

FACULDADE DE TECNOLOGIA DE SÃO PAULO

CONSTRUINDO CONEXÕES DIGITAIS:
ARQUITETURA DE SISTEMAS PARA UMA IMPLEMENTAÇÃO
OMNICHANNEL EFICIENTE

Enzo Giovanni Mendes de Souza

SÃO PAULO

2024

FACULDADE DE TECNOLOGIA DE SÃO PAULO
ENZO GIOVANNI MENDES DE SOUZA

CONSTRUINDO CONEXÕES DIGITAIS:
ARQUITETURA DE SISTEMAS PARA UMA IMPLEMENTAÇÃO
OMNICHANNEL EFICIENTE

Trabalho submetido como exigência parcial
para a obtenção do Grau de Tecnólogo em
Análise e Desenvolvimento de Sistemas
Orientador: Paulo Roberto Bernice.

SÃO PAULO
2024

Dedico este trabalho a mim mesmo, por ter tido a coragem de seguir meus sonhos e enfrentar os desafios ao longo do caminho. Agradeço por cada obstáculo superado, cada momento de dificuldade que me fortaleceu e cada oportunidade que me fez crescer. Isso é resultado de toda a jornada, das dores, amores e aprendizados que vivi. Que cada página dele seja uma celebração da minha dedicação e perseverança. Aos meus sonhos, que nunca deixei de perseguir.

Uma capacidade importante para as empresas prosperarem e crescerem, e até mesmo sobreviverem, é a capacidade de reagir e se adaptar às mudanças. Isso está relacionado às necessidades do cliente, à expansão para novos mercados ou à utilização do que há de melhor e mais recente em tecnologia.

(Anders Karlsen - Gerente de Produto Digital)

RESUMO

As interações digitais estão reformulando as estratégias de comunicação empresarial, impulsionadas pela rápida evolução tecnológica e pela crescente dependência da sociedade em dispositivos digitais. No cerne dessa mudança está a busca pela experiência omnichannel, que integra de maneira fluída e coesa os diversos canais de comunicação. Este estudo tem como objetivo analisar e propor uma arquitetura de sistemas eficiente para a implementação omnichannel, considerando desafios como integração de plataformas e coordenação de canais. A abordagem estratégica centrada no cliente é destacada como crucial para o sucesso, alinhando a arquitetura de sistemas aos objetivos empresariais e às expectativas do consumidor. A pesquisa adota uma metodologia qualitativa, combinando revisão bibliográfica e estudo de caso, para compreender as complexidades da arquitetura de sistemas omnichannel e suas implicações na comunicação empresarial digital. A revisão bibliográfica explora os conceitos de arquitetura de sistemas, omnichannel e comunicação empresarial, enquanto o estudo de caso analisa em detalhes a experiência de empresas que implementaram estratégias omnichannel com sucesso. A pesquisa contribui para a literatura sobre arquitetura de sistemas e omnichannel, oferecendo uma visão holística e estratégica para a implementação de estratégias omnichannel. O estudo também busca oferecer insights valiosos para as empresas que buscam aprimorar suas estratégias de comunicação empresarial e proporcionar uma experiência omnichannel de excelência para seus clientes.

Palavras-chave: Arquitetura de Sistemas; Omnichannel; Comunicação Empresarial; Experiência do Cliente; Integração de Plataformas.

ABSTRACT

Digital interactions are reshaping corporate communication strategies, driven by rapid technological advancements and society's increasing reliance on digital devices. At the heart of this transformation lies the pursuit of the omnichannel experience, seamlessly integrating various communication channels. This study aims to analyze and propose an efficient system architecture for omnichannel implementation, considering challenges such as platform integration and channel coordination. The customer-centric strategic approach is highlighted as crucial for success, aligning system architecture with business objectives and consumer expectations. The research adopts a qualitative methodology, combining literature review and case study, to understand the complexities of omnichannel system architecture and its implications for digital corporate communication. The literature review explores concepts of system architecture, omnichannel, and corporate communication, while the case study analyzes in detail the experiences of companies that have successfully implemented omnichannel strategies. The research contributes to the literature on system architecture and omnichannel, providing a holistic and strategic view for the implementation of omnichannel strategies. The study also seeks to offer valuable insights for companies looking to enhance their corporate communication strategies and provide an excellent omnichannel experience for their customers.

Keywords: Systems Architecture; Omnichannel; Business Communication; Customer Experience; Platform Integration.

SUMÁRIO

1 INTRODUÇÃO	8
1.1 MOTIVAÇÃO	8
1.2 JUSTIFICATIVA	8
1.3 OBJETIVOS DO TRABALHO.....	9
2 DESENVOLVIMENTO	10
2.1 OMNICHANNEL: DEFINIÇÃO E INTEGRAÇÃO AO DESENVOLVIMENTO.....	10
2.1.1 Aprofundando os aspectos técnicos do omnichannel	11
2.1.2 Tecnologias chave para o desenvolvimento de sistemas omnichannel:	11
2.1.3 Exemplos gerais de implementações omnichannel e seus desafios:	13
2.2 ARQUITETURA DE SISTEMAS PARA OMNICHANNEL.....	15
2.2.1 Visão Geral.....	15
2.2.2 A Relação entre Arquitetura de Sistemas e Código Limpo.....	16
2.2.3 Princípios de Código Limpo Aplicados ao Omnichannel	17
2.2.4 Benefícios do Código Limpo para Estratégias Omnichannel.....	19
2.2.5 Arquitetura Limpa para Sustentabilidade e Eficiência no Omnichannel	20
2.2.5.1 Fundamentos da Arquitetura Limpa	20
2.2.5.2 Integração com Estratégias Omnichannel.....	23
2.2.5.3 Benefícios da Arquitetura Limpa para o Omnichannel	24
2.2.6 Desafios, Estratégias e Considerações na Implementação Prática do Omnichannel	24
2.2.6.1 Arquitetura Orientada a Eventos (Event-Driven Architecture)	27
2.2.6.2 Padrões de Tolerância a Falhas.....	28
2.2.6.3 Escalabilidade do Banco de Dados com Sharding.....	30
2.2.6.4 Garantia de Ordem e Escalabilidade com Sistemas de Mensageria.....	32
2.2.6.5 A Chave para a Fluidez da Experiência do Usuário	35
2.2.7 O Custo do Código Sujo.....	37
2.2.8 O Código Sujo Não Produz Com Mais Eficiência.....	38
2.3 A EVOLUÇÃO DA COMUNICAÇÃO EMPRESARIAL COM OMNICHANNEL....	44
2.4 TRANSFORMAÇÃO DA EXPERIÊNCIA DO CLIENTE ATRAVÉS DO OMNICHANNEL.....	45
3 CONCLUSÃO	49
REFERÊNCIAS.....	51

1 INTRODUÇÃO

1.1 MOTIVAÇÃO

A crescente digitalização da sociedade tem transformado profundamente a maneira como as organizações se comunicam e interagem com seus públicos-alvo. Nesse cenário, as estratégias de comunicação empresarial passam por uma significativa metamorfose para atender às demandas de um ambiente cada vez mais digitalizado. Um dos pilares dessa transformação é a busca por experiências omnichannel, que buscam integrar de forma coesa e fluida os diversos canais de comunicação utilizados por uma empresa. Seja através de redes sociais, websites, aplicativos móveis ou outros meios digitais, a busca por uma presença unificada e consistente torna-se crucial para satisfazer as expectativas do consumidor contemporâneo.

A evolução da arquitetura de sistemas desempenha um papel fundamental nesse processo de implementação eficiente do omnichannel. Sistemas robustos e flexíveis são imprescindíveis para suportar a complexidade inerente à integração de diversos canais de comunicação. Desafios como a integração de plataformas distintas, a gestão eficaz de dados e a adaptação rápida às mudanças no ambiente digital são enfrentados pelas organizações que buscam proporcionar uma experiência omnichannel de qualidade.

1.2 JUSTIFICATIVA

Para compreender plenamente esse contexto em constante evolução, é essencial considerar as mudanças nas expectativas do consumidor. Hoje, o público não apenas busca conveniência, mas também exige consistência em suas interações com as marcas, independentemente do canal escolhido. A competição acirrada entre as empresas acrescenta uma pressão constante para a inovação, levando-as a repensarem suas estratégias de comunicação e a investirem em sistemas que permitam a integração eficiente de canais

Neste sentido, autores como Verhoef et al. (2015) destacam a necessidade de uma abordagem centrada no cliente na construção de sistemas omnichannel eficazes. Isso implica compreender as necessidades e preferências dos clientes em relação aos canais de compra e comunicação, desenvolvendo estratégias que proporcionem uma experiência de compra integrada e consistente em todos os canais disponíveis. Além disso, a personalização da experiência de compra, baseada nas informações coletadas sobre o comportamento e as preferências do cliente, pode aumentar a satisfação do cliente, a fidelidade e as vendas.

Enquanto isso, Ross, Weill e Robertson (2006) ressaltam a importância da arquitetura empresarial como uma ferramenta para alinhar a estratégia de negócios com a arquitetura de sistemas, permitindo maior agilidade e flexibilidade na adaptação às mudanças do ambiente digital. Por outro lado, Verhoef, Kannan e Inman (2015) enfatizam a importância da coordenação de canais para a implementação de uma estratégia omnichannel eficiente, o que requer uma arquitetura de sistemas ágil e flexível para suportar a integração de diferentes canais de venda.

Dessa forma, é evidente que o cenário contemporâneo das interações digitais e das estratégias de comunicação empresarial está intrinsecamente ligado à busca por experiências omnichannel eficientes.

1.3 OBJETIVOS DO TRABALHO

Compreender as mudanças nas expectativas do consumidor e adotar abordagens estratégicas na arquitetura de sistemas são elementos cruciais para o sucesso nesse ambiente digital em constante evolução. Este trabalho visa aprofundar essa compreensão e contribuir para o avanço das práticas de comunicação empresarial omnichannel, analisando e propondo uma arquitetura de sistemas eficiente para sua implementação

2 DESENVOLVIMENTO

2.1 OMNICHANNEL: DEFINIÇÃO E INTEGRAÇÃO AO DESENVOLVIMENTO

Omnichannel é um conceito que reflete a estratégia de integração de diversos canais de comunicação utilizados por uma empresa, visando proporcionar uma experiência unificada e consistente para o consumidor em todos os pontos de contato. Essa abordagem vai além da simples presença em múltiplos canais, buscando sincronizar as interações do cliente independentemente do canal escolhido, seja ele físico ou digital. Essa integração abrange diferentes pontos de contato, como websites, lojas físicas, aplicativos móveis, redes sociais, televidas, e-mail, chatbots, entre outros. Não visa apenas uma simples integração de canais. Trata-se de uma filosofia que coloca o cliente no centro e utiliza dados e tecnologias para criar uma experiência coesa e personalizada em todos os pontos de contato. Isso significa que, independentemente de qual canal o cliente escolher, ele terá acesso às mesmas informações, produtos e serviços, e poderá realizar suas transações de forma rápida, fácil e eficiente.

No contexto do desenvolvimento de sistemas, o omnichannel representa um desafio e uma oportunidade. Por um lado, implica na necessidade de criar sistemas flexíveis e robustos capazes de integrar e gerenciar a comunicação em diferentes plataformas. Por outro lado, oferece a chance de desenvolver soluções inovadoras que potencializem a experiência do usuário e impulsionem os resultados do negócio.

A integração do omnichannel requer uma arquitetura de sistemas ágil e adaptável, capaz de suportar a complexidade de múltiplos canais de comunicação. Isso envolve a integração de sistemas existentes, a implementação de APIs (interfaces de programação de aplicativos) para facilitar a comunicação entre diferentes plataformas e a adoção de tecnologias que permitam a coleta e análise de dados em tempo real.

Além disso, o desenvolvimento de sistemas omnichannel deve levar em consideração a necessidade de personalização da experiência do usuário. Isso significa utilizar dados coletados em diversos pontos de contato para oferecer conteúdo e serviços adaptados às preferências e comportamentos individuais de cada cliente. Através da implementação eficaz do omnichannel, as empresas podem

não apenas atender às expectativas dos consumidores modernos, mas também diferenciar-se da concorrência e alcançar o sucesso no mercado digital em constante evolução.

2.1.1 Aprofundando os aspectos técnicos do omnichannel

A integração de canais é um dos pilares do omnichannel, permitindo que os clientes transitem entre diferentes pontos de contato sem interrupções. Isso envolve a integração de plataformas, APIs, bancos de dados e outros sistemas para garantir a comunicação fluida entre os canais.

