

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Bruno dos Santos Sicchieri

**O papel da Garantia da Qualidade no processo
de desenvolvimento de software em equipes
ágeis**

Uberlândia, Brasil

2025

UNIVERSIDADE FEDERAL DE UBERLÂNDIA

Bruno dos Santos Sicchieri

O papel da Garantia da Qualidade no processo de desenvolvimento de software em equipes ágeis

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. Ronaldo Castro de Oliveira

Universidade Federal de Uberlândia – UFU

Faculdade de Computação

Bacharelado em Sistemas de Informação

Uberlândia, Brasil

2025

Bruno dos Santos Sicchieri

O papel da Garantia da Qualidade no processo de desenvolvimento de software em equipes ágeis

Trabalho de conclusão de curso apresentado à Faculdade de Computação da Universidade Federal de Uberlândia, como parte dos requisitos exigidos para a obtenção título de Bacharel em Sistemas de Informação.

Prof. Dr. Ronaldo Castro de Oliveira
Orientador

Profa. Dra. Maria Adriana Vidigal de Lima

Prof. Dr. Paulo Henrique Ribeiro Gabriel

Uberlândia, Brasil
2025

Dedico este trabalho ao meu avô, que, infelizmente, não está mais presente fisicamente em minha vida. Espero que, onde quer que esteja, sinta orgulho de mim.

Agradecimentos

Aos meus padrinhos, por sempre acreditarem em mim e embarcarem comigo nesta jornada tão importante da minha vida. Obrigado por me apoiarem, acompanharem e torcerem por mim de perto. Se não fosse por vocês, essa caminhada talvez nem tivesse começado. Essa conquista é nossa.

À minha família, por estarem sempre ao meu lado e me incentivar a seguir firme. Estando perto ou longe, vocês sempre se fizeram presentes na minha vida. Acreditaram no meu potencial e me mostraram que sou capaz de superar qualquer desafio. Estou aqui por vocês.

Ao meu namorado e aos amigos, por serem meu apoio e por confiarem em mim acima de tudo. Vocês estiveram comigo quando muitos não puderam estar, sempre segurando a minha mão, oferecendo palavras de apoio e acreditando que esse momento chegaria. Obrigado por tudo.

Ao meu orientador, por ter aceitado o desafio de me acompanhar nesse último e importante passo antes da formatura. Sua orientação foi essencial para o sucesso deste trabalho. Muito obrigado.

Aos professores que compõem a banca, agradeço por aceitarem o convite e estarem presentes neste momento tão significativo da minha trajetória pessoal e acadêmica. Não consigo expressar em palavras o quão desafiador foi chegar até aqui, por diversos motivos, o que torna este momento ainda mais especial.

Aos meus professores ao longo da graduação, meu sincero agradecimento por todo o conhecimento compartilhado. A dedicação e o esforço de cada um foram notados e valorizados.

Por fim, deixo um agradecimento especial ao Bruno, a versão de mim mesmo de alguns anos atrás, que teve a coragem de sair da zona de conforto, deixar para trás aqueles que mais ama e embarcar em uma jornada de crescimento e aprendizado longe de casa. Se não fosse pelas suas escolhas, persistência e comprometimento, eu não estaria vivendo esse momento.

Resumo

A garantia da qualidade tem sido cada vez mais relevante no desenvolvimento de software, na busca por produtos melhores. Diante desse cenário, este trabalho propôs analisar o processo atual de qualidade de uma equipe ágil de backend e apresentar melhorias voltadas à eficiência e confiabilidade das entregas. Realizou-se um estudo de caso em uma empresa do setor de tecnologia, por meio de uma entrevista com o analista de qualidade da equipe, com o objetivo de levantar as práticas de garantia da qualidade em uso no projeto. Com base nessa análise, foi elaborada uma proposta de melhorias com foco em automação de testes, integração contínua e maior colaboração do time, a partir da redefinição de responsabilidades. O estudo reforça a importância da atuação ativa da garantia da qualidade em todas as etapas do ciclo de desenvolvimento, e espera-se que a adoção da proposta proporcione aumento na cobertura de testes, redução de retrabalho e melhoria na comunicação entre os membros da equipe.

Palavras-chave: Garantia da qualidade, Desenvolvimento de software, Equipes ágeis, Testes automatizados, Integração contínua.

