CENTRO UNIVERSITÁRIO BRASILEIRO - UNIBRA
CURSO DE GRADUAÇÃO TECNÓLOGO EM REDES DE COMPUTADORES

GUSTAVO PINTO DA SILVA

# IMPLEMENTAÇÃO DE DEVOPS PARA MELHORIA DO DESENVOLVIMENTO E ENTREGA DE SOFTWARE

RECIFE
2023

GUSTAVO PINTO DA SILVA

# IMPLEMENTAÇÃO DE DEVOPS PARA MELHORIA DO DESENVOLVIMENTO E ENTREGA DE SOFTWARE

Artigo apresentado ao Centro Universitário Brasileiro – UNIBRA, como requisito parcial para obtenção do título de Bacharel em Redes de Computadores.

Professor(a) Orientador(a):

RECIFE

2023

Ficha catalográfica elaborada pela
bibliotecária: Dayane Apolinário, CRB4- 2338/ O.

S586i Silva, Gustavo Pinto da.
Implementação de devops para melhoria do desenvolvimento e entrega de software / Gustavo Pinto da Silva. - Recife: O Autor, 2023.
18 p.

Orientador(a): Wanuska Munique Portugal.

Trabalho de Conclusão de Curso (Graduação) - Centro Universitário Brasileiro - UNIBRA. Tecnólogo em Redes de Computadores, 2023.

Inclui Referências.

1. Devops. 2. Desenvolvimento de software. 3. Automação. I. Centro Universitário Brasileiro. - UNIBRA. II. Título.

CDU: 004

*Dedico esse trabalho a minha família e amigos*.

## AGRADECIMENTOS

Agradeço a Deus, minha família e as pessoas que estiveram ao meu lado durante o desenvolvimento deste trabalho.

*“Ninguém ignora tudo. Ninguém sabe tudo. Todos nós sabemos alguma coisa. Todos nós ignoramos alguma coisa. Por isso aprendemos sempre.”*

*(Paulo Freire)*

## LISTA DE ABREVIATURAS E SIGLAS

**ABNT -** Associação Brasileira de Normas Técnicas

**BDD -** Behavioral Driven Development

**CMM -** Modelo de Maturidade em Capacitação

**DevOps -** Desenvolvimento e Operações

**HTTP -** Hypertext Transfer Protocol

**IES -** Instituição de Ensino Superior

**ISO -** Organização Internacional de Normalização

**NBR -** Norma Regulamentadora

**P.O. -** Product Owner

**PBQP -** Programa Brasileiro de Qualidade e Produtividade em Software

**T.I. -** Tecnologia da Informação

**TDD -** Test Driven Development

**XP**- Extreme Programming

# IMPLEMENTAÇÃO DE DEVOPS PARA MELHORIA DO DESENVOLVIMENTO E ENTREGA DE SOFTWARE

Gustavo Pinto da Silva

**Resumo:** Este estudo explora a metodologia DevOps e sua aplicação para aprimorar o ciclo de desenvolvimento e entrega de software. O DevOps promove a colaboração entre equipes de desenvolvimento e operações, focando na automação, integração contínua e entrega contínua. O objetivo principal é analisar como a implementação das práticas DevOps pode aprimorar a eficiência, qualidade e agilidade no desenvolvimento de software. O trabalho discute os princípios fundamentais do DevOps, ferramentas de automação, integração e entrega contínua, além de explorar os benefícios e desafios associados à sua adoção. Adicionalmente, apresenta-se um estudo de caso detalhado, onde as práticas DevOps foram aplicadas em um ambiente de desenvolvimento real, evidenciando os resultados alcançados.

**Palavras-chave:** DevOps, Desenvolvimento de Software, Automação

**SUMÁRIO**

## 1. INTRODUÇÃO

No cenário tecnológico atual, onde a velocidade das inovações digitais é vertiginosa e a dependência das organizações em relação à infraestrutura de TI é crucial, a eficiência na administração de redes de computadores torna-se uma prioridade estratégica (DE MENDONÇA; DE SOUSA NETO, 2020). Nesse contexto, a cultura DevOps emerge como um paradigma revolucionário, promovendo uma integração estreita entre os departamentos de desenvolvimento (Dev) e operações (Ops), transcende as barreiras tradicionais e reforça uma colaboração contínua para alcançar um ciclo de vida de desenvolvimento ágil e operações robustas (MUNIZ, 2019).

Este trabalho explora a implementação da cultura DevOps na administração de redes de computadores, um campo no qual a agilidade, confiabilidade e escalabilidade são essenciais para o sucesso das organizações (SILVA, 2021).