O gerenciamento eficaz dos dados do cliente é fundamental para o sucesso do omnichannel. A centralização e a unificação dos dados coletados em diferentes canais permitem uma visão holística do cliente, possibilitando a personalização da experiência e a oferta de produtos e serviços relevantes.

A personalização da experiência do cliente é um diferencial crucial no mercado competitivo atual. O omnichannel permite que as empresas ofereçam conteúdo, ofertas e serviços personalizados de acordo com as necessidades, interesses e histórico de compras de cada cliente.

O mercado digital é dinâmico e exige constante adaptação. A arquitetura de sistemas omnichannel deve ser flexível e escalável para permitir a rápida integração de novos canais, a personalização de acordo com as necessidades específicas da empresa e a adaptação às mudanças no mercado e nas demandas dos clientes.

2.1.2 Tecnologias chave para o desenvolvimento de sistemas omnichannel:

A implementação de tecnologias adequadas é necessária no desenvolvimento de sistemas omnichannel para garantir a integração, agilidade e personalização necessárias para fornecer uma experiência fluida ao cliente. Nesta seção, discutiremos tecnologias essenciais, bem como outras que podem melhorar ainda mais a arquitetura omnichannel.

Por exemplo, as APIs facilitam a comunicação entre vários sistemas e plataformas e são essenciais para a integração desses sistemas. Os desenvolvedores podem facilitar a troca de dados e funções entre vários canais

usando uma abordagem baseada em APIs. Isso garante uma experiência consistente para o usuário final.

A arquitetura de microserviços tem se mostrado uma estratégia útil para criar sistemas omnichannel. Os microserviços facilitam a escalabilidade, manutenção e evolução contínua do sistema por dividir a aplicação em partes independentes e altamente especializadas. Além disso, permitem que as equipes de desenvolvimento trabalhem de forma mais ágil e iterativa, o que permite que elas se adaptem rapidamente às demandas do mercado. Podemos adotar três abordagens para a separação de um sistema em microserviços, visando a melhoria na escalabilidade, manutenibilidade e gestão de equipes:

1. Consiste na criação de microserviços de acordo com as equipes de desenvolvimento. Esta estratégia se justifica pela dificuldade de gerenciar múltiplas equipes trabalhando simultaneamente em um mesmo código base. Em um contexto de loja online, por exemplo, a equipe X poderia ser responsável pelos microserviços de compras, vendas e pagamentos, enquanto a equipe Y cuidaria dos microserviços de clientes, produtos e estoques. Essa abordagem promove a autonomia das equipes e facilita a manutenção e evolução do sistema.
2. Criação de microserviços baseados nas diferentes áreas de atuação do negócio. No exemplo da loja online, cada funcionalidade principal (compras, pagamentos, vendas) seria desenvolvida como um microserviço separado. Esta metodologia é apreciada pela organização que proporciona, facilitando a manutenção e a evolução independentes de cada área do sistema.
3. Inspirada nos princípios do Domain-Driven Design (DDD), onde um domínio é dividido em subdomínios menores. Neste caso, os módulos de compra, pagamento e vendas seriam agrupados em um único microserviço chamado financeiro. Esta técnica é particularmente útil para sistemas complexos, onde diferentes partes do sistema são agrupadas com base na lógica de negócio compartilhada.

Para suportar a complexidade e dinamismo do ambiente omnichannel, a infraestrutura de TI baseada em nuvem oferece recursos sob demanda, escalabilidade e flexibilidade. As empresas podem reduzir os custos operacionais,

aumentar a eficiência e garantir uma infraestrutura robusta e resiliente para suas operações omnichannel ao migrar para a nuvem.

O conhecimento do comportamento do usuário, a identificação de padrões e tendências e a personalização da experiência do cliente com base nas preferências e necessidades do cliente depende da análise de dados coletados em vários pontos de contato do cliente. O processamento e a análise de grandes quantidades de dados em tempo real são possíveis graças às tecnologias de big data, que permitem que as empresas façam escolhas estratégicas inteligentes para otimizar suas operações omnichannel.

A preocupação com a segurança cibernética cresce como resultado da crescente complexidade e interconexão dos sistemas omnichannel. A proteção dos dados do cliente e prevenção de violações de segurança depende de tecnologias de segurança avançadas, como análise comportamental, criptografia de ponta a ponta e autenticação biométrica.

As empresas também devem ser proativas em relação à segurança cibernética, implementando políticas rígidas de governança de dados e realizando auditorias regulares para descobrir e corrigir possíveis vulnerabilidades. As empresas podem oferecer uma experiência do cliente verdadeiramente integrada, personalizada e segura, diferenciando-se da concorrência e impulsionando o sucesso a longo prazo ao incorporar essas tecnologias na arquitetura omnichannel.

2.1.3 Exemplos gerais de implementações omnichannel e seus desafios:

A implementação de sistemas omnichannel é um passo crucial para empresas que buscam oferecer uma experiência integrada e personalizada aos clientes. Nesse sentido, exemplos práticos de implementações omnichannel em diferentes setores, como varejo, serviços financeiros e turismo, destacam os benefícios e desafios inerentes a essa abordagem.

No varejo, por exemplo, é comum que os clientes pesquisem produtos online, reservem-nos em lojas físicas e finalizem suas compras por meio de aplicativos móveis. Esse processo integrado visa proporcionar conveniência ao consumidor, mas enfrenta desafios significativos. A integração de sistemas legados com novas

tecnologias é uma tarefa complexa, pois exige a harmonização de diferentes plataformas e infraestruturas tecnológicas.

Da mesma forma, no setor de serviços financeiros, os clientes podem realizar transações bancárias por aplicativos, obter suporte de chatbots e agendar atendimentos presenciais. No entanto, garantir a segurança dos dados do cliente é essencial nesse contexto, demandando a implementação de medidas robustas de proteção cibernética.

No turismo, a integração de serviços online, como pesquisa de destinos, reserva de voos e hotéis, e recebimento de informações sobre atividades, destaca a necessidade de sistemas flexíveis e adaptáveis. A complexidade reside na coordenação eficiente de diferentes plataformas e na garantia de uma experiência contínua ao cliente, independentemente do canal utilizado.

Entre os desafios comuns enfrentados na implementação de sistemas omnichannel, destacam-se a complexidade e o custo associados à integração de sistemas legados com novas tecnologias. Além disso, a proteção dos dados do cliente é uma preocupação primordial, exigindo investimentos em segurança da informação. O desenvolvimento e a implementação de sistemas omnichannel também podem ser onerosos, especialmente para pequenas e médias empresas, que enfrentam recursos limitados.

Ademais, a adoção do omnichannel requer uma mudança na cultura organizacional, com ênfase na colaboração entre diferentes departamentos e na centralização do cliente. Essa mudança cultural é essencial para garantir uma abordagem integrada e orientada para o cliente em todos os pontos de contato.

Em suma, o desenvolvimento de sistemas omnichannel robustos e flexíveis requer uma análise cuidadosa dos desafios mencionados, visando superá-los de forma eficaz e sustentável.

2.2 ARQUITETURA DE SISTEMAS PARA OMNICHANNEL

2.2.1 Visão Geral

Para compreender plenamente o papel da arquitetura de sistemas na implementação eficiente do omnichannel, é crucial analisar como essa abordagem tecnológica se integra às estratégias de comunicação empresarial contemporâneas.

A arquitetura de sistemas desempenha um papel central na garantia da integração e da sincronia dos diversos canais de comunicação utilizados por uma organização, possibilitando uma experiência unificada e consistente para o consumidor final.

Ao considerar as questões abordadas, que destacam a evolução da comunicação empresarial e a importância do omnichannel, torna-se evidente que a arquitetura de sistemas desempenha um papel crucial nesse contexto. A integração eficiente de canais de comunicação requer uma infraestrutura tecnológica sólida e flexível, capaz de suportar a troca de informações em tempo real e a personalização das interações com o cliente.

A arquitetura de sistemas, nesse sentido, atua como o alicerce sobre o qual as estratégias omnichannel são construídas. Ela proporciona a base técnica necessária para a integração de sistemas e aplicativos, permitindo que dados sejam compartilhados e processados de forma coesa em todos os pontos de contato com o cliente. Além disso, uma arquitetura bem projetada facilita a adaptação rápida e eficiente às mudanças no ambiente digital, garantindo que a experiência do cliente permaneça relevante e atualizada.

É importante ressaltar que a arquitetura de sistemas não é apenas uma questão de infraestrutura tecnológica, mas também está intrinsecamente ligada à estratégia de negócios da organização. Uma arquitetura bem alinhada com os objetivos e valores da empresa permite uma maior agilidade e flexibilidade na implementação de iniciativas omnichannel, facilitando a entrega de uma experiência de comunicação coesa e personalizada em todos os canais.

Portanto, ao analisar o papel da arquitetura de sistemas na implementação do omnichannel, é essencial considerar não apenas os aspectos técnicos, mas também a sua integração com as estratégias de comunicação e os objetivos de negócio da

organização. Somente assim é possível garantir uma abordagem holística e eficaz para a criação de experiências omnichannel que atendam às expectativas dos consumidores contemporâneos e impulsionem o sucesso empresarial.

2.2.2 A Relação entre Arquitetura de Sistemas e Código Limpo

Com base nos dados apresentados pela pesquisa recente realizada pelo GitHub sobre a experiência do desenvolvedor, ou DevX, podemos observar que a compreensão do código pelo programador está diretamente relacionada à sua produtividade. Os resultados revelam que programadores que entendem o código que estão trabalhando têm um aumento significativo de 42% em sua produtividade.

As questões levantadas na pesquisa são pertinentes para compreendermos o cenário atual da indústria de desenvolvimento de software. Ao analisar a frequência com que os programadores conseguem entender o código em que estão trabalhando, observamos uma média de 3,8 em uma escala de 1 a 5, onde 5 representa sempre e 1 representa nunca. Isso sugere que, em geral, os programadores têm um nível razoável de compreensão do código, mas ainda há espaço para melhorias.

Outro aspecto relevante é a relação entre débito técnico e a capacidade de completar o trabalho. A média das respostas para essa questão foi de 2,8, indicando que o débito técnico muitas vezes afeta a eficiência do trabalho dos programadores. Esse resultado ressalta a importância de lidar de forma proativa com o débito técnico para garantir uma maior fluidez no processo de desenvolvimento.

Avaliando a qualidade do código em que estão trabalhando, os programadores atribuíram uma média de 3,5 em uma escala de 1 a 5, indicando que há espaço para melhorias na qualidade do código. Essa percepção reflete a necessidade contínua de se esforçar para alcançar padrões mais elevados de qualidade de código, a fim de promover uma melhor experiência de desenvolvimento e, consequentemente, uma maior produtividade (Forsgren et al., 2023).

Os resultados dessas questões apontam para uma conclusão importante: programadores que compreendem o código com o qual estão trabalhando são 42% mais produtivos do que aqueles que não o compreendem. Isso ressalta a

importância de garantir que o código seja claro, organizado e bem documentado, para facilitar a compreensão e a manutenção futura.

Nesse sentido, a relação entre arquitetura de sistemas e código limpo é essencial. Enquanto a arquitetura define a estrutura e organização dos componentes de software, o código limpo é responsável por implementar essa arquitetura de forma clara e eficiente. Como mencionado por Martin (2009), um código limpo é fácil de entender, modificar e manter ao longo do tempo, o que é fundamental para o sucesso de qualquer iniciativa omnichannel.

A arquitetura de sistemas desempenha um papel crucial na implementação eficiente do omnichannel, garantindo a integração e a sincronia dos diversos canais de comunicação. No entanto, a qualidade do código que sustenta essa arquitetura é igualmente importante. Um código limpo não apenas facilita a manutenção e a escalabilidade do sistema, mas também contribui para a coesão e a consistência da experiência do cliente.

Portanto, garantir que o código seja limpo não é apenas uma questão de eficiência, mas também de saúde na programação. Códigos sujos podem dificultar o trabalho dos programadores e comprometer a qualidade e a integridade do sistema como um todo. Assim, investir na escrita de código limpo é essencial para garantir a eficiência e o sucesso a longo prazo de qualquer projeto de desenvolvimento de sistemas.

2.2.3 Princípios de Código Limpo Aplicados ao Omnichannel

Conforme delineado por Martin (2009), código limpo é aquele que implica em práticas de codificação que promovem a clareza, a simplicidade e a eficiência na escrita de software. A aplicação dos princípios de Código Limpo, é fundamental para garantir a qualidade e a sustentabilidade do sistema omnichannel.

Aplicando essa abordagem, buscamos promover diversos benefícios. Um deles é a melhoria da manutenibilidade: um código limpo facilita a identificação e correção de bugs, reduzindo o tempo necessário para realizar atualizações e melhorias no sistema. Além disso, um código bem estruturado permite o aumento da escalabilidade do sistema, atendendo às demandas crescentes. A coesão da

experiência do cliente também é beneficiada, uma vez que um código limpo contribui para uma experiência mais coesa e consistente em todos os pontos de contato.