Lista de ilustrações

Figura 1 – Framework Scrum	16
Figura 2 – Metodologia Kanban	19
Figura 3 – Pirâmide de Testes	26
Figura 4 – Funcionamento do CI/CD em DevOps	29
Figura 5 – Diagrama do fluxo atual de Garantia da Qualidade	35
Figura 6 – Diagrama do fluxo proposto para a Garantia da Qualidade	45

Lista de abreviaturas e siglas

TI: Tecnologia da Informação

ABES: Associação Brasileira das Empresas de Software

QA: Garantia da Qualidade (Quality Assurance)

CI: Integração Contínua (Continuous Integration)

CD: Entrega Contínua (Continuous Delivery)

API: Interface de Programação de Aplicações (Application Programming Interface)

PO: Product Owner

E2E: Teste de Ponta a Ponta (End-to-End)

XP: Extreme Programming

TDD: Desenvolvimento Orientado por Testes (Test-Driven Development)

PRs: Pull Requests

HTTP: Protocolo de Transferência de Hipertexto (Hypertext Transfer Protocol)

Sumário

1	INTRODUÇÃO	10
1.1	Justificativa	11
1.2	Objetivos	12
1.3	Método	12
1.4	Organização do Trabalho	13
2	REVISÃO BIBLIOGRÁFICA	14
2.1	Processo de desenvolvimento de software	14
2.2	Métodos Ágeis	15
2.2.1	Scrum	15
2.2.1.1	Time Scrum	16
2.2.1.2	Eventos Scrum	17
2.2.1.3	Artefatos do Scrum	17
2.2.2	Kanban	18
2.2.3	Extreme Programming	19
2.3	Garantia da qualidade	20
2.3.1	O papel do analista de qualidade tradicional	20
2.3.2	O papel do analista de qualidade ágil	21
2.3.3	Revisão de Código (Code Review)	21
2.3.4	Métricas e Relatórios	22
2.3.5	Gestão de Documentos	22
2.4	Testes de Software	23
2.4.1	Testes Funcionais	24
2.5	Testes Automatizados	26
2.5.1	Testes Não Funcionais	27
2.6	DevOps	27
2.6.1	Integração Contínua (Continuous Integration - CI)	28
2.6.2	Entrega Contínua (Continuous Delivery - CD)	28
2.7	Trabalho Relacionados	29
3	ANÁLISE DO PROCESSO DE GARANTIA DA QUALIDADE NA EMPRESA OMEGA	31
3.1	Caracterização da empresa e do projeto	31
3.2	Processo de garantia da qualidade	32
3.3	Avaliação do impacto das práticas de garantia da qualidade no projeto	36

4	PROPOSTA DE MELHORIAS PARA O PROCESSO DE GARANTIA DA QUALIDADE NA EMPRESA OMEGA	38
4.1	Pontos focais do processo de garantia da qualidade	38
4.1.1	Automação de testes funcionais	38
4.1.2	Integração contínua e Entrega Contínua (CI/CD)	38
4.1.3	Revisão de código (Code Review)	39
4.1.4	Gestão do conhecimento e documentação	39
4.1.5	Análise de métricas	40
4.2	Proposta de melhorias do processo de garantia da qualidade para a Omega	41
4.3	Plano de ação sugerido para implementação da proposta	46
4.4	Resultados esperados e possíveis lições aprendidas	47
5	CONCLUSÃO	49
	REFERÊNCIAS	51

1 Introdução

Os softwares empresariais têm-se tornado fundamentais para garantir um mercado competitivo e organizações eficientes ([SARATKAR, 2019](#)) em um mundo onde a Tecnologia da Informação (TI) se tornou uma das partes mais importantes, provedora de inúmeras oportunidades de sucesso para as empresas que a souberem aproveitar ([ROGER; LIMA, 2015](#)).

De acordo com a Associação Brasileira das Empresas de Software (ABES), em 2022, o mercado de software apresentou um crescimento de 7,9%, com mais de 8 mil empresas atuando no desenvolvimento de softwares, demonstrando ser uma área em expansão no país ([SOFTWARE, 2023](#)). Com o setor em crescimento, a competitividade do mercado e a satisfação do cliente tornaram-se preocupações crescentes para as empresas, que buscam cada vez mais aprimorar a qualidade dos softwares desenvolvidos ([INMETRICS, 2023](#)).

A busca pela qualidade é um processo colaborativo, no qual todas as partes interessadas se unem para garantir que o produto final atenda aos requisitos e necessidades do cliente. Esse princípio é evidenciado na prática por equipes que adotam metodologias ágeis, onde a alta qualidade das entregas é uma responsabilidade coletiva e a equipe implementa as melhores práticas para alcançar esse propósito ([GANTAYAT, 2022](#)). O teste de software é uma prática fundamental adotada pelas equipes ágeis para aprimorar o produto, detectando defeitos e prevenindo problemas futuros ([ISTQB, 2023](#)). Como resultado, há uma economia de custos, dado que erros encontrados no processo de implantação podem aumentar os custos do projeto em até 60% ([BARBOSA; TORRES, 2011](#)). Além disso, durante a criação do produto, é importante focar não apenas em sua funcionalidade, mas também em aspectos como usabilidade, eficiência, entre outros. A área de Garantia da Qualidade, ou Quality Assurance (QA), é responsável por gerenciar as ações necessárias para assegurar a qualidade de um produto ou serviço, incluindo as atividades de teste em todas as fases da criação, até o acompanhamento pós-implantação ([GAEA, 2018](#)).