A integração de práticas DevOps na gestão de redes não apenas otimiza o processo de desenvolvimento e implementação, mas também aprimora a monitorização, a segurança e a manutenção dos sistemas, resultando em operações de TI mais ágeis e eficazes (SOARES, 2022).

Ao longo deste estudo, será analisado o impacto da cultura DevOps na administração de redes, considerando não apenas as transformações técnicas, mas também as mudanças culturais e organizacionais necessárias para sua adoção efetiva. Além disso, serão discutidos os desafios comuns enfrentados pelas organizações durante o processo de implementação do DevOps, bem como as estratégias para superá-los.

A pesquisa conduzida neste trabalho visa fornecer uma compreensão aprofundada das práticas DevOps aplicadas à administração de redes de computadores, destacando os benefícios tangíveis e intangíveis dessa abordagem. Ao final, espera-se não apenas apresentar um panorama detalhado sobre a implementação da cultura DevOps, mas também oferecer insights valiosos para

profissionais de TI e gestores que buscam aprimorar suas operações de rede e adaptar-se eficazmente às demandas dinâmicas do mundo digital contemporâneo.

### 1.1 Problema de Pesquisa

Com a adoção do movimento de software ágil, que visa acelerar e aprimorar a entrega de software, observa-se um desequilíbrio entre o desenvolvimento de software e as operações associadas, resultando na emergência do conceito de DevOps. O DevOps representa uma filosofia em constante evolução nas empresas, propondo uma abordagem inovadora para o desenvolvimento e gerenciamento de software. Dentro desse contexto, destaca-se a automação e otimização da gestão da qualidade do software.

Diante desse cenário, o presente estudo busca responder à seguinte indagação: de que maneira as práticas do DevOps podem influenciar positivamente na qualidade do desenvolvimento de software? Para abordar essa questão de pesquisa, este trabalho delineia objetivos gerais e específicos.

#### 1.1.1 Objetivo Geral

O propósito principal deste estudo consiste em investigar como as práticas adotadas pelo DevOps podem contribuir para aprimorar a qualidade do desenvolvimento de software.

#### 1.1.2 Objetivo Específico

Com o intuito de alcançar o objetivo geral estabelecido, delinearam-se os seguintes objetivos específicos:

1. Identificar e analisar as práticas do DevOps no contexto do desenvolvimento de software.
2. Avaliar o impacto das práticas do DevOps na garantia de qualidade durante o desenvolvimento de software.
3. Descrever de que maneira as práticas do DevOps influenciam positivamente na qualidade do desenvolvimento de software.

## 2. DELINEAMENTO METODOLÓGICO

A metodologia de investigação adotada para suportar o desenvolvimento deste projeto foi o estudo de caso, especificamente do tipo descritivo, conforme proposto por Runeson e Host (2009). Essa escolha se justifica pelo propósito da investigação, que visa aprofundar o entendimento do fenômeno DevOps dentro de seu contexto, buscando uma caracterização mais abrangente e detalhada.

O estudo de caso é a investigação qualitativa mais utilizada em sistemas de informação, sendo uma metodologia apropriada para a compreensão das interações entre as inovações relacionadas com a tecnologia da informação e os contextos organizacionais (DARKE, SHANKS, & BROADBENT, 1998).

## 3. REFERENCIAL TEÓRICO

Este Trabalho de Curso está dividido em quatro capítulos, cada um abordando aspectos específicos conforme descrito a seguir.

No primeiro capítulo, é realizada a introdução do conteúdo, abordando o problema de pesquisa e delineando os objetivos gerais e específicos.

O segundo capítulo detalha a metodologia de pesquisa empregada na realização do trabalho.

O terceiro capítulo apresenta o embasamento teórico, explorando os conceitos relacionados ao desenvolvimento de software, DevOps e qualidade de software. No contexto do desenvolvimento de software, são discutidos os princípios das metodologias ágeis, com ênfase em Scrum e Extreme Programming. A seção sobre DevOps abrange suas práticas, origens e princípios, enquanto a qualidade de software é abordada com ênfase nos conceitos, custos, controle e testes automatizados.

Por fim, o quarto capítulo conclui o trabalho, destacando os benefícios derivados da adoção das práticas DevOps.

## 3.1 Desenvolvimento de Software

Existem diversas metodologias disponíveis no mercado para o desenvolvimento de software. O DevOps surge como uma abordagem destinada a otimizar e aprimorar esse processo. Portanto, é crucial compreender como os processos de desenvolvimento de software e suas metodologias podem incorporar o DevOps de maneira eficiente.

