo
12. Concordo

Segregação de Banco de Dados

13. Concordo
14. Concordo
15. Indiferente
16. Concordo

Perfil da Equipe para realizar a migração

18. Discordo
19. Concordo
20. Concordo

Por Onde Começar a migração

21. Indiferente
22. Concordo
23. Concordo
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo
26. Concordo
27. Concordo

Infraestrutura

28. Concordo
29. Concordo
30. Concordo
31. Concordo totalmente

Considerações finais

Respondente não deixou comentário.

Respondente #20

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Correra.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Performance, manutenção, separação de times mais especializados, boas práticas, escalabilidade.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Requisições por serviço, crescimento da aplicação ficando cada vez mais pesada e com pouca performance, mudança para cloud retirando do servidor onpremisse.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: Requisições por serviço, tempo de deploy, tempo de manutenção.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Sim.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)
5. Concordo totalmente
6. Concordo totalmente

7. Concordo totalmente
8. Concordo

***Domain Driven Design* (DDD)**

9. Concordo totalmente
10. Concordo totalmente
11. Concordo totalmente
12. Concordo totalmente

Segregação de Banco de Dados

13. Concordo totalmente
14. Concordo
15. Discordo
16. Concordo

Perfil da Equipe para realizar a migração

18. Concordo
19. Concordo totalmente
20. Discordo totalmente

Por Onde Começar a migração

21. Concordo totalmente
22. Concordo totalmente
23. Concordo totalmente
24. Concordo totalmente

UML para identificar os microsserviços candidatos

25. Concordo totalmente
26. Concordo totalmente
27. Concordo totalmente

Infraestrutura

28. Concordo
29. Concordo totalmente
30. Concordo totalmente
31. Concordo totalmente

Considerações finais

Respondente não deixou comentário.

Respondente #21

Questão aberta: Qual sua opinião sobre a afirmação: “Qualquer empresa que desenvolvendo produto (software) independente do seu tamanho pode implantar Microsserviços em nuvem em sua operação.”?
Resposta: Discordo.

Questão aberta: 1. Caso já tenha participado de alguma migração de sistema com arquitetura monolítica para arquitetura de Microsserviços em nuvem, o que motivou a empresa a realizar essa migração?
Resposta: Custos quanto a escalabilidade.

Questão aberta: 2. Antes da migração, quais as informações/métricas foram consideradas?
Resposta: Tempo de uso de CPU, uso de disco, tempo de resposta de requisições.

Questão aberta: 3. Quais informações/métricas foram consideradas após a migração?
Resposta: As mesmas + grafo de dependências entre os serviços.

Questão aberta: 4. Os objetivos que motivaram a migração foram alcançados em sua totalidade?
Resposta: Não.

Questões fechadas (escala Likert)

DevOps (*Continuous Integration/Continuous Delivery*)

5. Discordo
6. Concordo totalmente
7. Concordo totalmente
8. Concordo totalmente

***Domain Driven Design* (DDD)**

9. Concordo totalmente
10. Indiferente
11. Discordo
12. Indiferente

Segregação de Banco de Dados

13. Indiferente
14. Concordo
15. Concordo
16. Discordo totalmente

Perfil da Equipe para realizar a migração

18. Concordo totalmente
19. Concordo totalmente
20. Concordo

Por Onde Começar a migração

21. Concordo
22. Concordo totalmente
23. Concordo
24. Concordo

UML para identificar os microsserviços candidatos

25. Concordo totalmente
26. Indiferente
27. Discordo totalmente

Infraestrutura

28. Indiferente
29. Concordo

30. Discordo

31. Discordo

Considerações finais

Quero conhecer o resultado quando divulgado.

APÊNDICE H – Método Delphi (Rodada 2)

Questionário publicado

Perfil do respondente:

• Função na organização.

• Quantos anos de experiência você possui em sua função?

• Quantos anos de experiência como arquiteto de software ou desenvolvedor de software você possui?

• Quantos anos de experiência atuando com o conceito de serviços você possui?

• Quantos anos de experiência atuando com Microsserviços você possui?

UML para identificar os microsserviços candidatos

1. Contextos limitados separados por casos de uso ou por diagrama de classes é um caminho para identificar os microsserviços candidatos (FRITZSCH et al., 2019).

2. Transformar o monolito em representação gráfica pode ajudar a identificar os serviços candidatos. No grafo, cada vértice representa a classe do monolito e as arestas não direcionadas representam seu acoplamento com outras classes do monolito. Ao cortar o grafo em componentes é possível observar os candidatos a microsserviços (SILVA; CARNEIRO; MONTEIRO, 2019).

3. Através de técnicas de engenharia reversa é possível criar artefatos de alto nível, como diagramas UML. Com esses diagramas é possível listar as funcionalidades para as quais o sistema foi projetado e isso ajudará a identificar os Microsserviços candidatos (WOLFART et al., 2021).

Perfil dos respondentes

Os tempos de experiência na Tabela 8 são exibidos em anos.