Embora muitas vezes passe despercebido, o uso de softwares é constante na rotina das pessoas, presente em aparelhos celulares, automóveis, televisores e até mesmo nos pagamentos realizados com cartões. Interrupções em serviços, como os bancários, por exemplo, podem acarretar problemas tanto para os usuários quanto para as instituições provedoras desses serviços. Em agosto de 2023, o maior banco privado do Brasil enfrentou problemas em suas plataformas, com diversos sistemas sendo afetados por instabilidades, dificultando o acesso e os pagamentos. Na ocasião, a instituição foi notificada pelos

órgãos responsáveis de que deveria avaliar a compensação de prejuízos financeiros, caso comprovados pelos consumidores (NOVAES, 2023). Desse modo, torna-se evidente que as empresas precisam se preocupar e investir cada dia mais em padrões de qualidade elevados nos processos de desenvolvimento de software.

Em "Quality Assurance Activities in Agile" (ULLAH; WAQAR; ZAIDI, 2009), os autores contextualizam a importância do gerenciamento de qualidade para as organizações, garantindo a conquista dos padrões esperados nos produtos entregues e em todo o ciclo de desenvolvimento. As atividades de garantia, planejamento e controle da qualidade são apontadas como essenciais para o gerenciamento da qualidade. O objetivo do estudo é descrever as atividades executadas no método ágil para assegurar a qualidade, abordando o desenvolvimento ágil de software com base nos princípios do manifesto ágil, como o Extreme Programming (XP), o Scrum e o Crystal Clear Methodology. Os autores também discutem os desafios ao adotar o desenvolvimento ágil, destacando práticas como o Test-Driven Development (TDD) para os testes, programação em pares e refatoração. Após explorar esses temas, concluem propondo uma redefinição no papel das práticas de garantia da qualidade em projetos ágeis, ressaltando a importância de focar na qualidade do processo e do produto. O trabalho dos autores se relaciona diretamente com a pesquisa em andamento, abordando temas como processo de desenvolvimento, modelos ágeis, aplicação de práticas e propostas de melhoria do processo.

Nesse sentido, este estudo busca identificar as ações e atividades que podem ser realizadas pela área de Garantia da Qualidade no contexto do desenvolvimento ágil, contribuindo para a melhoria contínua das entregas. Por meio de um estudo de caso em uma empresa que ainda não adota integralmente essas práticas, analisou-se o processo atual, seus desafios e oportunidades de melhoria, e elaborou-se um plano de ação para a adoção de novas práticas alinhadas aos princípios da Garantia da Qualidade ágil.

1.1 Justificativa

Devido à crescente importância da qualidade do software em um mercado competitivo e exigente, garantir níveis elevados de qualidade em ambientes de desenvolvimento ágil, com entregas rápidas e frequentes, torna-se essencial para manter a satisfação dos clientes e a reputação das empresas. A busca pela agilidade e a pressão por inovação, como ocorre na organização estudada, demonstram a necessidade de prevenir defeitos que possam impactar negativamente seus serviços e resultados.

Reconhecendo a importância da melhoria contínua, a equipe analisada busca constantemente aprimorar a eficiência e confiabilidade de suas entregas. Nesse contexto, a análise detalhada do processo atual de garantia da qualidade e a identificação de oportunidades de melhoria, com foco em automação de testes, integração contínua e colaboração

do time, mostram-se relevantes para aprimorar suas práticas.

Sendo assim, este trabalho apresenta-se como uma contribuição significativa para o aperfeiçoamento do processo de qualidade, resultando em entregas mais confiáveis, redução de retrabalho e melhor aproveitamento do esforço da equipe. Espera-se que as boas práticas propostas possam ser estendidas a outros projetos, fortalecendo a cultura de qualidade em toda a organização.

1.2 Objetivos

O objetivo geral deste trabalho é analisar o processo de garantia da qualidade de uma equipe ágil de desenvolvimento backend e propor um plano de ação para aprimorar as entregas da equipe.

Como objetivos específicos, pretende-se:

- Descrever o processo atual de garantia da qualidade utilizado pela equipe ágil;
- Identificar os pontos fortes, fracos e as oportunidades de melhoria no processo atual;
- Detalhar as melhores práticas da garantia da qualidade no contexto do desenvolvimento ágil;
- Elaborar um plano de ação detalhado para a implementação de melhorias no processo de garantia da qualidade, incluindo a adoção de novas ações e responsabilidades.

1.3 Método

Este trabalho adotará a metodologia de estudo de caso para analisar e propor melhorias na gestão da qualidade em uma empresa que ainda adota integralmente processos estruturados nessa área.

Primeiramente, será realizada a apresentação do contexto da empresa e serão coletadas as informações de um dos seus projetos, bem como das equipes envolvidas, com foco nos processos existentes, na maturidade do time em relação à qualidade e nos seus desafios enfrentados ao longo do seu ciclo de desenvolvimento de software. Esse levantamento será realizado por meio de entrevistas.