Os procedimentos de desenvolvimento de software têm evoluído ao longo do tempo, adaptando-se e melhorando para atender à crescente dependência estratégica das empresas em relação aos softwares. Diante dessa importância, é imperativo dedicar uma atenção especial aos processos que podem aprimorar e otimizar o desenvolvimento de software.

A definição de um processo de desenvolvimento de software, conforme Pressman (2011), engloba um conjunto de atividades, ações e tarefas realizadas para criar uma solução específica. No contexto do desenvolvimento de software, essa abordagem altera a perspectiva desses processos, atribuindo à equipe de desenvolvimento a responsabilidade pela escolha do processo. Essa responsabilidade visa garantir a entrega do software com qualidade suficiente para satisfazer as necessidades daqueles que o patrocinam.

Uma metodologia genérica de processo de desenvolvimento de software estabelece uma base para as atividades no ciclo de desenvolvimento do software, envolvendo cinco etapas: Comunicação, Planejamento, Modelagem, Construção e Emprego (PRESSMAN, 2011). Esse conjunto de atividades caracteriza um modelo clássico de desenvolvimento, seguindo uma abordagem cascata, onde cada etapa só se inicia após a conclusão da anterior. Contudo, é importante observar que esse modelo é considerado inflexível, pois limita a divisão do projeto em fases distintas.

### 3.1.1 Ciclo de Desenvolvimento de Software

O processo de desenvolvimento de software compreende várias etapas, sendo crucial ter um mapeamento claro, pois existem aquelas em que a equipe de desenvolvimento atua e outras em que as operações desempenham seu papel.

Em um cenário típico, o desenvolvimento de software é subdividido em seis etapas distintas, cada uma com seu responsável e equipe específica encarregada da execução. A fim de guiar de forma coordenada essas fases, recorremos à aplicação de uma metodologia de desenvolvimento de software.

A etapa de planejamento, a primeira do ciclo, tem como responsabilidade agendar recursos humanos e datas para a realização do software. A análise, segunda fase do processo, detalha as funcionalidades. Em seguida, temos a fase de desenvolvimento do software. Após essa etapa, inicia-se a fase de testes, que pode ser executada simultaneamente ao desenvolvimento. Ao concluir o desenvolvimento e os testes, ocorre a entrega do software às partes interessadas. A fase de manutenção do sistema encerra o ciclo de desenvolvimento de software.

### 3.1.2 Metodologia Ágil

Com base no Manifesto Ágil (BECK; BEEDLE; BENNEKUM, 2018), foi identificada uma abordagem alternativa para compreender o desenvolvimento de software, centrada na valorização dos seguintes princípios:

- Priorização de indivíduos e interações em detrimento de processos e ferramentas.
- Ênfase em software em funcionamento em vez de documentação abrangente.
- Foco na colaboração com o cliente em oposição à negociação de contratos.
- Preferência por responder a mudanças em vez de seguir um plano predefinido.

Para atender a esses princípios delineados no Manifesto Ágil, surgiram diversos frameworks de desenvolvimento e gestão de software. A próxima seção abordará os frameworks Scrum e Extreme Programming.

#### 3.1.2.1 SCRUM

O SCRUM, de acordo com Schwaber e Sutherland (2013), é um framework amplamente empregado no desenvolvimento de software. Criado para gerenciar eficientemente os fluxos de trabalho na concepção de produtos, o SCRUM teve sua origem no final dos anos 80.

Este framework, também emprega diversos processos e técnicas com o objetivo de gerenciar e aprimorar o desenvolvimento do software de maneira interativa e incremental, promovendo uma melhoria contínua ao longo do ciclo de desenvolvimento do produto (SCHWABER E SUTHERLAND, 2013).

O modelo de trabalho do SCRUM fundamenta-se em equipes pequenas, caracterizadas por sua autogestão e compromisso com os pilares essenciais do SCRUM: Transparência, Inspeção e Adaptação (SCHWABER E SUTHERLAND, 2013).

#### 3.1.2.2 Extreme Programming

O Extreme Programming (XP) é um método de desenvolvimento de software que incorpora os princípios da abordagem ágil.

De acordo com Teles (2004), o Extreme Programming (XP) é particularmente indicado para cenários de desenvolvimento de software em que os requisitos não estão claramente definidos e sofrem alterações frequentes. Ele utiliza a orientação a objetos como modelo de desenvolvimento do software, envolve equipes pequenas, com um limite de até 12 desenvolvedores, e adota uma abordagem incremental. Isso significa que, desde o início do projeto, o sistema é implementado e recebe novas funcionalidades ao longo do tempo.