Alguns exemplos de como os princípios do Código Limpo podem ser aplicados ao desenvolvimento omnichannel incluem:

Ao desenvolver aplicativos ou sistemas omnichannel, é importante usar nomes de variáveis, funções e componentes que reflitam claramente sua finalidade e função em todos os canais. Por exemplo, ao criar um componente de carrinho de compras, é melhor nomeá-lo de forma descritiva, como "ShoppingCartComponent", em vez de algo genérico como "Componente1".

Divida a lógica de negócios em funções pequenas e específicas, o que facilita a compreensão e a manutenção do código em diferentes canais. Por exemplo, ao processar um pedido, divida as etapas em funções separadas, como "processPayment", "updateInventory" etc. Separação de preocupações entre os diferentes componentes do sistema buscando facilitar a modificação de partes do sistema sem afetar outras partes, um fator crucial em uma abordagem omnichannel para manter a consistência. Sempre promovendo a implementação de funções modulares e reutilizáveis, pois a reutilização de código, reduz a redundância e melhora a eficiência geral do código.

Ao lidar com erros, certifique-se de que a lógica de tratamento de erros seja clara e consistente em todos os canais. Use exceções de forma apropriada e forneça mensagens de erro informativas para orientar os usuários em caso de problemas.

Mantenha uma formatação consistente em todo o código, independentemente do canal de entrega. Isso facilita a colaboração entre equipes e garante uma experiência uniforme para os usuários, independentemente do dispositivo ou plataforma que estão usando.

Certifique-se de que todos os canais de entrega sejam testados adequadamente para garantir que a funcionalidade seja consistente e confiável em diferentes ambientes. Isso inclui testes de unidade, testes de integração e testes de aceitação em todos os canais suportados.

Utilização de padrões de design adequados, pois padrões testados asseguram uma base sólida para o código, tornando-o mais confiável e fácil de manter. Realização de testes unitários e de integração para garantir a qualidade do

código, visto que os testes são vitais para detectar problemas precocemente e garantir que as mudanças não quebrem funcionalidades existentes.

Documente adequadamente o código para garantir que outros membros da equipe possam entender facilmente a lógica de negócios e as decisões de design, independentemente do canal de entrega. Isso é especialmente importante em um ambiente omnichannel, onde diferentes equipes podem estar trabalhando em partes diferentes do sistema.

Utilização de interfaces claras e consistentes para interação com o sistema garantindo que todas as interações do usuário sejam legíveis e consistentes, independentemente do canal.

2.2.4 Benefícios do Código Limpo para Estratégias Omnichannel

Os benefícios do código limpo para estratégias omnichannel são diversos e impactam diretamente o sucesso da iniciativa. Por exemplo, um código limpo é mais fácil e rápido de manter, reduzindo os custos associados à correção de bugs e implementação de novos recursos. Além disso, com um código limpo e bem estruturado, as equipes de desenvolvimento podem responder mais rapidamente às mudanças nas demandas do mercado e nas preferências dos clientes, garantindo uma adaptação ágil e eficiente do omnichannel.

Adicionalmente, um código limpo facilita a integração de novos canais de comunicação ao omnichannel, permitindo que a organização expanda sua presença digital de forma rápida e eficiente. Isso se traduz em uma maior flexibilidade para acompanhar as tendências e necessidades do mercado, mantendo a empresa competitiva e relevante.

Além disso, contribui para uma experiência do cliente mais consistente e livre de erros, garantindo que cada interação com a marca seja positiva e memorável. Isso fortalece o relacionamento com o cliente e aumenta a fidelidade, fundamentais para o sucesso a longo prazo no ambiente empresarial atual.

Por fim, investir na criação de um código limpo promove uma cultura de excelência técnica dentro da equipe de desenvolvimento, incentivando a colaboração, a inovação e o aprendizado contínuo. Essa abordagem não apenas

melhora a qualidade do produto, mas também fortalece o time e aumenta sua capacidade de enfrentar desafios complexos de forma eficaz.

Em suma, esses benefícios demonstram claramente o valor de priorizar a qualidade do código no contexto do omnichannel, destacando como um código limpo pode impulsionar o sucesso das estratégias de comunicação empresarial contemporâneas. Por tanto, a qualidade do código tem um impacto direto na manutenibilidade e na escalabilidade do sistema omnichannel, facilita a identificação e correção de bugs, o que reduz o tempo necessário para realizar atualizações e melhorias no sistema. Além disso, um código bem estruturado e documentado permite que novos desenvolvedores se integrem mais rapidamente ao projeto, garantindo a continuidade do desenvolvimento.

2.2.5 Arquitetura Limpa para Sustentabilidade e Eficiência no Omnichannel

A implementação eficiente do omnichannel requer uma abordagem de desenvolvimento de sistemas que vai além da funcionalidade imediata. Em vez disso, deve haver uma consideração da sustentabilidade do sistema a longo prazo e da sua capacidade de evoluir e adaptar-se às mudanças no ambiente de negócios e às necessidades dos clientes.

O propósito primário da arquitetura é suportar o ciclo de vida do sistema. Uma boa arquitetura torna o sistema fácil de entender, fácil de desenvolver, fácil de manter e fácil de implantar. O objetivo final é minimizar o custo da vida útil do sistema e não maximizar a produtividade do programador. (Martin, 2019)

Nesse sentido, a Arquitetura Limpa é vista como uma metodologia crítica que pode permitir a qualidade e a capacidade de manutenção do código – pré-requisitos para o desenvolvimento omnichannel bem-sucedido.

2.2.5.1 Fundamentos da Arquitetura Limpa

A Arquitetura Limpa se baseia em princípios fundamentais que visam promover a clareza, a modularidade e a escalabilidade do código-fonte. Ao separar claramente as diferentes responsabilidades do sistema e manter baixo o

acoplamento entre os componentes, a Arquitetura Limpa facilita a compreensão e a modificação do código, mesmo em contextos complexos como o omnichannel. Além disso, ao adotar padrões de design consistentes e bem definidos, a Arquitetura Limpa promove a consistência e a coesão do sistema, garantindo uma experiência de comunicação unificada para o cliente.

A arquitetura de sistemas para omnichannel envolve a estruturação em camadas de domínio que são fundamentais para garantir a organização e eficiência do sistema. A Arquitetura Limpa, proposta por Martin (2019), enfatiza a separação de preocupações e a manutenção da independência de frameworks e detalhes de implementação. Seus princípios, como a divisão em camadas (Interface do Usuário, Regras de Negócio, Adaptadores Externos), ajudam a manter o sistema organizado e adaptável. A separação de conceitos, como Entidades, Casos de Uso e Interfaces de Usuário, pode ser especialmente benéfica em um contexto omnichannel, onde diferentes canais podem ter requisitos e comportamentos distintos.

O Domain-Driven Design (DDD) concentra-se na modelagem do domínio do problema e na comunicação entre especialistas do domínio e desenvolvedores, garantindo que o software reflita com precisão as nuances e complexidades do negócio. O conceito de bounded contexts (contextos delimitados) pode ser útil em um ambiente omnichannel, onde diferentes canais podem representar diferentes contextos de negócios. A identificação e o modelamento de agregados, entidades, objetos de valor e serviços do domínio podem ajudar a estruturar o sistema de forma coesa e eficaz.

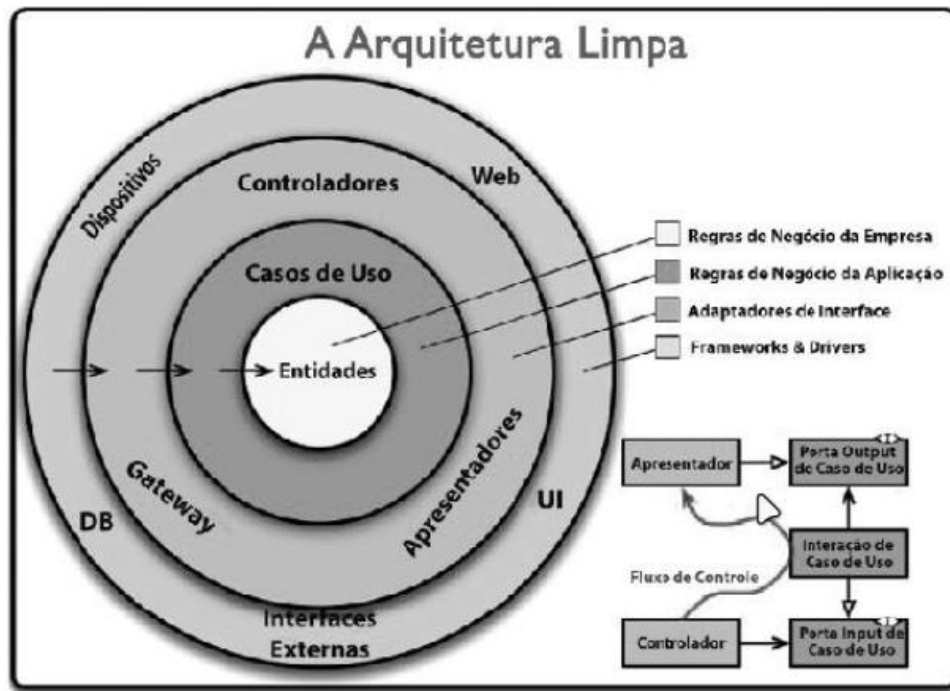
A Arquitetura Limpa fornece a estrutura geral e os princípios para a organização do sistema, enquanto o DDD orienta a modelagem e a implementação do domínio.

Podem ser aplicados os princípios da Arquitetura Limpa para organizar as diferentes camadas do sistema, mantendo a separação de conceitos e a independência de detalhes de implementação. Enquanto isso, o DDD pode ser usado para identificar os bounded contexts, modelar os domínios específicos de cada canal e definir as relações entre as entidades e objetos de valor.

A combinação de Design Patterns da Arquitetura Limpa com o Domain-Driven Design pode proporcionar uma base sólida para o desenvolvimento de sistemas omnichannel, permitindo a adaptação às complexidades do negócio e dos diferentes canais de interação com os clientes. Seguindo as diretrizes da Arquitetura Limpa,

adaptada ao contexto do Domain-Driven Design (DDD), podemos identificar e descrever essas camadas de forma detalhada como demonstra o diagrama 1.

Diagrama 1 - Camadas da arquitetura limpa



Fonte: Martin, 2019, p.209.

Na camada de Entidades (Domínio), concentram-se as entidades do sistema, que representam conceitos do domínio do negócio. Aqui residem as regras de negócio mais fundamentais e essenciais para o funcionamento do sistema. Por exemplo, em um sistema de comércio eletrônico, poderíamos ter entidades como "Produto", "Cliente" e "Pedido", cada uma com suas respectivas regras de validação e comportamento específico. É importante ressaltar que nesta camada o código deve ser o mais puro possível, livre de dependências externas e frameworks. Qualquer lógica relacionada ao domínio deve ser implementada aqui, sem contaminação por questões de infraestrutura.

A camada de Casos de Uso (Use Cases) abriga os casos de uso ou interações específicas do usuário com o sistema. Aqui são implementadas as regras de aplicação, que coordenam as ações entre as entidades do domínio para atender aos requisitos funcionais do sistema. Por exemplo, em um sistema de gerenciamento de pedidos, poderíamos ter casos de uso como "Realizar Pedido",

"Cancelar Pedido" e "Consultar Status do Pedido". Cada caso de uso encapsula a lógica necessária para coordenar as operações entre as entidades do domínio, sem se preocupar com detalhes de interface ou infraestrutura.

Nesta camada de Adaptadores de Interface (Interface Adapters), são implementados os adaptadores que traduzem as informações entre os formatos utilizados pela aplicação e os formatos necessários para interação com o mundo externo. Isso inclui interfaces com o usuário, serviços externos, APIs, entre outros. Por exemplo, poderíamos ter adaptadores para interfaces de usuário web, interfaces de programação de aplicativos (APIs) e interfaces de integração com sistemas de pagamento externos. Esses adaptadores garantem a flexibilidade e a interoperabilidade do sistema, permitindo que ele se comunique com diferentes dispositivos e serviços.

Na camada mais externa (Frameworks e Drivers), encontram-se os frameworks e drivers que fornecem suporte para aspectos técnicos, como acesso a banco de dados, comunicação de rede e gerenciamento de recursos do sistema. Aqui são incorporadas as tecnologias e bibliotecas necessárias para implementar os requisitos não funcionais do sistema, como desempenho, escalabilidade e segurança. Por exemplo, poderíamos ter frameworks ORM (Object-Relational Mapping) para acesso a banco de dados, frameworks de autenticação e autorização, e drivers para comunicação com dispositivos de hardware específicos. Essa camada permite que o sistema interaja com o ambiente de execução de forma eficiente e confiável, sem comprometer a integridade das camadas internas.