Com base nessas informações, será realizado um estudo sobre os principais problemas e impactos no processo de desenvolvimento. Serão identificados processos a serem aprimorados, alinhados a boas práticas e metodologias recomendadas. Então, a proposta de implantação será detalhada, definindo objetivos, responsabilidades, ferramentas e indicadores de sucesso.

1.4 Organização do Trabalho

O trabalho é organizado da seguinte maneira:

Capítulo 1 - Introdução: apresenta e contextualiza o tema, incluindo a justificativa, os objetivos, a metodologia adotada no desenvolvimento do trabalho e os trabalhos relacionados.

Capítulo 2 - Revisão Bibliográfica: aborda os principais conceitos relacionados à agilidade e à garantia da qualidade em projetos ágeis, com destaque para o papel do analista de qualidade e os testes funcionais.

Capítulo 3 - Estudo de Caso: Análise do Processo de Garantia da Qualidade na Empresa Omega: descreve o processo atual de garantia da qualidade adotado pela equipe analisada, destacando práticas, desafios e o papel do analista de qualidade.

Capítulo 4 - Proposta de Melhorias para o Processo de Garantia da Qualidade na Empresa Omega: apresenta uma proposta de implementação com ações práticas de qualidade, elaborada com base nos problemas identificados no capítulo anterior.

Capítulo 5 - Conclusão: apresenta as considerações finais, reforçando o papel da garantia da qualidade no desenvolvimento ágil e apresentando sugestões para trabalhos futuros.

2 Revisão Bibliográfica

2.1 Processo de desenvolvimento de software

Dada a importância dos softwares no mundo moderno, devido à sua relevância em diversas áreas e à dependência dos indivíduos e das sociedades deles, é fundamental considerar a preocupação em desenvolver sistemas de qualidade, confiáveis, econômicos e rápidos. Um sistema de software profissional é constituído por uma série de programas separados, exigindo documentação, recursos e informações adicionais para seu uso. Portanto, não se pode desenvolver um software focando apenas na implementação e no código final, sendo necessário adotar métodos, práticas e um conjunto de atividades que envolvam especificação, projeto, implementação, validação e evolução do produto. Essas etapas são complexas e abrangem diversas ações até a entrega final (SOMMERVILLE, 2011).

Cada software é único, seja por seu funcionamento ou pela forma como foi desenvolvido. E, assim como os softwares, os processos para sua idealização também são únicos, levando em consideração as atividades, ferramentas, discussões e imprevistos pelos quais passaram e foram necessários para a sua construção. Cada projeto apresenta particularidades que o tornam singular. Portanto, não existe um modelo ideal, sendo necessário adaptar a abordagem conforme as características do sistema.

Para garantir o sucesso do processo, é essencial a comunicação e a colaboração entre a equipe e as partes interessadas do projeto (HALVORSEN, 2020). Muitas vezes, existem visões distintas sobre a entrega entre as partes envolvidas, o que torna a comunicação fluida e contínua fundamental para o êxito no desenvolvimento de software, juntamente com outras habilidades interpessoais e técnicas.

Os processos utilizados para criar software nas empresas normalmente se baseiam em um ou mais modelos existentes, como os dirigidos a planos e os métodos ágeis, com o objetivo de manter a organização e alcançar os resultados esperados. Os modelos dirigidos a planos são tradicionais, focando no planejamento prévio de todas as atividades e na avaliação do progresso com base nesse planejamento. Já os métodos ágeis são mais dinâmicos, permitindo um planejamento gradual que pode ser alterado sempre que necessário para atender às mudanças desejadas pelos clientes (SOMMERVILLE, 2011).

Independentemente do modelo adotado, seja tradicional ou ágil, as abordagens aplicadas ao projeto desempenham um papel fundamental na garantia da qualidade. A adoção de práticas baseadas em processos de software bem definidos reduz a probabilidade de falhas e a necessidade de retrabalho (ALESSIO; SABADIN; ZANCHETT, 2017).

2.2 Métodos Ágeis

Com o avanço da tecnologia, o desenvolvimento ágil de aplicações tem ganhado cada vez mais espaço. Utilizando metodologias ágeis, um conjunto de práticas e técnicas aplicadas para aumentar a eficiência e produtividade dos projetos, que entregam softwares com mais agilidade e simplificam os processos (LIMA, 2022). Com isso, surgiram diversos métodos ágeis, como o Kanban, Scrum e o eXtreme Programming (XP) (KOSCHEVIC, 2011). Esses métodos têm em comum a aplicação de princípios ágeis, definidos no Manifesto para Desenvolvimento Ágil, publicado em 2001. Esse manifesto é responsável por embasar os métodos ágeis, que buscam valorizar mais:

1. Indivíduos e interações do que processos e ferramentas;
2. Software em funcionamento do que documentação abrangente;
3. Colaboração do cliente do que negociação de contrato;
4. Resposta às mudanças do que seguir um plano.

Os métodos ágeis visam ciclos curtos de desenvolvimento com entregas incrementais, onde cada incremento inclui uma nova funcionalidade no sistema. Além disso, os métodos facilitam que sejam feitas alterações conforme a necessidade do cliente, visto que os requisitos iniciais muito provavelmente serão alterados (SOMMERVILLE, 2011).