Para implementar esse modelo, a empresa deve incorporar alguns valores essenciais, tais como Feedback, Comunicação, Simplicidade e Coragem. Esses valores são fundamentais para uma adoção eficaz deste método de desenvolvimento de software.

## 3.2 *DevOps* - Desenvolvimento e Operações

Com a crescente necessidade de desenvolver soluções de software de maneira mais ágil para atender rapidamente às demandas do mercado, as metodologias ágeis se tornaram predominantes no cenário de desenvolvimento de software.

Na abordagem ágil, a filosofia central é a colaboração estreita entre desenvolvedores e clientes, visando direcionar o desenvolvimento do software de acordo com as reais necessidades do cliente. Isso permite que o software evolua como um produto que o cliente pode acompanhar e moldar ao longo do desenvolvimento. Essa flexibilidade contrasta com os modelos de desenvolvimento tradicionais, nos quais os softwares eram construídos a partir de especificações extensas, resultando em entregas volumosas com inúmeras funcionalidades.

Entretanto, com a adoção do desenvolvimento ágil, as empresas e equipes de T.I. enfrentaram desafios no ciclo de vida do software. À medida que as interações entre as equipes de desenvolvimento e os clientes aumentavam, era necessário que as entregas seguissem o mesmo ritmo. Foi nesse contexto que surgiu o DevOps.

DevOps é um termo que combina "Desenvolvimento e Operações", com o objetivo de unificar essas partes no processo de desenvolvimento de software (ZENTGRAF, 2012). Para adotar o DevOps, é necessário uma mudança cultural na organização, buscando uma colaboração efetiva entre as áreas de negócios, operações e desenvolvimento.

### 3.2.1 Práticas DevOps

Com a emergência do conceito de DevOps e a intensificação da competitividade nos ambientes corporativos, observa-se a adoção de processos destinados a aprimorar os ambientes de desenvolvimento de software.

O termo DevOps passou a ser vinculado a diversos conceitos nesses ambientes de desenvolvimento, destacando-se a Integração Contínua (*Continuous Integration*), a Implementação Contínua (*Continuous Deployment*), e as Entregas Contínuas (*Continuous Delivery*) (RATO, 2017). Esses elementos constituem uma base para identificar a presença de DevOps nos ambientes de desenvolvimento, visto que o DevOps representa uma abordagem para integrar tanto processos quanto pessoas nesses ambientes.

Ao implementar esses três termos, torna-se possível efetivar a integração e o comprometimento das partes envolvidas nos processos de desenvolvimento. Dessa

forma, é viável alcançar os objetivos de entrega de software com maior precisão e eficiência.

#### 3.2.2.1 Integração Contínua

A prática de Integração Contínua se concentra na integração entre o código desenvolvido e os testes, e frequentemente visa concretizar outra prática, a Entrega Contínua (RATO, 2017).

O processo associado a essa prática envolve a fusão de códigos de software desenvolvidos com uma frequência elevada, podendo ocorrer várias fusões ao longo de um dia de desenvolvimento. Posteriormente, essas fusões têm potencial para se transformar em entregáveis de software, dependendo do feedback da equipe de desenvolvimento (RATO, 2017).

#### 3.2.2.2 Entrega Contínua

Uma prática de extrema relevância no ciclo de vida do software é a Entrega Continua, desempenhando um papel crucial no acompanhamento do desenvolvimento do software. De acordo com Sharma e Coyne (2015), essa prática possibilita que as funcionalidades desenvolvidas no software estejam disponíveis para os usuários o mais rapidamente possível.

Essa prática ganha destaque por ter originado o movimento DevOps. Esse surgimento ocorreu porque a Integração Contínua proporciona uma evolução na automação dos processos de software, permitindo a implantação automatizada de um software. É crucial, nesse contexto, implementar testes automatizados, pois isso contribui para uma detecção mais ágil de problemas (LEVITA; NETO, 2018).

#### 3.2.2.3 Implementação Contínua

No desfecho do desenvolvimento do software, inicia-se a fase de implementação e entrega. Com a utilização de um modelo automatizado, a implementação contínua possibilita que todo o software, após passar pelos processos de testes automatizados, seja automaticamente encaminhado ao sistema de produção, marcando, assim, a conclusão do ciclo de desenvolvimento do software (HUMBLE; FARLEY, 2015).

Essa prática se reproduz ao longo do ciclo de vida do software, representando a última etapa sob responsabilidade da equipe de desenvolvimento. Além disso, é válido considerar que, ao término desse processo, o cliente pode perceber um valor agregado no software, pois é nesse momento que ele começa a se tornar útil para o cliente.