2.2.5.2 Integração com Estratégias Omnichannel

No contexto do omnichannel, a Arquitetura Limpa desempenha um papel crucial na garantia da integração e sincronização dos diversos canais de comunicação. Ao estabelecer uma estrutura modular e bem definida, a Arquitetura Limpa facilita a reutilização de componentes e a integração de novos canais, permitindo que a organização expanda sua presença digital de forma ágil e eficiente. Além disso, ao promover a separação de preocupações e a escalabilidade do sistema, a Arquitetura Limpa permite uma adaptação ágil do omnichannel às mudanças no ambiente de negócios e às preferências dos clientes.

2.2.5.3 Benefícios da Arquitetura Limpa para o Omnichannel

Os benefícios da Arquitetura Limpa para estratégias omnichannel são diversos e impactam diretamente o sucesso da iniciativa. Em primeiro lugar, um código limpo e bem estruturado facilita a manutenção do sistema, reduzindo o tempo necessário para correção de bugs e implementação de novos recursos. Isso garante uma resposta ágil do omnichannel às mudanças no ambiente de negócios e às preferências dos clientes, mantendo a organização competitiva e relevante. Além disso, ao promover a reutilização de componentes e a escalabilidade do sistema, a Arquitetura Limpa facilita a expansão do omnichannel para novos canais de comunicação, garantindo uma presença digital abrangente e coesa.

A Arquitetura Limpa emerge como um elemento-chave na implementação bem-sucedida do omnichannel, proporcionando a base técnica necessária para garantir a integração e a sincronia dos diversos canais de comunicação. Ao adotar os princípios da Arquitetura Limpa, as organizações podem promover a qualidade e a manutenibilidade do código, garantindo uma resposta ágil e eficiente do omnichannel às demandas do mercado e às preferências dos clientes. Assim, investir na sustentabilidade do omnichannel por meio da Arquitetura Limpa não é apenas uma decisão técnica, mas também estratégica, que pode impulsionar o sucesso e a competitividade da organização a longo prazo.

2.2.6 Desafios, Estratégias e Considerações na Implementação Prática do Omnichannel

Nesta seção, iremos aprofundar a implementação prática do omnichannel sob a ótica dos desafios específicos enfrentados no desenvolvimento e integração de sistemas para suportar uma estratégia omnichannel, bem como considerações práticas e estratégias de desenvolvimento de sistemas para superá-los.

Um dos principais desafios no desenvolvimento de sistemas para o omnichannel é a integração de múltiplos sistemas e tecnologias heterogêneas. Muitas empresas operam com uma variedade de sistemas legados, cada um com sua própria arquitetura e lógica de negócios. Integrar esses sistemas para fornecer

uma experiência omnichannel coesa pode ser uma tarefa complexa devido à falta de interoperabilidade entre eles.

Além disso, garantir a consistência de dados e informações em todos os canais é essencial para proporcionar uma experiência unificada ao cliente. Isso requer o desenvolvimento de sistemas que possam sincronizar e compartilhar dados em tempo real, minimizando o risco de inconsistências ou duplicações de informações.

Outro desafio significativo é garantir a escalabilidade e a flexibilidade dos sistemas para lidar com picos de demanda e mudanças nas preferências dos clientes. Os sistemas omnichannel devem ser projetados para suportar um grande volume de transações e interações em diversos canais, ao mesmo tempo em que permitem a rápida introdução de novos recursos e funcionalidades para atender às necessidades em constante evolução dos clientes. Para superar esses desafios, é crucial adotar considerações e estratégias específicas no desenvolvimento de sistemas para o omnichannel. Uma abordagem orientada a serviços, baseada em arquiteturas de microserviços e APIs (Interfaces de Programação de Aplicativos), pode facilitar a integração de sistemas heterogêneos, permitindo uma comunicação eficiente entre os diferentes componentes da infraestrutura de TI.

Para alcançar uma entrega contínua de software sem interromper o acesso dos clientes, são empregadas diversas estratégias na área de desenvolvimento de sistemas. Uma dessas estratégias, amplamente adotada e reconhecida por sua eficácia, é a implantação azul-verde. Esta abordagem, pensando em arquitetura de sistemas para omnichannel, oferece uma solução elegante para a implementação de atualizações e correções de software de forma transparente aos usuários finais.

A implantação azul-verde, em sua essência, consiste na criação de dois ambientes operacionais distintos: o ambiente "azul" e o ambiente "verde". Enquanto os clientes continuam a utilizar ativamente o ambiente "azul", o ambiente "verde" permanece inativo, aguardando por atualizações e modificações. Quando surge a necessidade de implementar uma nova versão de software ou realizar correções, essas alterações são aplicadas ao ambiente "verde", mantendo o ambiente "azul" como referência operacional inalterada.

Após a conclusão das modificações e a estabilização do ambiente "verde", o tráfego de usuários é direcionado para este ambiente, tornando-o ativo, enquanto o ambiente "azul" é temporariamente desativado. Essa transição ocorre de maneira

imperceptível para os clientes, garantindo a continuidade do serviço sem qualquer interrupção perceptível.

A adoção desse método proporciona uma série de benefícios, destacando-se a eliminação de períodos de inatividade do sistema, a preservação da qualidade da experiência do usuário e a garantia de uma entrega contínua de software. Além disso, essa abordagem promove uma maior confiabilidade e estabilidade ao processo de atualização, mitigando possíveis impactos negativos sobre os usuários e otimizando a eficiência operacional das empresas no contexto omnichannel.

Dessa forma, a implantação azul-verde emerge como uma prática altamente relevante e benéfica no contexto acadêmico e profissional de desenvolvimento de sistemas, contribuindo significativamente para aprimorar a gestão e a entrega de software em ambientes omnichannel.

O uso de tecnologias de nuvem e contêineres pode fornecer escalabilidade e flexibilidade adicionais, permitindo que os sistemas se adaptem dinamicamente às demandas. Isso permite que as empresas dimensionem recursos conforme necessário para lidar com picos de demanda sazonais ou eventos inesperados.

Também é fundamental adotar práticas robustas de segurança cibernética e conformidade com regulamentações de privacidade, especialmente ao lidar com dados sensíveis do cliente. Isso inclui a implementação de criptografia, autenticação de usuários e auditorias regulares para garantir a integridade e a confidencialidade dos dados.

Além disso, a automação de processos pode ajudar a simplificar e otimizar as operações de suporte ao cliente, reduzindo o tempo de resposta e melhorando a eficiência geral do sistema. O uso de chatbots e assistentes virtuais pode fornecer suporte instantâneo aos clientes, enquanto a análise de dados em tempo real pode ajudar a identificar padrões e tendências para melhorar continuamente a experiência do cliente.

Em resumo, o desenvolvimento de sistemas para o omnichannel requer uma abordagem cuidadosamente planejada e executada, que leve em consideração os desafios específicos do ambiente omnichannel e adote as melhores práticas e estratégias de desenvolvimento de sistemas para superá-los. Ao fazer isso, as empresas podem criar uma infraestrutura de TI robusta e flexível que suporte uma experiência omnichannel excepcional e impulsione o sucesso do negócio.

2.2.6.1 Arquitetura Orientada a Eventos (*Event-Driven Architecture*)

Na arquitetura de microserviços, a comunicação entre os diferentes serviços é uma questão crucial que pode impactar diretamente a escalabilidade, a manutenibilidade e a confiabilidade do sistema como um todo. Nesta seção, abordaremos as melhores práticas para a comunicação entre microserviços, destacando a importância de escolher a abordagem adequada para cenários específicos, como o exemplo de comunicação entre um serviço de usuário e um serviço de envio de e-mails.

Ao projetar a comunicação entre microserviços, os desenvolvedores frequentemente se deparam com o dilema de escolher a abordagem mais adequada para garantir a eficiência e a resiliência do sistema. No exemplo fornecido, onde um serviço de usuário precisa acionar um serviço de envio de e-mails toda vez que um novo usuário é criado, surge a necessidade de uma comunicação eficaz e desacoplada entre esses dois serviços.

Entre as opções tradicionais de comunicação entre microserviços, destacam-se a comunicação síncrona através de requisições diretas e a comunicação assíncrona através de filas de mensagens. No entanto, ambas as abordagens apresentam suas limitações. No caso da comunicação síncrona, há um alto acoplamento entre os serviços, o que pode resultar em falhas em cascata caso um dos serviços esteja inacessível. Por outro lado, o uso de filas de mensagens resolve o problema de acoplamento, mas pode introduzir complexidade adicional, especialmente quando múltiplas ações precisam ser executadas em resposta a um evento.

Diante dessas limitações, a melhor opção para a comunicação entre microserviços é a adoção de um modelo baseado em eventos. Nesse modelo, os serviços comunicam entre si por meio da emissão e recebimento de eventos, permitindo uma comunicação desacoplada e assíncrona. No exemplo mencionado, sempre que um novo usuário é criado, o serviço de usuário emite um evento de "usuário criado", ao qual o serviço de envio de e-mails pode estar inscrito para receber e processar.

A comunicação por eventos apresenta várias vantagens em relação às abordagens tradicionais. Em primeiro lugar, ela elimina o acoplamento direto entre os serviços, garantindo que a falha de um serviço não afete diretamente o

funcionamento dos outros. Além disso, ela facilita a escalabilidade do sistema, permitindo que novos serviços sejam adicionados ou removidos sem impactar a comunicação entre os demais. Por fim, ela promove a modularidade e a flexibilidade do código, tornando o sistema mais fácil de entender, manter e evoluir ao longo do tempo.

Em suma, a comunicação por eventos, parte integrante da arquitetura orientada a eventos, emerge como a melhor prática para a comunicação entre microserviços. Ao adotar esse modelo, há uma garantia para uma comunicação eficiente, desacoplada e escalável entre os diferentes serviços, promovendo a resiliência e a qualidade do sistema como um todo. Assim, a escolha da abordagem correta para a comunicação entre microserviços é fundamental para o sucesso da arquitetura de microserviços e para o alcance dos objetivos de negócio da organização.

2.2.6.2 Padrões de Tolerância a Falhas

A resiliência e a alta disponibilidade são fundamentais na arquitetura de sistemas para garantir a continuidade dos serviços, especialmente em ambientes de alta demanda e criticidade, como os sistemas omnichannel.

Os modelos "ativo-ativo" e "ativo-passivo" são algumas das estratégias de tolerância a falhas mais utilizadas para atingir esses objetivos.

O modelo ativo-ativo caracteriza-se pela operação simultânea de dois servidores, ambos lidando com o tráfego em tempo real. Nesse arranjo, os dois servidores estão constantemente processando solicitações, o que permite a distribuição equilibrada da carga. Em caso de falha de um dos servidores, o outro automaticamente assume toda a carga, garantindo a continuidade do serviço sem interrupção perceptível para os usuários. Essa redundância simultânea melhora a disponibilidade e a confiabilidade do sistema, essencial para aplicações críticas que requerem alta disponibilidade. Além da tolerância a falhas, oferece escalabilidade horizontal, permitindo o acréscimo de novos servidores para lidar com o aumento da demanda (Balasubramanian et al., 2009)

1. Vantagens do Modelo Ativo-Ativo:

- Alta disponibilidade: ambos os servidores operam continuamente, proporcionando redundância.
- Desempenho otimizado: o tráfego é distribuído, evitando sobrecarga em um único ponto.
- Resposta rápida a falhas: a transferência de carga é imediata, minimizando o impacto da falha.

2. Desafios do Modelo Ativo-Ativo:

- Complexidade de implementação: requer mecanismos avançados de sincronização e balanceamento de carga.
- Custo elevado: necessita de recursos adicionais para manter dois servidores operacionais simultaneamente.

No modelo ativo-passivo, há uma distinção clara entre o servidor ativo e o passivo. O servidor ativo é responsável por todo o processamento de carga, enquanto o servidor passivo permanece em modo de espera. Em caso de falha do servidor ativo, o sistema de gerenciamento de tráfego, seja por meio de um API Gateway ou de um load balancer, redireciona automaticamente o tráfego para o servidor passivo. Esse modelo é particularmente útil em situações em que a duplicação da carga não é necessária constantemente, mas a disponibilidade do serviço deve ser assegurada em caso de falha (Balasubramanian et al., 2009).