A pesquisa publicada no 16th State of Agile Report (AGILE, 2022), uma análise anual sobre técnicas e práticas ágeis conduzida por meio de eventos presenciais na Europa e nos Estados Unidos, além de consultas online, revela que 4 em cada 5 respondentes trabalham em organizações com times ágeis, sendo que metade está adotando uma combinação entre ágil e outros métodos. De acordo com o estudo, aproximadamente 87% dos respondentes afirmam utilizar o Scrum, enquanto 56% utilizam o Kanban, 27% um híbrido entre Scrum e Kanban, 13% um híbrido entre Scrum e XP e 7% o XP. Mais de um terço dos respondentes atuam como Scrum Master e um quarto na área de tecnologia, evidenciando a ampla adoção dos métodos ágeis nas empresas, tornando-se parte essencial do processo de desenvolvimento.

2.2.1 Scrum

De acordo com o Guia do Scrum (SCHWABER; SUTHERLAND, 2013), desenvolvido pelos criadores do framework ágil, o Scrum foi projetado para o desenvolvimento de produtos complexos nos quais diversos processos e técnicas podem ser agregados, e nos quais os problemas são resolvidos durante o processo de entrega. Portanto, o Scrum consiste em um conjunto de práticas incrementais e iterativas que visam assegurar a entrega de produtos com alto valor agregado para o cliente.

A Figura 1 apresenta uma visão geral do ciclo de vida de um projeto no framework

Scrum. A seguir, será apresentado um detalhamento de cada etapa do processo, com informações mais específicas sobre cada fase.

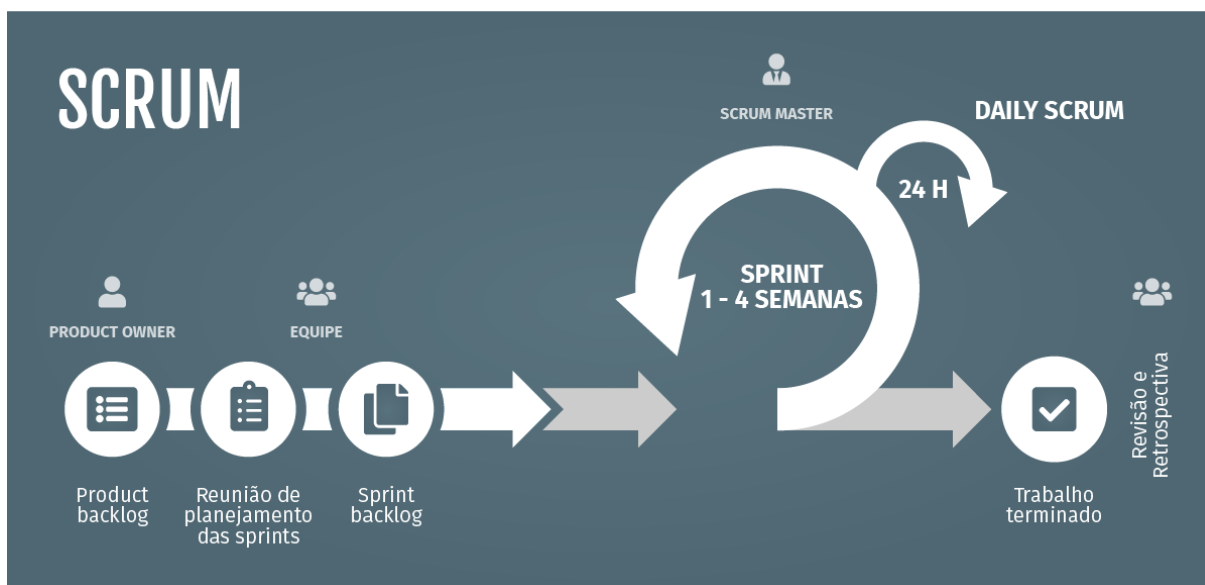


Figura 1 – Framework Scrum

Fonte: <https://www.dio.me/articles/scrum-o-que-e-quais-seus-pilares-principios-e-eventos>.

2.2.1.1 Time Scrum

Os Times Scrum são responsáveis pelas entregas dos produtos e são compostos pelo Product Owner, Scrum Master e Time de Desenvolvimento. A intenção é que os times sejam auto-organizáveis e multifuncionais, organizando-se para entregar seu trabalho da melhor forma possível, possuindo todas as competências necessárias para isso.

O Product Owner (PO), ou dono do produto, é o profissional que possui um alto nível de conhecimento sobre o produto, responsável por representar os interesses do cliente e das demais partes interessadas no projeto. Ele define e gerencia o backlog, prioriza as tarefas e garante que o trabalho seja realizado conforme o esperado (SCHWABER; SUTHERLAND, 2013). Ao contrário do cliente, o Product Owner é um membro ativo do time, participando das cerimônias e atuando na resolução de impedimentos e dúvidas de negócios durante as sprints.