#### 3.2.2.4 Monitoramento Contínuo

A prática de Monitoramento Contínuo desempenha um papel crucial no acompanhamento da evolução de um sistema. Essa abordagem possibilita que métricas relacionadas ao desempenho e desenvolvimento do software estejam acessíveis aos envolvidos durante e após o processo de desenvolvimento. Dessa forma, é viável obter uma compreensão clara do progresso do projeto, fornecendo insights para avaliar o desempenho dos processos e apoiar a tomada de decisões diante de possíveis obstáculos (LEVITA; NETO, 2018).

### 3.2.3 Princípios do DevOps

Conforme apresentado no projeto *The Phoenix Project* (KIM; BEHR; SPAFFORD, 2013), há três princípios fundamentais para a incorporação do DevOps em uma metodologia de desenvolvimento de software.

O primeiro princípio concentra-se na criação de um modelo de trabalho otimizado que se alinhe tanto com a equipe de desenvolvimento quanto com as operações. Nesse sentido, a abordagem propõe a quebra dos processos em partes menores, permitindo uma otimização gradual do fluxo de trabalho global, priorizando o controle e a segurança das mudanças ao longo do processo.

O segundo princípio preconiza a adoção de uma cultura de feedback contínuo, em forma de loops. Esses loops de feedback têm o propósito de corrigir as fontes de desenvolvimento, prevenindo problemas e retrabalho. Além disso, esses loops contribuem para a qualidade intrínseca do processo, destacando as partes que estão ativamente envolvidas no desenvolvimento e nas operações.

O terceiro princípio promove uma cultura que encoraja a experimentação e a mentalidade de que a repetição dos processos é crucial para adquirir domínio. Essa cultura estimula a ação prática em vez de uma análise aprofundada.

### 3.2.4 DevOps e a Otimização do Desenvolvimento do Software

Dentro das fases do ciclo de desenvolvimento de software, há três estágios nos quais a integração com o DevOps pode ser aplicada: desenvolvimento, testes e entrega.

Nesses estágios, o DevOps está associado à automação de processos rotineiros. Diversas ferramentas são utilizadas para implementar essa automação, sendo duas notáveis no mercado:

1. Microsoft Team Foundation Server
2. Jenkins

Ao serem gerenciadas pela equipe de operações, essas ferramentas viabilizam a implementação das práticas do DevOps.

## 3.3 Qualidade de Desenvolvimento de Software

A importância da qualidade no desenvolvimento de software tem crescido significativamente, tornando-se um tema crucial nos processos de criação de programas. A precisão nas entregas de um produto de software e em suas implementações desempenha um papel fundamental na melhoria da experiência do usuário.

A abordagem da qualidade relacionada ao software se manifesta em dois momentos distintos: na qualidade do desenvolvimento do software e na qualidade do produto final. Para alcançar esses dois padrões, são necessários processos específicos ao longo do ciclo de desenvolvimento.

A norma ISO 9000 proporciona uma orientação para a gestão da qualidade e auxilia na implementação de um modelo de qualidade reconhecido no mercado brasileiro, abrangendo diretrizes para a qualidade de software.

Os padrões ISO 9001, 9002 e CMM são modelos principais recomendados para atender às diretrizes de qualidade nos processos relacionados ao desenvolvimento de software (RINCONM, 2006).

Para efetivar a implementação da qualidade nesses processos, é possível destacar alguns pontos de atenção, conforme apontado por Bueno (2006):

- Definição da qualidade
- Controle de qualidade
- Garantia de qualidade
- Custo da qualidade

### 3.3.1 Definição de Qualidade de Software

De acordo com a Norma ISO 9000, os clientes demandam que as características de seus produtos satisfaçam suas necessidades e expectativas. Essa satisfação é alcançada quando o produto atende aos requisitos estabelecidos previamente pelos clientes (ABNT, 2005).

Partindo desse princípio, o software pode ser considerado um produto que deve corresponder às expectativas do cliente. Quando isso ocorre, o software alcança um nível de qualidade.

### 3.3.2 Controle de Qualidade

A gestão da qualidade ao longo do processo de desenvolvimento implica a integração de diversas ações para assegurar a conformidade com padrões e procedimentos estabelecidos (RODRIGUES, 2015).

Para efetuar esse controle, são realizadas revisões, inspeções e testes de forma contínua durante o desenvolvimento do software. É viável conduzir o controle de qualidade com a participação do cliente e com base na documentação das regras de negócio. Qualquer aspecto que não esteja em conformidade com os critérios de qualidade esperados pelo cliente deve ser considerado um ponto de atenção para a equipe de gestão do projeto (RODRIGUES, 2015).