1. Vantagens do Modelo Ativo-Ativo:

- Custo reduzido: apenas um servidor processa o tráfego em condições normais, economizando recursos.
- Simplicidade de implementação: menos complexidade em comparação ao modelo ativo-ativo.

2. Desafios do Modelo Ativo-Ativo:

- Tempo de resposta à falha: Pode haver um breve período de inatividade enquanto o servidor passivo assume a carga.
- Desempenho não otimizado: o servidor passivo não contribui para o processamento do tráfego até que seja necessário.

A escolha entre os modelos ativo-ativo e ativo-passivo depende das necessidades específicas do sistema e dos recursos disponíveis. Em um ambiente omnichannel, onde a experiência do usuário e a disponibilidade contínua são prioritárias, a arquitetura ativo-ativo pode oferecer uma solução mais robusta, apesar dos custos e da complexidade envolvidos. Por outro lado, o modelo ativo-passivo

pode ser uma opção viável para organizações que buscam balancear disponibilidade e custos operacionais.

Em ambos os casos, a implementação de um API Gateway ou um balanceador de carga é essencial para direcionar o tráfego de forma eficiente e assegurar que a transição entre os servidores ocorra de maneira suave e sem impacto perceptível para o usuário final. Estes componentes são responsáveis por monitorar a saúde dos servidores e redirecionar o tráfego conforme necessário, garantindo a resiliência do sistema.

Por fim, a resiliência de um sistema em um ambiente omnichannel depende não apenas da escolha de um dos modelos, mas também de uma arquitetura bem planejada e da utilização de tecnologias avançadas para monitoramento e gestão de tráfego. A implementação cuidadosa desses elementos pode resultar em um sistema capaz de fornecer uma experiência consistente e de alta qualidade para os usuários, mesmo diante de falhas e interrupções inesperadas.

2.2.6.3 Escalabilidade do Banco de Dados com Sharding

A escalabilidade de uma aplicação refere-se à capacidade de um sistema lidar com um aumento crescente de carga sem comprometer o desempenho. Quando se trata de banco de dados, especialmente em aplicações com um volume elevado de dados, como é o caso de uma base com milhões de usuários e múltiplos canais, torna-se essencial implementar estratégias que permitam a distribuição eficiente desses dados. Uma das técnicas mais eficazes nesse cenário é o sharding.

Sharding é uma técnica de particionamento de banco de dados que envolve a divisão dos dados em fragmentos menores, chamados shards. Cada shard é uma instância de banco de dados independente que armazena uma parte dos dados. Esta divisão pode ser baseada em diferentes critérios, como ordem alfabética ou localização geográfica, permitindo uma distribuição eficiente e balanceada das informações. Essa abordagem permite que a carga de trabalho seja compartilhada entre diferentes servidores, evitando gargalos e melhorando a performance geral do sistema.

Para ilustrar o funcionamento do sharding, considere uma base de dados com 20 milhões de usuários. Esta base pode ser dividida de diversas maneiras, como:

1. Divisão por Ordem Alfabética:

- Uma maneira de implementar sharding é dividir a base de dados de acordo com a ordem alfabética dos usuários.
 - Por exemplo, cada letra do alfabeto pode representar uma instância de banco de dados separada. Usuários cujos nomes começam com 'A' são armazenados na instância 'A', usuários com nomes começando com 'B' na instância 'B', e assim por diante.

2. Divisão por Continente:

- Outra abordagem é particionar os dados com base em critérios geográficos.
 - Cada continente pode ter sua própria instância de banco de dados, armazenando informações específicas daquela região. Isso não apenas facilita a gestão dos dados, mas também pode melhorar a latência e o desempenho ao reduzir a distância entre os usuários e o servidor.

Essa distribuição de dados proporciona diversos benefícios para a escalabilidade do sistema:

- Aumento da capacidade de armazenamento: com múltiplas instâncias, a capacidade total de armazenamento é ampliada, permitindo o crescimento da base de dados sem comprometer a performance.
- Melhora da performance de leitura e escrita: a carga de trabalho é distribuída entre as instâncias, reduzindo a concorrência e otimizando as operações de leitura e escrita.
- Alta disponibilidade: em caso de falha em uma instância, as demais continuam operando, garantindo a disponibilidade do sistema.
- Facilidade de manutenção: cada instância pode ser gerenciada de forma independente, facilitando a realização de backups, atualizações e outras tarefas de manutenção.

Para ambientes omnichannel, onde a interação contínua e consistente entre diferentes canais é crucial, a implementação de sharding deve ser feita com atenção às particularidades do sistema e um planejamento cuidadoso, considerando a natureza dos dados, os padrões de acesso e as necessidades de escalabilidade do

sistema. É fundamental definir uma estratégia de particionamento adequada, que garanta a distribuição equilibrada dos dados e evite a sobrecarga de determinadas instâncias.

Algumas estratégias incluem:

- Particionamento baseado no canal de origem: distribuir os dados conforme o canal de origem das interações pode ajudar a balancear a carga entre diferentes shards, facilitando a manutenção de uma performance consistente.
- Sincronização e Consistência: garantir que os dados sincronizados entre diferentes shards permaneçam consistentes é essencial para evitar discrepâncias e garantir uma experiência de usuário fluida.
- Monitoramento em Tempo Real: implementar sistemas de monitoramento em tempo real permite a detecção rápida de falhas e a redistribuição da carga conforme necessário, mantendo a alta disponibilidade e a responsividade do sistema.

Em suma, o sharding se apresenta como uma técnica poderosa para escalar bancos de dados em aplicações omnichannel, proporcionando alta performance, disponibilidade e capacidade de crescimento. No entanto, sua implementação exige um planejamento cuidadoso e a adoção de boas práticas de gerenciamento e monitoramento (Abdelhafiz, 2020, 2021).

2.2.6.4 Garantia de Ordem e Escalabilidade com Sistemas de Mensageria

A arquitetura de sistemas tradicional, baseada em múltiplos servidores independentes, pode apresentar desafios na gestão de recursos compartilhados e na sincronização de eventos concorrentes. Diante disso, a adoção de uma arquitetura orientada a eventos como mencionado anteriormente, com o uso de sistemas de mensageria como o Kafka ou o RabbitMQ, surge como uma alternativa promissora.

No cenário atual de desenvolvimento de sistemas, a arquitetura omnichannel emerge como uma estratégia crucial para proporcionar uma experiência integrada e consistente aos usuários através de múltiplos canais. Neste contexto, a eficiência e a confiabilidade na troca de informações entre diferentes componentes do sistema

são fundamentais. É aqui que os sistemas de mensageria desempenham um papel vital, oferecendo uma infraestrutura robusta para a comunicação assíncrona e escalável.

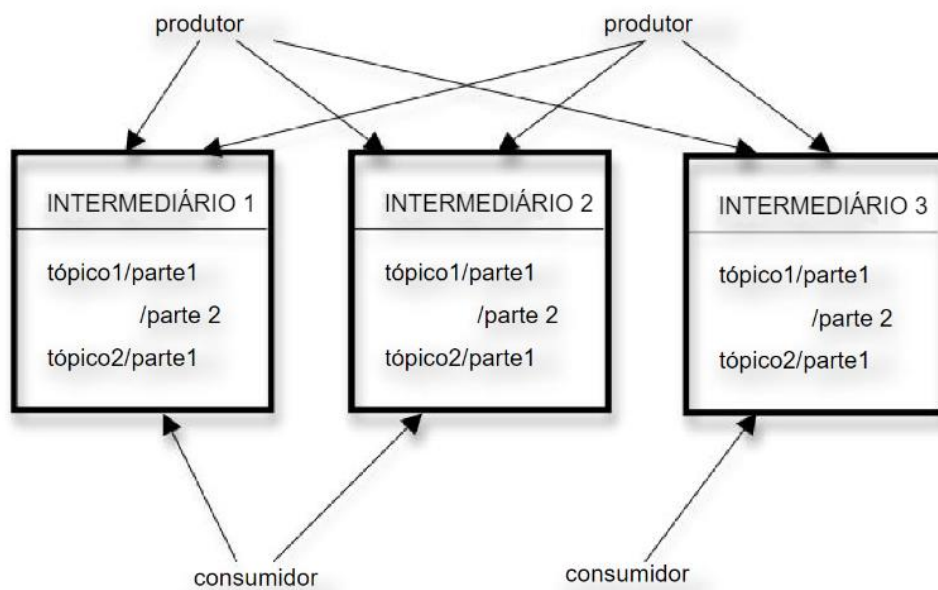
Os sistemas de mensageria são plataformas que facilitam a comunicação entre diferentes serviços e aplicações através da troca de mensagens. Estes sistemas permitem que as mensagens sejam enviadas e recebidas de forma assíncrona, garantindo que os serviços possam operar de maneira independente, sem a necessidade de estarem diretamente conectados. Isso é particularmente útil em arquiteturas e ambientes com alto volume de transações, como os sistemas omnichannel. Em essência, um sistema de mensageria funciona como um intermediário confiável na comunicação entre diferentes componentes de um sistema. Ele permite que os produtores de mensagens enviem dados para o sistema, que se encarrega de armazená-los e entregá-los aos consumidores de forma organizada e eficiente (Curry, 2005).

Existem diferentes modelos de comunicação em sistemas de mensageria, cada um com suas próprias características e casos de uso:

- Filas de Mensagens: utilizadas para armazenar mensagens que são processadas de acordo com a ordem de chegada (FIFO - First In, First Out). Exemplo: Amazon SQS (“Enfileiramento de mensagens totalmente gerenciado – Amazon Simple Queue Service – Amazon Web Services”, [s.d.]; T Sharvari; K Sowmya, 2019).
- Tópicos de Mensagens: permitem a publicação de mensagens para múltiplos consumidores, sendo uma escolha ideal para cenários onde a mesma mensagem precisa ser processada por vários serviços. Exemplo: Apache Kafka (“Apache Kafka”, [s.d.]; T Sharvari; K Sowmya, 2019).
- Modelo de Publicação/Assinatura: similar ao modelo de tópicos, mas com a possibilidade de filtrar as mensagens que serão entregues aos consumidores com base em critérios específicos. Exemplo: RabbitMQ (T Sharvari; K Sowmya, 2019).
- Mensageria de Ponto a Ponto: cada mensagem tem um único destinatário, garantindo a entrega única e exclusiva. Exemplo: RabbitMQ (“RabbitMQ: One broker to queue them all | RabbitMQ”, [s.d.]; T Sharvari; K Sowmya, 2019).

Um exemplo prático da aplicação de sistemas de mensageria pode ser visto em um e-commerce que opera em múltiplos canais, como aplicativo móvel, website e lojas físicas. Imagine um cenário onde milhões de usuários tentam realizar compras simultaneamente através desses diferentes canais. Para garantir que o primeiro usuário a clicar no botão de compra realmente consiga adquirir o produto, independentemente do canal que esteja utilizando, um sistema de mensageria eficiente é crucial.

Figura 1 - Arquitetura Kafka



Fonte: Kreps et al., 2011, p.2, tradução nossa

O Funcionamento de uma solução de mensageria seguiria a seguinte lógica:

1. **Centralização dos Requests:** Ao invés de cada canal processar as solicitações de compra de forma isolada, todas as requisições são enviadas para um sistema de mensageria centralizado.
2. **Fila de Mensagens:** Utiliza-se uma fila FIFO para armazenar as requisições de compra. A primeira requisição a chegar na fila é a primeira a ser processada.
3. **Processamento Assíncrono:** Cada requisição é processada conforme sua ordem na fila. Se o produto ainda estiver disponível, a compra é efetuada; caso contrário, uma mensagem de rejeição é enviada ao usuário.
4. **Escalabilidade e Confiabilidade:** Ferramentas como Apache Kafka ou Amazon SQS garantem que o sistema seja escalável e confiável, capaz de

lidar com grandes volumes de transações sem comprometer a performance.

A implementação de uma arquitetura de sistemas baseada em mensageria e filas é fundamental para aplicações omnichannel que exigem alta confiabilidade e eficiência no processamento de requisições. No contexto no caso de e-commerce, esta abordagem não só melhora a experiência do usuário, garantindo justiça na ordem de processamento, mas também assegura a integridade e desempenho do sistema sob condições de alta demanda.

2.2.6.5 A Chave para a Fluidez da Experiência do Usuário

A fluidez na experiência do usuário em um sistema omnichannel transcende a mera integração de canais. Trata-se de orquestrar uma sinfonia de interações que se desenrola de forma harmoniosa, independentemente do ponto de contato escolhido pelo usuário. O conceito de quebra de inércia emerge como um obstáculo crucial a ser superado nesse contexto.