O Scrum Master (SM) é o especialista em Scrum dentro do time, responsável por garantir a compreensão e aplicação do framework, facilitando e apoiando a realização das práticas (SCHWABER; SUTHERLAND, 2013). Ele é quem acompanha as métricas do time, como Velocity e Lead Time, auxiliando o time de desenvolvimento na remoção de quaisquer impedimentos que possam surgir antes ou durante a execução de suas tarefas, para garantir que os objetivos da sprint sejam alcançados.

Os desenvolvedores são responsáveis por implementar novos incrementos para o produto a cada sprint. São profissionais multidisciplinares que possuem o conhecimento técnico necessário para criar o incremento do produto, com autonomia para se organizar, definir suas tarefas com base no backlog e estabelecer a capacidade de trabalho a ser realizada.

2.2.1.2 Eventos Scrum

O Scrum possui uma rotina de eventos com objetivos e cargas horárias específicos. Esses eventos se repetem a cada sprint, sendo esta responsável por abranger todos eles.

A sprint representa um período de tempo, normalmente de 2 a 4 semanas, que se repete ao longo do projeto, no qual são desenvolvidos os incrementos do produto. Durante esse ciclo, ocorrem os demais eventos do Scrum: as reuniões diárias, o planejamento, a revisão e a retrospectiva (SCHWABER; SUTHERLAND, 2013).

Na reunião de planejamento, realizada no primeiro dia do ciclo, define-se o que será desenvolvido nas próximas semanas. O Product Owner é responsável por estabelecer o objetivo e apresentar os itens do backlog necessários para alcançá-lo. Os desenvolvedores devem discutir as propostas, definir o que pode ser atendido e quais atividades serão executadas. O Scrum Master conduz o encontro, garantindo que ele ocorra conforme o previsto e dentro do tempo estipulado. Ao final, é definida em conjunto a meta a ser alcançada até o encerramento do período.

As reuniões diárias são eventos curtos de 15 minutos, no mesmo local e horário, onde o time se atualiza sobre o progresso das atividades, informando o que foi feito no dia anterior, o que será feito hoje e se existe algum impedimento que possa comprometer o andamento do trabalho.

A revisão da sprint é realizada ao final de cada ciclo, onde é apresentado o incremento desenvolvido e são discutidos os acontecimentos durante a sprint, positivos e negativos. Avalia-se o backlog do produto, considerando seus itens prontos e pendentes, o que fornece uma perspectiva para a próxima sprint. Todo o Time Scrum participa da reunião, além da presença de alguma outra parte interessada.

A retrospectiva da sprint acontece após a reunião de revisão, marcando o encerramento da sprint. Nesse momento, o Time Scrum reflete sobre o ciclo anterior, identifica pontos de melhoria e elabora um plano de ação para a próxima sprint (SCHWABER; SUTHERLAND, 2013).

2.2.1.3 Artefatos do Scrum

O backlog do produto é uma lista que compreende todas as funcionalidades que devem ser implementadas para a entrega do produto no final do projeto. O Product Owner

é o responsável por gerir o backlog e cuidar de sua evolução durante as sprints.

O backlog da sprint é uma lista de atividades que o time de desenvolvimento delibera durante a reunião de planejamento e que precisam ser completadas durante a sprint para que o incremento do produto seja entregue ao final dela.

O incremento representa o conjunto de funcionalidades do Product Backlog consideradas prontas que foram entregues ao final da sprint atual, somado a todas as funcionalidades entregues nas sprints anteriores. Para ser considerado "pronto", o incremento deve atender a um conjunto de critérios que definem quando uma funcionalidade está completa. Fica a cargo do time de desenvolvimento definir os critérios de "pronto" do produto quando estes ainda não foram definidos.

2.2.2 Kanban

O Kanban é uma metodologia de origem japonesa e amplamente utilizada em empresas de produção que buscam produzir o que é necessário, com qualidade e no tempo certo. Seu objetivo é a redução de desperdícios em todo o ciclo de desenvolvimento, tendo como principais características a visualização do trabalho, a limitação do trabalho em cada etapa, o foco no fluxo e a melhoria contínua (RAUT et al., 2015).

A visualização do trabalho se dá por meio de um quadro visual dividido em colunas, onde cada uma representa diferentes etapas do desenvolvimento. A quantidade de colunas e suas nomenclaturas podem variar para se ajustar às necessidades de cada projeto e representar as etapas do fluxo de trabalho. Cada atividade a ser realizada é representada por um cartão que percorre as colunas à medida que sua execução avança. Dessa forma, é possível acompanhar visualmente o andamento do desenvolvimento do produto.

A limitação de trabalho em cada etapa visa evitar o acúmulo de atividades, restringindo o número de tarefas por coluna no fluxo, o que garante um desenvolvimento mais fluido e previne desperdícios e sobrecargas.