O objetivo principal do controle de qualidade é garantir a estabilidade do software na entrega final do projeto. Uma entrega de qualidade contribui para que o custo do projeto permaneça dentro das estimativas, uma vez que a correção de problemas após a conclusão do software pode resultar em prejuízos financeiros.

### 3.3.3 Custo da Qualidade

Um software possui uma característica de extrema importância que deve ser considerada durante o planejamento e controle do custo de sua qualidade. É praticamente impossível entregar um software completamente isento de bugs (RODRIGUES, 2015).

É comum que um software apresente problemas, portanto, é necessário definir o nível de qualidade que o software deve alcançar para atender às necessidades e expectativas do cliente.

Os custos relacionados à qualidade do software podem ser classificados como custo da falha, custo da prevenção e custo da análise, de acordo com Rodrigues (2015). O custo da falha é mais desafiador de prever, pois depende do tamanho da falha e do impacto que ela tem nos usuários do software. O custo da prevenção é um orçamento que deve prever ações para garantir a qualidade do software. O custo da análise está vinculado à garantia da qualidade do software, dependendo das ações tomadas durante o desenvolvimento, como os testes para identificação de erros (RODRIGUES, 2015).

### 3.3.4 Qualidade do Produto de Software

Para compreender e avaliar a qualidade de um produto de software, conforme o Programa Brasileiro de Qualidade e Produtividade em Software (CRUZ; ANDRADE; FIGUEIREDO, 2010), é necessário percorrer alguns critérios de avaliação para determinar o grau de qualidade do produto.

Com o apoio do Ministério da Ciência e Tecnologia e da Secretaria de Política de Informática, o PBQP (Programa Brasileiro de Qualidade e Produtividade em Software) promove iniciativas focadas na melhoria dos processos, projetos e

serviços de software. Para avaliar a qualidade dos produtos, o PBQP de software adota critérios de avaliação com conjuntos específicos de atributos e pesos (CRUZ; ANDRADE; FIGUEIREDO, 2010). Esses critérios incluem:

- Relevância
- Impacto
- Abrangência
- Inovação
- Qualidade da Apresentação do Artigo
- Resultados Obtidos pelo Projeto

Ao analisar cada critério, considerando seus pesos, é possível mensurar o grau de qualidade ao qual o produto se adequa em relação ao programa. Importante destacar que a avaliação da qualidade dos produtos de software segue as normas da série NBR ISO/IEC 9126 (Tecnologia de Informação – Avaliação de Produto de Software – Características de qualidade e diretrizes para o seu uso) e NBR ISSO/IEC 14598 (Tecnologia de Informação - Pacotes de software - Testes e requisitos de qualidade). Essas normas estão em processo de substituição pelas diretrizes da série NBR ISO/IEC 25000 - Engenharia de Software - Requisitos e avaliação da qualidade de produtos de software (SQuaRE) (CRUZ; ANDRADE; FIGUEIREDO, 2010).

### 3.3.5 Testes Automatizados

Uma característica presente no modelo de desenvolvimento ágil é a prática de entregas contínuas. Essas entregas visam desenhar o software em pequenas partes e realizar entregas menores para atender às demandas do negócio à medida que as mudanças no mercado ocorrem.

No contexto dessa entrega contínua, destaca-se a automação de testes, uma prática amplamente aceita no modelo ágil. Os testes automatizados consistem em programas ou scripts desenvolvidos em conjunto com o software, utilizados para exercitar rapidamente as funcionalidades do sistema. Uma vantagem desse modelo é a capacidade de percorrer todos os casos de teste previamente estabelecidos (LIMA; DANTAS; VASCONCELOS, 2012).

A repetição manual desses testes pode propiciar erros, mas quando executada por uma ferramenta de automação, é possível evitar esses erros e identificá-los de maneira mais rápida. Além disso, a utilização desse modelo contribui para a garantia da qualidade na manutenção do software, pois já incorpora todos os casos de testes automatizados.

A prática de garantir entregas com qualidade, resultado da automação de testes, é uma estratégia do DevOps que se alinha com a ideia de entrega contínua. Conforme descrito na ISO 9000, o estabelecimento de métodos para medir a gestão da qualidade, a alocação de recursos necessários para garantia da qualidade e a definição de processos e responsabilidades são premissas fundamentais para a implementação da automação de testes (ABNT, 2005).