Tabela 8 – Perfil dos respondentes Delphi (Rodada 2)

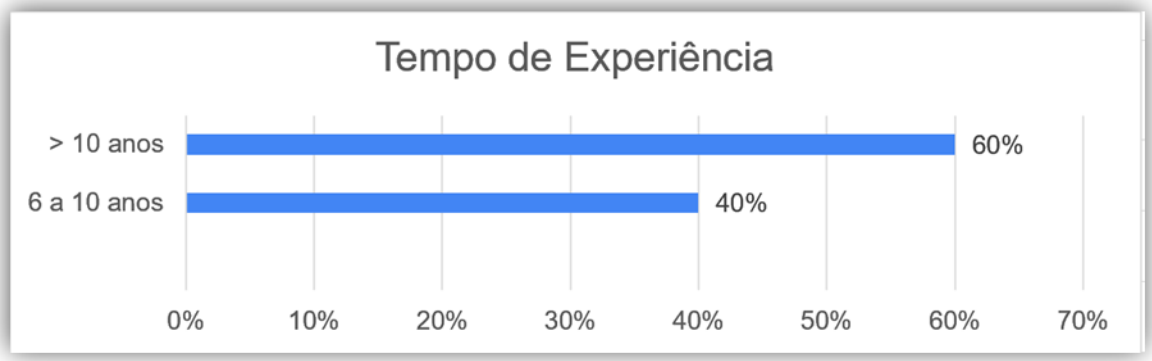
#	Resposta	Cargo	Tempo de experiência na área de desenvolvimento de *software*	Tempo de experiência atuando com orientação a serviços	Tempo de experiência atuando com microsserviços
1	17/04/2023 23:20	Arquiteto de solução	15	3	3
2	18/04/2023 07:09	Software Engineering	17	12	5
3	18/04/2023 07:36	Engenheiro de software	22	16	8
4	18/04/2023 09:44	Arquiteto	10	4	4
5	18/04/2023 20:09	Arquiteto de software	20	15	5
6	20/04/2023 15:41	Engineer Coordinator	10	8	2
7	21/04/2023 05:23	Engenheiro de software	6	5	5
8	21/04/2023 17:47	Arquiteto de software	9	7	3
9	22/04/2023 11:59	Engenheiro de software	15	6	5
10	23/04/2023 02:54	Desenvolvedor Java	12	2	1
11	24/04/2023 19:08	Engenheiro de software	24	15	10
12	25/04/2023 08:41	Desenvolvedor .NET	6	6	2
13	25/04/2023 20:53	Arquiteto de software	11	4	4

14	26/04/2023 16:18	Desenvolvedor Java	20	5	3
15	27/04/2023 08:02	Desenvolvedor .NET	6	6	2
16	28/04/2023 22:59	Analista desenvolvedor PL/SQL	10	10	8
17	29/04/2023 11:13	Desenvolvedor	11	6	3
18	29/04/2023 23:42	Engenheiro de software	12	4	4
19	30/04/2023 16:27	Desenvolvedor	10	5	2
20	04/05/2023 18:56	Engenheiro de software	15	10	8

Fonte: autor.

O gráfico da Figura 38 apresenta o tempo de experiência dos profissionais.

Figura 38 – Tempo de experiência dos especialistas Delphi (Rodada 2)



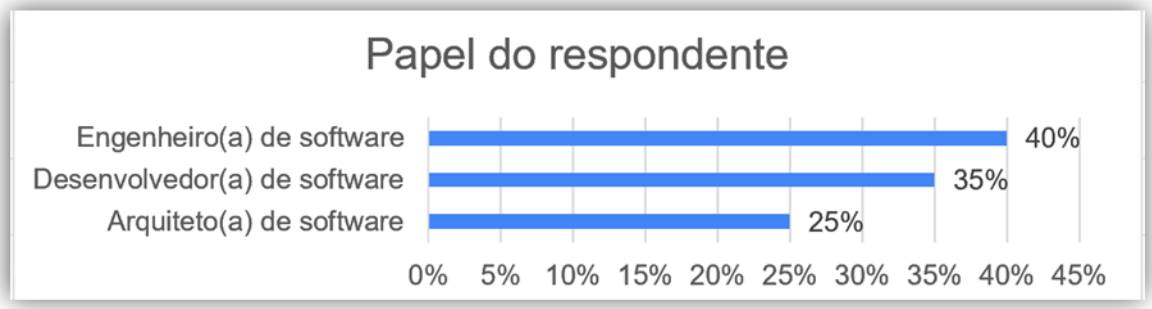
Fonte: autor.

Como apresenta o gráfico da Figura 38, 60% dos respondentes têm mais de 10 anos de experiência atuando no processo de desenvolvimento de *software*, e

40% entre 6 e 10 anos.

O gráfico da Figura 39 apresenta o papel dos respondentes dentro da área de desenvolvimento de *software*.

Figura 39 – Papel dos respondentes Delphi (Rodada 2)



Fonte: autor.

Como apresenta o gráfico da Figura 39, 40% dos respondentes são engenheiros(as) de *software*, 35% são desenvolvedores de *software* e 25% são arquitetos(as) de *software*.

Respostas dos respondentes

Respondente #1

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #2

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo

2. Indiferente
3. Concordo

Respondente #3

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #4

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo totalmente
2. Concordo totalmente
3. Concordo

Respondente #5

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Indiferente
3. Discordo

Respondente #6

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Indiferente

2. Concordo totalmente
3. Indiferente

Respondente #7

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #8

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #9

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Indiferente
2. Concordo
3. Concordo

Respondente #10

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Discordo

2. Concordo
3. Concordo

Respondente #11

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #12

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Discordo
2. Concordo
3. Indiferente

Respondente #13

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #14

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo totalmente

2. Concordo
3. Concordo

Respondente #15

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #16

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #17

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo

Respondente #18

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo

2. Concordo
3. Concordo

Respondente #19

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Discordo
2. Concordo
3. Concordo

Respondente #20

Questões fechadas (escala Likert)

UML para identificar os microsserviços candidatos

1. Concordo
2. Concordo
3. Concordo