O foco no fluxo e na melhoria contínua decorre do fato de a metodologia promover uma entrega contínua e regular de incrementos, além da realização periódica de revisões no trabalho em busca de melhorias (RAUT et al., 2015).

A Figura 2 demonstra um exemplo de quadro Kanban, no qual cada coluna representa uma etapa e cada cartão corresponde a uma atividade. Essa abordagem possibilita à equipe visualizar, de forma centralizada, as tarefas pendentes, as que estão em andamento e as concluídas.

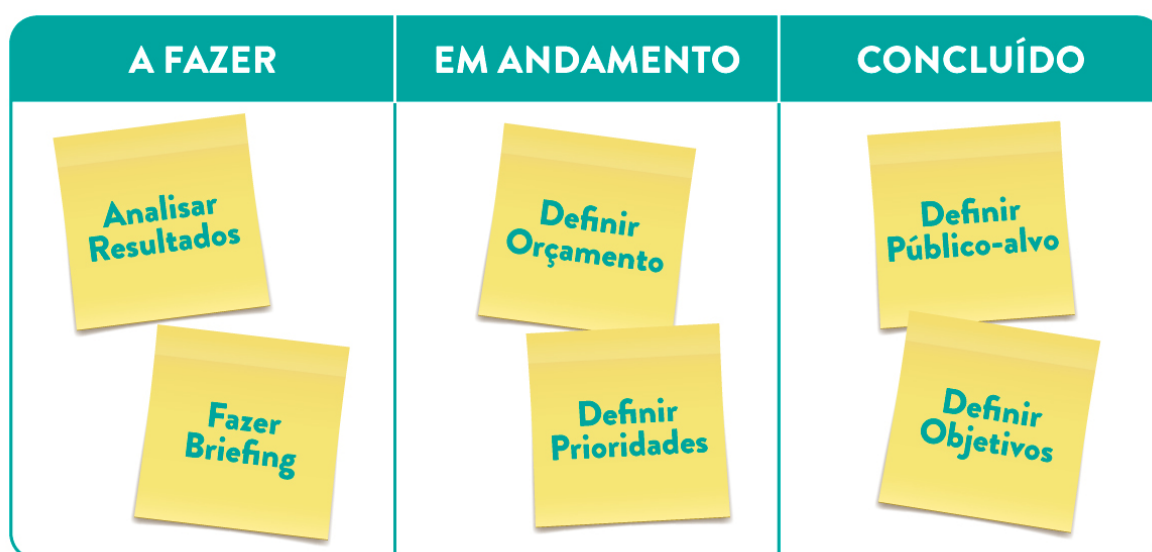


Figura 2 – Metodologia Kanban

Fonte: <https://acqio.com.br/blog/metodologia-kanban-eficiencia-e-productividade/>

2.2.3 Extreme Programming

O Extreme Programming (XP) é um framework ágil projetado para equipes de pequeno a médio porte, com o objetivo principal de acelerar o desenvolvimento de software enquanto atende aos requisitos e expectativas do cliente. Baseado em um conjunto de princípios e valores que orientam sua aplicação, incorpora as melhores práticas no desenvolvimento de software (SCHNOELLER; SANTOS, 2013).

Os valores do XP incluem a comunicação eficaz, simplicidade, coragem, feedback rápido e o respeito. Seus princípios fundamentais abrangem conceitos como humanismo, economia, benefício mútuo, diversidade, auto-semelhança, aceitação da falha, fluidez, melhoria contínua, oportunidades, qualidade, reflexão, redundância, responsabilidade assumida e a prática de dar "passos de bebê" no desenvolvimento (BECK; ANDRES, 2012).

O processo do Extreme Programming é iterativo e incremental, onde há uma divisão do projeto em entregas menores, chamada release, em que cada uma delas contribui com um incremento de novas funcionalidades à entrega final (DUDZIAK, 2000). Além das releases, o XP inclui outras práticas principais, como testes constantes, refatoração, programação em pares e integração contínua.

Juntamente à adoção de métodos ágeis, é comum o uso de ferramentas que apoiem o gerenciamento e a organização das entregas nos projetos. O Jira é umas das soluções utilizadas nesse contexto, oferecendo quadros baseados em Scrum e Kanban para que os times visualizem seu fluxo de trabalho, além de outras funcionalidades como linhas do tempo, relatórios e a integração com outras ferramentas da própria plataforma, como o

Zephyr, utilizado para a gestão de testes de software ([ATLASSIAN, b](#)).

2.3 Garantia da qualidade

Devido à crescente integração dos softwares na vida das pessoas e sua importância na rotina diária, somada à intensa competitividade do mercado, a necessidade de entregar sistemas de alta qualidade e que agreguem valor ao usuário se torna cada vez mais importante ([ISTQB, 2023](#)).

Consistindo em um conjunto de ações planejadas para garantir o cumprimento dos requisitos de qualidade, a garantia da qualidade busca fornecer confiança no produto e assegurar que ele está apto para uso ([RUSSELL, 2011](#)). Focada na prevenção de falhas, busca assegurar a qualidade durante o desenvolvimento do produto e em sua entrega final. Trata-se de um processo abrangente que envolve todo o ciclo de desenvolvimento e a participação de todo o time.