A integração entre testes e desenvolvimento de software pode ser abordada de diversas maneiras. Duas abordagens notáveis nesse cenário são o Desenvolvimento Orientado a Testes (TDD) e o Desenvolvimento Orientado ao Comportamento (BDD). Esses modelos representam uma tendência no desenvolvimento ágil, alinhados ao conceito de DevOps (ERICH; AMRIT; DANEVA, 2014).

#### 3.3.5.1 Desenvolvimento Orientado a Testes (TDD)

Com a adoção do desenvolvimento ágil, especialmente o método Extreme Programming, surgiu uma abordagem de desenvolvimento de software denominada Desenvolvimento Orientado a Testes (Test-Driven Development - TDD), similar ao desenvolvimento orientado a objetos.

Essa metodologia envolve a criação de testes automatizados antes da implementação do código funcional (JANZEN; SAIEDIAN, 2005). Todo o desenvolvimento do software é orientado para atender aos requisitos dos testes automatizados previamente escritos. Com esse modelo, a qualidade do software é aprimorada, e a confiança na eficácia dos testes automatizados é reforçada.

#### 3.3.5.2 Desenvolvimento Orientado a Comportamento (BDD)

O TDD é uma abordagem eficaz para realizar testes de software; no entanto, enfrenta desafios durante a implementação. Como os desenvolvedores são responsáveis por criar esses testes, pode ser complexo abstrair a essência de uma funcionalidade para aplicar esse modelo.

Diante dos obstáculos enfrentados pelo TDD, o Behavior Driven Development (BDD) surgiu para superá-los. O BDD é uma maneira simplificada de realizar testes de aceitação de software, começando com um ponto de partida e um resultado final. Esse modelo busca proporcionar um entendimento comum a todos os membros do projeto sobre os testes que serão realizados no software, utilizando uma linguagem natural nos testes de software (SOEKEN; WILLE; DRECHSLER, 2012).

O BDD opera por meio de cenários, descritos pela equipe de testes e implementados pelos desenvolvedores. Esses cenários devem conter informações suficientes para cobrir a maioria das possibilidades de uma funcionalidade ou fluxo de software. Com os cenários definidos, a automação dos testes pode ser desenvolvida, proporcionando melhor desempenho e agilidade no desenvolvimento do software.

## 4. CONSIDERAÇÕES FINAIS

A presente pesquisa buscou abordar a seguinte indagação: De que forma as práticas do DevOps podem influenciar positivamente na qualidade do desenvolvimento de software? Ao explorar os tópicos neste estudo, procurou-se oferecer uma visão genérica sobre o desenvolvimento de software e a metodologia ágil, elementos cruciais para compreender a interação do DevOps no ciclo de vida de um software. A qualidade do software foi examinada de maneira abrangente devido ao seu papel como principal diferencial competitivo nos produtos de software.

Para enriquecer a pesquisa, foi apresentado um estudo de caso que comparou dois cenários: um sem as práticas do DevOps e outro com a sua implementação. Esse estudo destacou os benefícios que o DevOps pode proporcionar ao desenvolvimento de software.

O tema DevOps é relativamente recente, mas mostra-se promissor ao introduzir novas possibilidades na construção de softwares. Suas práticas estão alinhadas à arquitetura de um sistema, baseando-se nos princípios da colaboração, interação e automatização. Ao contrário de uma metodologia de desenvolvimento de software, não existem padrões rígidos para sua implementação, associando-se mais a uma filosofia do que a um modelo estrito de desenvolvimento.

Como sugestões para estudos futuros, recomenda-se investigar como o DevOps está influenciando a substituição de postos de trabalho, por meio da automação intensiva dos processos de desenvolvimento de software. Além disso, seria interessante explorar os impactos colaterais, como a robotização de testes e desenvolvimento, identificados empiricamente nesta pesquisa.

**REFERÊNCIAS**


ABNT (2005). NBR ISO 9000-2005 - Sistemas de gestão da qualidade - Fundamentos e vocabulário. Rio de Janeiro, Sede da ABNT, 2005.

BECK, Kent.; BEEDLE, Mike.; BENNEKUM, Arie van (2018). Manifesto para Desenvolvimento Ágil de Software. Disponível em: <http://agilemanifesto.org/iso/ptbr/manifesto.html> Acesso em Julho de 2018.

BUENO, Cassiane de Fátima dos; CAMPELO, Gustavo Bueno. Qualidade de Software. Recife.

CRUZ, Cláudio Silva da; ANDRADE Edméia Leonor Pereira de; FIGUEIREDO, Rejane Maria da Costa. Programa Brasileiro da Qualidade e Produtividade em Software. Brasilia: PBQP Software, 2008. cap 2.