A quebra de inércia, representa um fenômeno crucial na experiência do usuário em aplicações digitais. Ele ocorre quando a fluidez de interação é interrompida por processos que demandam maior esforço do usuário, como preenchimento excessivo de campos ou transições abruptas entre telas. Este fenômeno pode resultar na desistência do usuário, impactando diretamente as taxas de conversão em plataformas. Este conceito é crucial em ambientes omnichannel, onde a fluidez do processo de compra, por exemplo, pode determinar o sucesso ou fracasso de uma transação.

Na arquitetura de sistemas omnichannel, a minimização da quebra de inércia é essencial para proporcionar uma experiência de compra contínua e eficiente. Isso requer um projeto detalhado dos fluxos de aplicação e processos de aplicação, visando identificar e mitigar gargalos que possam desacelerar ou interromper o fluxo do usuário. Esses gargalos podem surgir em diferentes pontos, como integrações de sistemas, requisições de API, processamento de dados e renderização de interfaces.

Neste contexto, é fundamental identificar os pontos de fricção dentro do fluxo de aplicação. Por exemplo, em um e-commerce, um usuário pode navegar

rapidamente pelo site, adicionar produtos ao carrinho de forma eficiente, mas, ao enfrentar muitos formulários extensos ou várias telas para finalizar a compra, pode ser desencorajado e abandonar o processo.

Para otimizar a conversão em sistemas omnichannel, é necessário realizar um exercício constante de avaliação e ajuste dos processos de aplicação. Isso envolve identificar e eliminar os gargalos que podem causar a quebra de inércia e reduzir a conversão. A arquitetura de sistemas desempenha um papel crucial nesse sentido, garantindo que os processos sejam fluidos e eficientes em todos os canais de interação.

A fim de mitigar a quebra de inércia, é recomendado implementar estratégias como:

- Design centrado no usuário: a arquitetura do sistema deve ser moldada em torno das necessidades e expectativas do usuário, eliminando pontos de atrito e simplificando fluxos de navegação.
- Micro interações significativas: pequenas interações, como animações sutis e feedback visual imediato, podem contribuir para a sensação de fluidez e responsividade do sistema.
- Uso de tecnologias de Cache e CDN: para melhorar o desempenho e a velocidade de carregamento.
- Automatização de Processos: redução do número de interações necessárias para completar uma transação.
- Testes e iterações constantes: a otimização da experiência do usuário é um processo contínuo que exige testes rigorosos e iterações constantes para garantir que o sistema esteja sempre evoluindo em sintonia com as necessidades do usuário.

No contexto de aplicações omnichannel, o desenvolvimento de sistemas deve estar alinhado com a necessidade de proporcionar uma experiência de usuário fluida e coesa em diferentes canais de interação. Isso requer a implementação de estratégias de arquitetura que favoreçam a integração de dados em tempo real, a sincronização de estados entre dispositivos e a personalização da experiência do usuário de acordo com o contexto de interação.

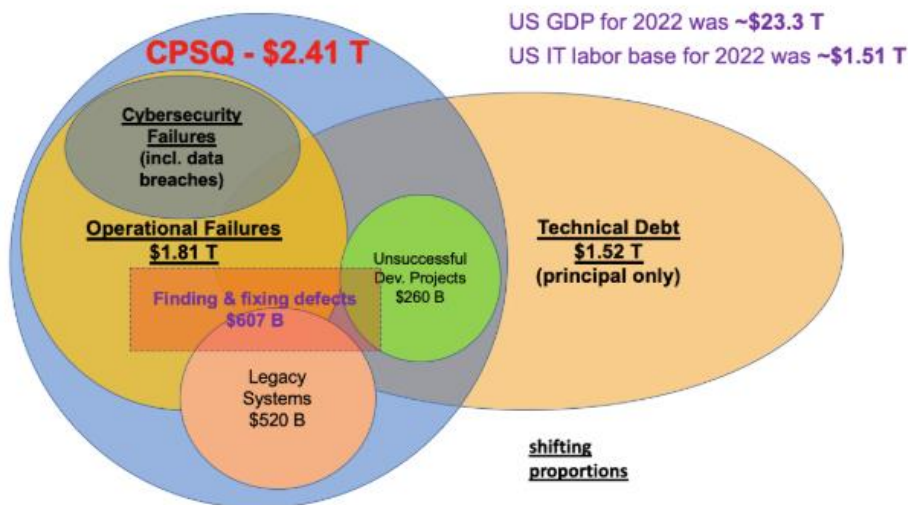
Ao analisar a quebra de inércia, é possível identificar oportunidades significativas para melhorar a experiência do usuário e aumentar a conversão. Essas

melhorias não só beneficiam a empresa, aumentando suas taxas de conversão, mas também melhoram a satisfação do cliente ao oferecer uma experiência mais intuitiva e eficiente.

2.2.7 O Custo do Código Sujo

De acordo com o CISQ (Consortium for Information & Software Quality), o custo associado ao código sujo atingiu a marca de aproximadamente 2,41 trilhões de dólares em 2022. Este montante abrange predominantemente o débito técnico, estimado em cerca de 1,52 trilhões de dólares como demonstra o gráfico 1. O débito técnico, conforme definido pela pesquisa, engloba todos os elementos que contribuem para a entrega acelerada de código, como bugs, código legado, código mal escrito e código de baixa qualidade (Krasner, 2022).

Gráfico 2 - Custo do Software de Má Qualidade



Fonte: Krasner, 2022, p.3.

É notável o interesse crescente das empresas em compreender os custos associados ao código sujo, demonstrando uma mudança de foco além da mera entrega rápida de funcionalidades ao usuário. Este reconhecimento evidencia que a dívida técnica possui implicações monetárias significativas. Como resultado, a demanda por profissionais capacitados para lidar com o débito técnico tende a aumentar consideravelmente nos próximos anos. Profissionais que não apenas compreendem como quitar essa dívida, mas também têm habilidades para

desenvolver códigos livres de débito técnico, emergem como diferenciais cruciais em nossa área de atuação.

Assim, torna-se essencial refletir sobre a capacidade individual de criar códigos que minimizem ou eliminem o débito técnico. Este atributo não apenas agrega valor ao profissional, mas também contribui para a construção de sistemas robustos e sustentáveis em um contexto omnichannel.

Além disso, é crucial ressaltar a estreita relação entre a negligência em adotar práticas de código limpo e arquitetura limpa e os custos associados ao débito técnico. A ausência de padrões de qualidade no desenvolvimento de software pode resultar em sistemas complexos, de difícil manutenção e evolução. A falta de atenção à estruturação adequada do código pode gerar um aumento exponencial no débito técnico, tornando-o ainda mais difícil de ser pago no futuro. Ignorar a importância do código limpo e da arquitetura limpa não apenas compromete a qualidade do software, mas também pode impactar negativamente a eficiência operacional e a competitividade das empresas no mercado omnichannel.

Nesse sentido, a pesquisa do CISQ serve como um alerta, destacando a necessidade premente de investir em práticas que promovam a qualidade do código desde as fases iniciais do desenvolvimento. A adoção de abordagens como código limpo e arquitetura limpa não apenas reduz o custo do débito técnico a longo prazo, mas também contribui para a criação de sistemas mais robustos, escaláveis e resilientes, capazes de atender às demandas do mercado omnichannel com excelência.

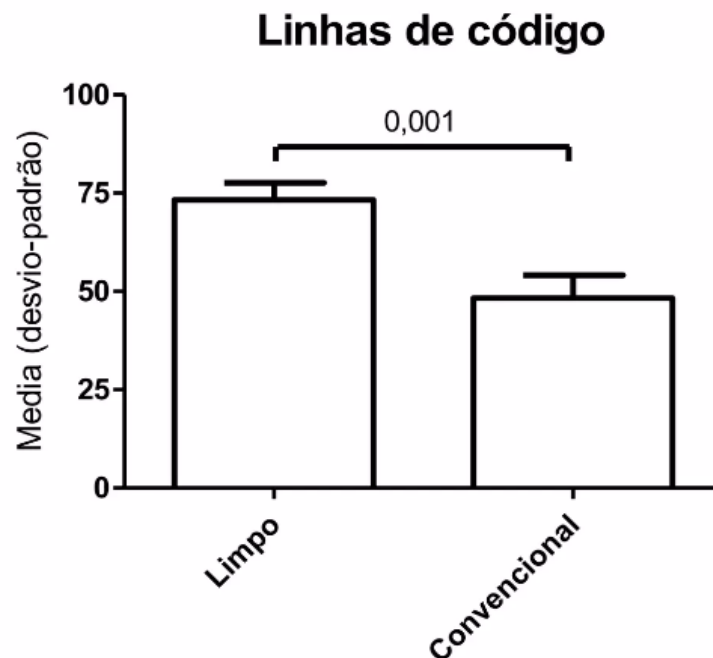
2.2.8 O Código Sujo Não Produz Com Mais Eficiência

A arquitetura de sistemas para omnichannel é uma abordagem estratégica para o desenvolvimento de sistemas de informação que integram diversos canais de comunicação e interação com os clientes, como lojas físicas, sites, aplicativos móveis, redes sociais, entre outros. Essa integração busca oferecer uma experiência consistente e personalizada aos consumidores, independentemente do canal utilizado, promovendo a fidelização e satisfação do cliente.

No contexto do desenvolvimento de sistemas para omnichannel, a qualidade do código desempenha um papel fundamental na eficiência e manutenibilidade da

plataforma. Contrariamente à crença comum de que o código sujo é mais rápido de produzir, conforme demonstrado por Diniz no artigo "A Importância do Código Limpo na Perspectiva dos Desenvolvedores e Empresas de Software", essa percepção não se sustenta. Ele destaca a necessidade de um investimento maior de tempo e recursos na produção de código limpo, em comparação com alternativas mais improvisadas. É válido ressaltar que o código limpo demanda uma quantidade significativa de linhas adicionais como demonstra o gráfico 2, o que pode levar a uma impressão inicial de menor produtividade.

Gráfico 3 - Linhas de código (limpo x convencional)



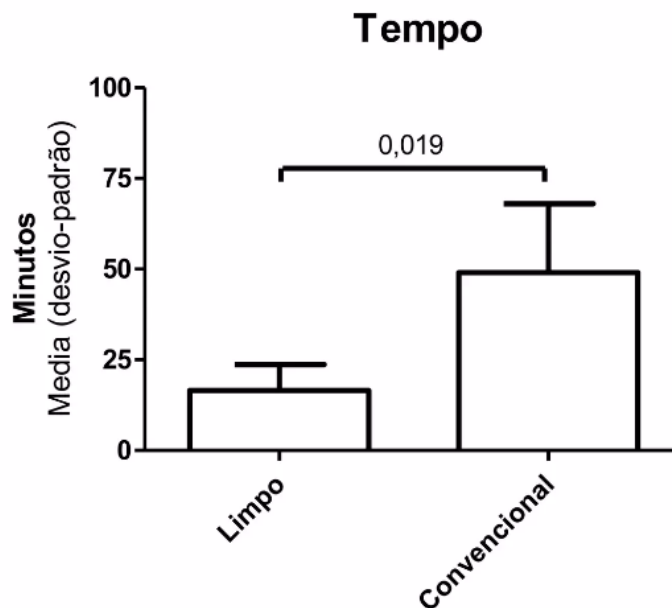
Fonte: Diniz, 2012, p.17.

No entanto, a análise mais aprofundada revela que essa aparente desvantagem se traduz em benefícios substanciais a longo prazo. Em particular, quando se trata de realizar modificações ou atualizações no sistema, o código limpo demonstra uma vantagem incontestável em termos de agilidade e eficácia. O tempo necessário para efetuar alterações em código limpo é significativamente inferior ao exigido para realizar as mesmas modificações em um ambiente de código sujo como mostra o gráfico 3.

Portanto, é fundamental reconhecer a importância da qualidade do código no processo de desenvolvimento de sistemas, especialmente em um contexto

omnichannel. Em vez de buscar atalhos ou soluções temporárias, é preciso adotar uma abordagem centrada na arquitetura de sistemas, priorizando a clareza, organização e manutenibilidade do código. Ao investir no desenvolvimento de sistemas com código limpo e bem estruturado, as organizações poderão colher os frutos de uma operação mais eficiente e resiliente, capaz de se adaptar às demandas dinâmicas do ambiente omnichannel.

Gráfico 4 - Tempo que leva para efetuar alterações (limpo x convencional)



Fonte: Diniz, 2012, p.17.

Ao implementar uma nova funcionalidade, surge a necessidade premente de garantir que o código a ser inserido no produto respeite integralmente as definições preconizadas pela arquitetura e os princípios de código limpo prezando pela qualidade predefinida.