ERICH, Floris; AMRIT, Chintan; DANEVA, Maya. DevOps Literature Review. Enschede, 6, Outubro de 2014.

HUMBLE, Jezz; FARLEY, David. Continuous Delivery. Boston: Person Education, 2011.

JANZEN, David; SAIEDIAN, Hossein (2005). Test-Driven Development: Concepts, Taxonomy,and Future Direction. IEEE Computer. Setembro, 2005.

KIM, Gene; BEHR, Kevin; SPAFFORD, George. The Phoenix Project: A Novel About IT, DevOps, and Helping Your Business Win. Portland: IT Revolution Press, 2013.

LEVITA, Carlos de Amorim.; NETO, João Augusto Mattar. Proposta de Modelo para Avaliação da Maturidade DevOps. São Paulo, 06 de Novembro de 2017.

LIMA, TLL; DANTAS, Ayla; VASCONCELOS, Livia M. R. Usando o SilkTest para automatizar testes: um Relato de Experiência. João Pessoa, 2012.

PRESSMAN, Roger S.. Engenharia de Software - Uma Abordagem Profissional - 7° Edição. Porto Alegre: Bookman, 2011. Cap 1, p. 29-51.

RATO, Francisco. DevOps em Sistemas de Informação: Implementação em Operações de Tecnologias de Informação. 2017. Universidade Nova de Lisboa, Lisboa, 2017.

RINCONM, André Mesquita. Qualidade de Software. Goiânia, 2006.

RODRIGUES, Milena (2015). Introdução ao Gerenciamento de Qualidade. DEVMEDIA. Disponível em: <https://www.devmedia.com.br/introducao-aogerenciamento-de-qualidade/33435> Acesso em Outubro de 2018.

SCHWABER, Ken; SUTHERLAND, Jeff. Um guia definitivo para o Scrum: As regras do jogo. 2013.

SHARMA, Sanjeev; COYNE, Bernie. DevOps for Dummies. Hoboken: John Wiley e Sons, 2015.

SMARTSHEET (2018). The Way of DevOps: A Primer on DevOps Principles and Practices. Disponível em: <https://www.smartsheet.com/devops> Acesso em Setembro de 2018.

SOEKEN, Mathias; WILLE, Robert; DRECHSLER, Rolf. Assisted behavior driven development using natural language processing. Bremen, 2012.

TELES, Vinícius Manhães. Extreme Programming : Aprenda como encantar os seus usuários desenvolvendo software com agilidade e alta qualidade. Novatec, 2004. cap 1, p 21-29.

ZENTGRAF, Dan. Definindo a implementação distribuível em DevOps. 23, Outubro De 2012.

Extreme Programming (2013). Extreme Programming: A Gentle Introduction. Disponível em <http://www.extremeprogramming.org/> Acesso em Outubro de 2018.

MUNIZ, Antonio et al. Jornada DevOps: Unindo Cultura ágil, Lean e tecnologia para entrega de software com qualidade. Brasport, 2019. Disponível em: https://books.google.com.br/books?hl=pt-BR&lr=lang_pt&id=94qXDwAAQBAJ&oi=fnd&pg=PA3&dq=devops&ots=sFqHWUCIuF&sig=CgbzoyjyKlfBy0tQWGsK5JkhDgY&redir_esc=y#v=onepage&q&f=false

SILVA, Flávio Henrique Rocha. A ADOÇÃO DO GERENCIAMENTO DE SEGREDOS PARA PROTEÇÃO DE ATIVOS SENSÍVEIS NAS PRÁTICAS DEVOPS. 2021. Disponível em: https://bibliotecadigital.stf.jus.br/xmlui/bitstream/handle/123456789/2594/Flavio%20Henrique_Artigo_Cientifico_Final.pdf?sequence=1&isAllowed=y

DE MENDONÇA, Cláudio Márcio Campos; DE SOUSA NETO, Manoel Veras. Serviço da computação em nuvem e sua relação com os arranjos de Governança de TI e o alinhamento estratégico. REVISTA DE TECNOLOGIA APLICADA, v. 8, n. 2, p. 41-62,2020. Disponível em: https://www.cc.faccamp.br/ojs-2.4.8-2/index.php/RTA/article/view/1383/679

SOARES, Mateus Oliva et al. Solução Integrada para Análise e Monitoramento de Redes Empresariais. In: Anais Estendidos do XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais. SBC, 2022. p. 264-271. Disponível em: https://sol.sbc.org.br/index.php/sbseg_estendido/article/view/21716