A qualidade pode ser definida como um conjunto de características de uma unidade (por exemplo, produto ou processo) quanto à sua adequação, para atender a requisitos especificados e pressupostos. Portanto, tanto a qualidade do produto quanto a do processo precisam ser abordadas para atingir a qualidade exigida do produto. Para inspeção de qualidade, pode ser feita uma distinção entre tecnologia de medição no produto e tecnologia de medição no processo para atender aos requisitos de qualidade. (Filz et al., 2020, tradução nossa)

Nesse contexto, adotamos dentro do nosso processo de integração contínua o conceito de "Quality Gate", ou seja, um portão de qualidade que serve como filtro

para garantir a integridade e eficácia do código em questão, utilizando ferramentas como o SonarQube, por exemplo, para realizar análises estáticas e garantir a conformidade com as melhores práticas de codificação.

O "Quality Gate" funciona como um portão de qualidade, filtrando o código antes de sua integração à produção. Diversas ferramentas de análise estática de código, como o SonarQube, são utilizadas para avaliar diversos aspectos do código, incluindo: conformidade com as regras de codificação, a análise verifica se o código segue as boas práticas de codificação pré-definidas pela equipe, como convenções de nomenclatura, formatação de código e uso de padrões de codificação. Métricas de complexidade, a análise avalia a complexidade do código, como a métrica ciclomática, para identificar trechos que podem ser refatorados e facilitar a leitura e a manutenção do código. Detecção de vulnerabilidades, a análise verifica se o código contém vulnerabilidades de segurança conhecidas, como injeção de SQL, cross-site scripting e falhas de autenticação. Cobertura de testes, a análise verifica a cobertura dos testes unitários e de integração, garantindo que o código seja suficientemente testado e robusto (Querido, 2023).

Ao ultrapassar o "Quality Gate", o código avança para a próxima etapa do processo de integração contínua, estando apto para prosseguir na implementação e entrega do produto. No entanto, caso o código não atenda aos requisitos mínimos de qualidade, o processo de integração contínua automaticamente o bloqueia, impedindo sua inserção na produção até que as correções necessárias sejam realizadas e os critérios de qualidade estabelecidos sejam atendidos.

A implementação do "Quality Gate" oferece diversos benefícios, como: redução de bugs e falhas, onde a análise estática ajuda a identificar e corrigir bugs e falhas no código antes que eles causem problemas em produção. Melhoria da qualidade do código, visto que a análise garante que o código siga as melhores práticas de codificação, resultando em um código mais legível, fácil de manter e menos propenso a erros. Aumento da produtividade, pois a análise automatizada de código libera os desenvolvedores para se concentrarem em tarefas mais complexas e criativas (Giebel et al., 2009).

Basicamente, são exames de uma composição de critérios de qualidade pré-definidos que devem ser cumpridos para continuar de uma etapa do projeto para a próxima. Os "Quality Gates" garantem maior foco por meio de listas de verificação específicas de fase, minimizam o risco do projeto,

garantem feedback direto do processo (e a qualidade da execução) para o estágio atual do projeto/produto, reduzem o tempo de desenvolvimento "fazendo certo da primeira vez", e garantem um projeto/produto que atenda aos requisitos do cliente. (Giebel et al., 2009, tradução nossa)

Ao implementar esse validador, alguns aspectos específicos do projeto devem ser levados em consideração. Por exemplo, a escolha da ferramenta de análise estática de código deve ser baseada nas necessidades do projeto e da equipe envolvida. Essa escolha estratégica é essencial para garantir uma avaliação completa e precisa do código, que possa identificar possíveis falhas e oportunidades de melhoria.

Além disso, estabelecer critérios claros e mensuráveis de qualidade é essencial para a avaliação pelo "Quality Gate". Esses padrões orientam o desenvolvimento do código e fornecem uma base razoável para avaliação e tomada de decisão durante o processo de integração contínua.

Outra prática crucial é incorporar ele ao processo de integração contínua. Como resultado dessa integração, o código pode ser validado automaticamente antes de cada integração à produção, garantindo que apenas código de alta qualidade e que atenda aos padrões definidos pode passar para a próxima etapa do desenvolvimento.

É fundamental que a equipe de desenvolvimento tenha a capacidade de usar a ferramenta de análise estática de código e entender os resultados. O treinamento adequado ajuda os membros da equipe a compreenderem e agir efetivamente com base nas informações da análise estática, ajudando a melhorar continuamente a qualidade do código e do processo de desenvolvimento como um todo.

Dessa forma, a implementação do "Quality Gate" dentro do ciclo de desenvolvimento de sistemas para omnichannel representa um mecanismo crucial para assegurar a robustez, segurança e confiabilidade do software entregue, alinhando-se às exigências e padrões de qualidade estabelecidos pelo time responsável pelo desenvolvimento.

A obtenção de uma métrica para medir a testabilidade e legibilidade do código é igualmente importante. Uma métrica precisa que possa quantificar esses elementos fornece uma visão objetiva e medível da qualidade do código, permitindo a identificação precisa de pontos de melhoria.

Além disso, a facilitação da manutenção do código é outro ponto chave. Códigos com alta legibilidade são mais fáceis de entender e modificar, o que agiliza e reduz os custos de manutenção a longo prazo. Uma métrica de legibilidade auxilia na identificação de trechos de código que podem ser confusos ou complexos, direcionando os esforços de melhoria para onde são mais necessários.

A redução de erros também é um benefício significativo. Códigos com boa testabilidade são mais fáceis de testar de forma abrangente, o que aumenta a confiabilidade e robustez do software. Uma métrica de testabilidade ajuda a identificar partes do código que são difíceis de testar, permitindo que sejam priorizadas para garantir uma cobertura de teste adequada. Uma métrica de medição da legibilidade e testabilidade do código é fundamental para garantir a qualidade, manutenibilidade e eficiência do desenvolvimento de software. Ela proporciona uma base sólida para avaliação, melhoria contínua e sucesso do produto final.

Para isso podemos utilizar a métrica de complexidade ciclomática. A complexidade ciclomática é uma métrica que quantifica a complexidade de um programa de computador medindo a quantidade de caminhos independentes no código-fonte. Quanto maior a complexidade ciclomática, maior a dificuldade de compreensão e testabilidade do código.

A complexidade ciclomática é calculada com base na estrutura do código fonte, especialmente em torno das estruturas de controle de fluxo, como condicionais e loops. Por exemplo, em uma estrutura de controle if-else, a complexidade ciclomática é determinada pelo número de caminhos independentes que podem ser percorridos pelo programa. Cada condicional aumenta a complexidade, já que cada ramificação cria caminhos a serem considerados.

Para calcular a complexidade ciclomática, utiliza-se a fórmula:

$$M = E - N + 2P$$

Onde:

M é a complexidade ciclomática

E é o número de arestas no grafo do programa

N é o número de nós no grafo do programa

P é o número de componentes conexos no grafo do programa

Uma alta complexidade ciclomática torna o código mais difícil de entender e testar. Isso ocorre porque aumenta a quantidade de caminhos que precisam ser

considerados durante os testes unitários, aumentando a carga de trabalho e a possibilidade de erros não detectados. Além disso, a compreensão do código é comprometida, pois se torna mais difícil acompanhar e compreender todos os caminhos de execução possíveis.

Alta complexidade ciclomática (CC) significa código complexo, apresentando desafios potenciais em legibilidade, manutenção e testes. No contexto do desenvolvimento de software, a complexidade é um fator crítico. Impacta significativamente a qualidade do software, influenciando a facilidade de compreensão e utilização. As organizações de software, reconhecendo sua importância primordial, contam com diversas métricas, incluindo CC, para avaliar o valor do código, agilizar processos e aprimorar protocolos de manutenção. Além disso, o desafio reside não apenas em compreender a complexidade, mas também em testar eficazmente o código. A CC determina diretamente o número mínimo de casos de teste necessários para alcançar a cobertura completa das filiais, um aspecto fundamental dos testes de software. (Odeh et al., 2024, tradução nossa)

Portanto, embora não seja necessário minimizar a complexidade ciclomática a todo custo, é importante estar ciente do impacto que ela pode ter na legibilidade e testabilidade do código. Manter a complexidade ciclomática em níveis razoáveis ajuda a garantir que o código seja mais fácil de entender, manter e testar, contribuindo para a qualidade geral do software desenvolvido.

2.3 A EVOLUÇÃO DA COMUNICAÇÃO EMPRESARIAL COM OMNICHANNEL

Em primeiro plano, a comunicação empresarial tem continuamente evoluído por meio de transformações nas infraestruturas tecnológicas e nas preferências e preferências do consumidor. Inicialmente, a comunicação entre empresas foi amplamente praticada através de canais tradicionais, como cartas, telefonemas e anúncios. A era digital virou a chave para a maior transformação na prática da comunicação empresarial. Segundo Bell, Gallino e Moreno (2017, p. 10), a introdução da utilização de omnichannel não apenas afeta a demanda, mas também traz novas descobertas sobre eficiência operacional, destacando a importância de choques de informação e a migração de clientes para canais mais eficientes.

No contexto atual, as organizações têm o desafio de integrar canais de comunicação para atender o aumento no número de consumidores conectados e exigentes. Diante dessa realidade, a estratégia omnichannel se torna essencial para

garantir uma experiência de comunicação alinhada em todos os pontos de contato com o cliente. Ou seja, o omnichannel possibilita a unificação de diferentes canais de comunicação, como redes sociais, websites, aplicativos e lojas físicas, para entregar uma experiência humanizada, unificada e personalizada para o consumidor tanto online quanto offline.

A importância do omnichannel na comunicação empresarial reside na capacidade de atender às expectativas dos consumidores modernos, que buscam conveniência, consistência e personalização em suas interações com as marcas. Ao adotar uma abordagem omnichannel, as empresas podem fortalecer o relacionamento com os clientes, aumentar a fidelidade e melhorar a experiência do cliente como um todo.

Por todos esses aspectos, a evolução da comunicação empresarial destaca a necessidade de as empresas se adaptarem às mudanças no ambiente digital e adotarem estratégias omnichannel para garantir uma comunicação eficaz e alinhada com as expectativas dos consumidores contemporâneos.

2.4 TRANSFORMAÇÃO DA EXPERIÊNCIA DO CLIENTE ATRAVÉS DO OMNICHANNEL

Conforme mencionado por Subramaniam (2021) em seu artigo "Omnichannel Customer Service in Retail: From Miserable to Memorable", os clientes estão frustrados, podemos expor um exemplo como: devido ao fato de que 41% dos diferentes agentes de atendimento ao cliente fornecem respostas discrepantes e 34% dos agentes de atendimento ao cliente não possuem a resposta adequada ao tentarem resolver um problema com um produto. Essas questões ressaltam a importância da integração do omnichannel para garantir consistência e eficácia nas interações com os clientes, abordando assim as lacunas identificadas na experiência atual.

No cenário atual, os consumidores enfrentam frequentemente experiências negativas ao tentarem resolver problemas ou obter suporte para produtos e serviços adquiridos. A transição de um canal de comunicação para outro, longos períodos de espera e a repetição de informações são apenas algumas das frustrações enfrentadas pelos clientes. Neste contexto, a implementação eficaz de estratégias

omnichannel pode oferecer uma solução promissora para melhorar a experiência do cliente e resolver esses desafios. Esta seção explorará como o conceito de omnichannel pode ser aplicado para superar as más experiências atuais dos clientes e atender às expectativas crescentes dos consumidores.

No ambiente altamente conectado de hoje, os consumidores esperam respostas rápidas e em tempo real, independentemente do canal de comunicação utilizado. As experiências negativas, como longos períodos de espera e a repetição de informações, podem ser evitadas através da implementação de um sistema omnichannel eficaz, onde os clientes podem continuar suas interações sem interrupções, independentemente do canal utilizado. Esta coesão simplifica o processo para o cliente e demonstra um alto nível de compreensão por parte da empresa, aspectos cruciais para a fidelização do cliente.

Por exemplo, em vez de serem direcionados para um número de telefone genérico, os clientes podem optar por interagir por meio de um chat automatizado, onde suas consultas podem ser resolvidas instantaneamente ou encaminhadas para o representante mais adequado. Isso não apenas aumenta a eficiência do suporte ao cliente, mas também melhora significativamente a satisfação do cliente.

Além disso, a integração de canais de comunicação permite uma experiência mais unificada, onde os clientes não são passados de um departamento para outro. Ao lidar com um problema, os clientes podem iniciar uma conversa em um canal e continuar exatamente de onde pararam em outro, sem a necessidade de repetir informações. Isso não só simplifica o processo para o cliente, mas também demonstra um alto nível de compreensão e atenção por parte da empresa. Outro ponto crucial é a adaptação das estratégias empresariais aos canais digitais preferidos pelos clientes. Em vez de impor canais específicos, as empresas devem ajustar suas abordagens para atender às preferências dos clientes. Estar presente nos canais onde os clientes estão ativos facilita a interação e aumenta a probabilidade de resolver problemas de forma eficaz e satisfatória.

A abordagem omnichannel também aborda a necessidade de as empresas estarem presentes nos canais digitais preferidos pelos clientes. Em vez de impor canais de comunicação específicos, as empresas devem adaptar suas estratégias para atender às preferências dos clientes. Ao estar presente nos canais onde os clientes já estão ativos, as empresas podem facilitar a interação e aumentar a probabilidade de resolver problemas de forma eficaz e satisfatória.

Em um mundo onde as expectativas dos clientes estão em constante evolução, a implementação visa garantir uma experiência do cliente positiva e satisfatória. Ao oferecer respostas rápidas, unificar as interações e estar presente nos canais preferidos pelos clientes, as empresas podem não apenas resolver os desafios atuais de experiência do cliente, mas também construir relacionamentos mais fortes e duradouros com seus clientes.

A unificação das interações em uma única plataforma garante que o histórico do cliente seja acessível a todos os atendentes, eliminando a necessidade de repetir informações e agilizando o atendimento. Além disso, a presença da empresa nos canais preferidos pelos clientes, demonstra proatividade e facilita a comunicação, resultando em maior satisfação e fidelização.

A integração do omnichannel não apenas soluciona os desafios da experiência do cliente, mas também impulsiona o crescimento dos negócios. A convergência de múltiplos canais de venda amplia o alcance do varejista, atraindo consumidores com preferências diversas e aumentando a base de clientes. Além disso, a coleta de dados em vários canais oferece informações valiosas sobre o comportamento e as preferências do consumidor, permitindo a personalização da experiência e a otimização de estratégias de marketing. Os varejistas podem analisar esses dados para compreender os padrões de compra, otimizar estratégias de marketing e personalizar ainda mais a experiência do cliente. Esses insights permitem que as decisões sejam baseadas em dados e que ocorra adaptação de suas ofertas para atender às demandas dos clientes.

A importância da comunicação eficiente com o cliente é inegável. Uma pesquisa recente revelou que 90% dos consumidores desejam se comunicar com as marcas por meio da tecnologia, porém, menos da metade das empresas globais possuem a infraestrutura necessária para atender a essa demanda (“Global Contact Centre Benchmarking Report Digital transformation with a human touch”, 2016; “Global Mobile Messaging Consumer Report”, 2016). Essa lacuna na comunicação pode resultar em frustração, visto que 42% dos consumidores abandonam a marca frustrados, dos quais 21% nunca mais voltam (“The Power of Brand Purpose | Accenture”, [s.d.]). Diante desse cenário, torna-se evidente que as empresas precisam oferecer uma experiência do cliente impecável, que englobe respostas rápidas, experiências unificadas e atendimento nos canais preferidos pelos clientes.

Olhando o comércio de consumo online vemos a presença de um paradoxo: apesar da alta taxa de usuários globais que realizam compras online (93,5%), a conversão de visitas em compras em sites de e-commerce permanece baixa (menos de 3%). Este cenário sugere um vasto potencial de crescimento, especialmente ao considerar o comportamento de compra de consumidores jovens e a frequência de compras em plataformas como a Amazon, onde 41% dos clientes nos EUA recebem múltiplos pacotes semanais. A preferência por métodos de pagamento digitais, como cartões de crédito (53%) e sistemas de pagamento digital (43%), reforça a necessidade de soluções que integrem diferentes canais de forma eficiente e segura. Adicionalmente, a crescente aceitação de chatbots como ferramenta de compra (50%) indica uma oportunidade para empresas que buscam inovar na experiência do cliente (“The ultimate list of 70+ eCommerce facts and statistics for 2024 - AppMySite”, [s.d.])

Ao proporcionar uma experiência de compra fluida e consistente em todos os canais, as empresas podem otimizar a jornada do cliente, desde a descoberta do produto até o pós-venda. Esta abordagem integrada não apenas aumenta as chances de atrair novos clientes, mas também fortalece o relacionamento com os clientes existentes, promovendo a fidelização e o aumento do ticket médio.

O omnichannel não apenas resolve os desafios da comunicação fragmentada, mas também se apresenta como um diferencial competitivo, permitindo que as empresas se destaquem em um mercado cada vez mais exigente e conectado. Ao adotar essa abordagem, as empresas investem em um relacionamento de longo prazo com seus clientes, construindo uma reputação positiva, impulsionando o crescimento sustentável do negócio e aproveitar as oportunidades de um mercado em constante expansão, oferecendo soluções que atendam às expectativas dos consumidores e impulsionem o crescimento do negócio.

Em suma, o omnichannel não apenas soluciona os desafios da experiência do cliente, mas também impulsiona o crescimento dos negócios, tornando-se uma ferramenta essencial para as empresas que desejam se destacar em um mercado cada vez mais competitivo e conectado.

3 CONCLUSÃO

Com base nas análises realizadas, fica claro que a implementação eficaz do omnichannel requer uma arquitetura de sistemas robusta e flexível, capaz de integrar e coordenar diversos canais de comunicação de maneira coesa. A evolução da comunicação empresarial para esse modelo reflete uma resposta direta às expectativas crescentes dos consumidores por experiências integradas e consistentes.

A integração de plataformas distintas, a gestão eficaz de dados e a adaptação rápida às mudanças no ambiente digital emergem como desafios significativos enfrentados pelas organizações que buscam adotar o omnichannel. Esses desafios demonstram profunda relação entre o desenvolvimento de sistemas omnichannel e a aplicação de princípios de código limpo e arquitetura limpa, evidenciando o impacto direto dessas práticas no sucesso dos negócios.

Verificou-se que a implementação de uma arquitetura omnichannel, fundamentada em código limpo e arquitetura limpa, facilita a integração de múltiplos canais de comunicação, resultando em uma experiência do cliente mais fluida, personalizada e engajadora. Adicionalmente, a adoção de práticas como arquitetura orientada a eventos, tolerância a falhas, sharding de banco de dados e sistemas de mensageria robustos, contribui para a escalabilidade e confiabilidade dos sistemas, garantindo a continuidade do serviço mesmo em cenários de alta demanda, o que se traduz em maior disponibilidade do negócio e aumento da satisfação do cliente.

Constatou-se também que a negligência em relação à qualidade do código e à arquitetura pode acarretar custos significativos para as empresas, tanto em termos financeiros quanto de reputação. O código sujo e a arquitetura inadequada dificultam a manutenção e evolução dos sistemas, resultando em atrasos, erros, aumento dos custos de desenvolvimento e perda de oportunidades de negócio.

A análise demonstrou que a convergência entre tecnologia e estratégia não apenas otimiza a experiência do usuário, mas também promove a eficiência, sustentabilidade e, conseqüentemente, a lucratividade dos sistemas.

Foi destacado a necessidade contínua de adaptação e inovação por parte das organizações, a fim de satisfazer as expectativas cada vez mais exigentes dos consumidores digitais. Através de uma arquitetura de sistemas adequada e estratégias omnichannel bem definidas, as empresas podem não apenas melhorar a

experiência do cliente, mas também alcançar maior competitividade e sustentabilidade no mercado atual. A personalização da experiência do cliente, baseada em dados comportamentais e preferenciais, reforça a importância de uma abordagem centrada no cliente.

Por fim, este estudo evidencia que a comunicação omnichannel, em sinergia com uma arquitetura de sistemas sólida e eficaz, emerge como um pilar fundamental para o êxito empresarial no cenário digital atual. A adoção de práticas de código limpo e arquitetura limpa, aliada a investimentos em tecnologias pertinentes, configuram-se como fatores determinantes para assegurar a eficiência, escalabilidade e qualidade dos sistemas, culminando em uma experiência do cliente aprimorada e em um posicionamento competitivo mais robusto no mercado.

REFERÊNCIAS

ABDELHAFIZ, B. M. **Distributed Database Using Sharding Database Architecture**. 2020 IEEE Asia-Pacific Conference on Computer Science and Data Engineering, CSDE 2020, 16 dez. 2020.

ABDELHAFIZ, B. M.; ELHADEF, M. **Sharding database for fault tolerance and scalability of data**. Proceedings of 2nd International Conference on Computation, Automation and Knowledge Management, ICCAKM 2021, p. 17–24, 19 jan. 2021.

Apache Kafka. Disponível em: <<https://kafka.apache.org/>>. Acesso em: 18 maio. 2024.

BALASUBRAMANIAN, J. et al. **Adaptive failover for real-time middleware with passive replication**. Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS, p. 118–127, 2009.

BELL, David; GALLINO, Santiago; MORENO, Antonio. (2017). **Offline showrooms in omni-channel retail: demand and operational benefits**. *Management Science*, v. 64, p. 1629-1651. DOI: 10.1287/mnsc.2016.2684

CURRY, E. **Message-Oriented Middleware**. *Middleware for Communications*, p. 1–28, 1 jul. 2005.

DINIZ, J. **A Importância do Código Limpo na Perspectiva dos Desenvolvedores e Empresas de Software**. [s.l.] Fatec Rio Preto, 2012.

Enfileiramento de mensagens totalmente gerenciado – **Amazon Simple Queue Service – Amazon Web Services**. Disponível em: <<https://aws.amazon.com/pt/sqs/>>. Acesso em: 18 maio. 2024.

FILZ, M. A. et al. Virtual quality gates in manufacturing systems: Framework, implementation and potential. **Journal of Manufacturing and Materials Processing**, v. 4, n. 4, 2020.

FORSGREN, N. et al. DevEx in Action - A study of its tangible impacts. **Queue**, v. 21, n. 6, p. 47–77, 31 dez. 2023.

Global Contact Centre Benchmarking Report Digital transformation with a human touch. . [s.l: s.n.]. . Acesso em: 18 maio. 2024.

Global Mobile Messaging Consumer Report. . [s.l: s.n.]. Disponível em: <<https://www.twilio.com/en-us/press/releases/twilio-study-finds-that-9-out-of-10-consumers-globally-want-to-message-with-brands>>. Acesso em: 18 maio. 2024.

GIEBEL, M. et al. Improved innovation through the integration of Quality Gates into the Enterprise and Product Lifecycle Roadmaps. **CIRP Journal of Manufacturing Science and Technology**, v. 1, n. 3, 2009.

KREPS, J. et al. **Kafka: a Distributed Messaging System for Log Processing**. 2011.

KRASNER, H. **Cost of Poor Software Quality in the U.S.: A 2022 Report - CISQ**. [s.l.: s.n.]. Disponível em: <<https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/>>. Acesso em: 3 maio. 2024.

MARTIN, R. **Arquitetura limpa: o guia do artesanato para estrutura e design de software**. [s.l.] Alta Books, 2019.

MARTIN, R. **Código limpo: habilidades práticas do Agile software (1ª edição)**. Alta Books, 2009.

ODEH, A. H. et al. **Measuring Cyclomatic Complexity of Source Code Using Machine Learning**. *Revue d'Intelligence Artificielle*, v. 38, n. 1, 2024.

QUERIDO, J. F. C. **Avaliação de Qualidade de Desenho de Software por Cobertura e Sincronismo com o Código**. [s.l.: s.n.]. Instituto Superior de Engenharia do Porto, 2023. Disponível em: <<https://recipp.ipp.pt/handle/10400.22/24222/>>. Acesso em: 8 maio. 2024.

RabbitMQ: **One broker to queue them all** | RabbitMQ. Disponível em: <<https://www.rabbitmq.com/>>. Acesso em: 18 maio. 2024.

ROSS, J. W.; WEILL, P.; ROBERTSON, D. *Enterprise architecture as strategy : creating a foundation for business execution*. Boston, Mass.: **Harvard Business School Press**, 2006.

SUBRAMANIAM, Anand. Omnichannel Customer Service in Retail: From Miserable to Memorable. **eGain**, 2021, Disponível em: <<https://hd.egain.com/articles/retail-customer-service-best-practices.pdf>>. Acesso em: 14 abr. 2024.

VERHOEF, P. C.; Kannan, P. K.; Inman, J. J. From multi-channel retailing to omnichannel retailing: introduction to the special issue on multi-channel retailing. **Journal of Retailing**, v. 91, n. 2, p. 174-181, 2015.

The Power of Brand Purpose | Accenture. Disponível em: <<https://www.accenture.com/us-en/insights/strategy/brand-purpose>>. Acesso em: 18 maio. 2024.

The ultimate list of 70+ eCommerce facts and statistics for 2024 - AppMySite. Disponível em: <<https://www.appmysite.com/blog/ultimate-ecommerce-facts-and-statistics/>>. Acesso em: 19 maio. 2024.