O analista de qualidade, também conhecido como analista de testes, tem a missão, junto com todo o time, de garantir a construção de um produto de qualidade que gere valor para o negócio. Suas responsabilidades não se limitam à execução de testes, abrangendo o acompanhamento de todo o desenvolvimento e a identificação precoce de problemas ([INVILLIA, 2025](#)).

2.3.1 O papel do analista de qualidade tradicional

No modelo tradicional de desenvolvimento de software, como o Waterfall, o trabalho ocorre de forma sequencial. Nesse cenário, o analista de qualidade geralmente atua de maneira independente da equipe de desenvolvimento, aguardando a finalização da implementação para iniciar os testes e validar o produto antes da entrega ([INVILLIA, 2025](#)). Como resultado, a verificação dos requisitos e a validação de que o produto atende às expectativas ficam adiadas, o que pode acarretar consequências negativas, como aumento de custos, surgimento de erros inesperados e atrasos na entrega ([MAGOWAN, 2020](#)).

As atividades do analista vão além da execução de testes. Ele também é responsável por planejar, detalhar e documentar os casos de teste, além de gerar relatórios sobre a execução e os problemas identificados.

Os testes funcionais são amplamente utilizados pelos analistas de qualidade, tanto em contextos tradicionais ([MOUSAEI, 2020](#)) quanto ágeis, diferenciando-se apenas no momento de sua aplicação. Esse tipo de teste é baseado nas especificações dos requisitos, sem a necessidade de análise do código-fonte. Ele se concentra em validar se o sistema se comporta conforme esperado, utilizando um conjunto de entradas e saídas previsíveis ([NIDHRA, 2012](#)).

2.3.2 O papel do analista de qualidade ágil

Em projetos ágeis, os testes começam já nos estágios iniciais do desenvolvimento e são realizados em diferentes níveis, do mais alto ao mais baixo. Além de verificar se o que foi desenvolvido está em conformidade com os requisitos, validar se o sistema atende às necessidades do cliente e identificar defeitos, o analista de qualidade no contexto ágil deve ter uma visão ampla da qualidade do produto, garantindo a entrega de soluções de alta qualidade ([AZZAM, 2022](#)).

Ao contrário do modelo tradicional, no contexto ágil o analista não se concentra apenas em encontrar defeitos, mas principalmente em preveni-los, assegurando a qualidade desde o início do desenvolvimento ([INVILLIA, 2025](#)). Os testes, nesse contexto, são essenciais não apenas para garantir a qualidade, mas também para orientar os esforços do time de desenvolvimento, ajudando a direcioná-los de forma mais eficaz e antecipada ([NIKOLOVA, 2019](#)).

Em equipes ágeis, a qualidade não é mais uma responsabilidade exclusiva de um único papel. Ela resulta de um esforço coletivo, em que todos os membros do time se envolvem nos diferentes aspectos da qualidade e nas atividades que contribuem para os testes ([NIKOLOVA, 2019](#)).

O profissional de qualidade ágil trabalha de forma integrada ao ciclo de desenvolvimento e à equipe, como no Scrum, onde participa de todas as cerimônias e contribui ativamente para nas atividades em cada sprint. Ele fornece feedback rápido e contínuo para o time, analisa métricas e busca constantemente aprimorar as práticas de teste e a qualidade do produto.

2.3.3 Revisão de Código (Code Review)

A garantia da qualidade engloba diversas ações, especialmente em contextos ágeis, nos quais há um foco na prevenção de defeitos desde os estágios iniciais do desenvolvimento. A revisão de código, mais conhecida pelo termo em inglês code review, é uma prática essencial nesse contexto, pois permite detectar problemas no código antes que ele seja integrado ao repositório principal.

Trata-se de uma prática no qual o código é revisado, normalmente entre pares, ou seja, desenvolvedores analisam o código uns dos outros com o objetivo de identificar erros lógicos, bugs e possíveis problemas de desempenho. Essa análise baseia-se na experiência técnica dos desenvolvedores, no conhecimento específico sobre o projeto e nas boas práticas e padrões de código adotados pela equipe ([SILVA et al., 2023](#)).

Além de contribuir para a redução de falhas e para o aumento da confiança no software, a revisão de código promove o compartilhamento de conhecimento entre os membros da equipe, fortalecendo o trabalho colaborativo ([SANTOS; NUNES, 2018](#)).

2.3.4 Métricas e Relatórios

O desenvolvimento ágil traz consigo a rapidez na entrega de novas funcionalidades, mantendo a qualidade dessas entregas. Essa abordagem exige que as equipes de desenvolvimento e garantia da qualidade estejam constantemente empenhadas em aprimorar seus processos internos. Nesse contexto dinâmico, as métricas de software surgem como ferramentas fundamentais, permitindo a geração de relatórios que facilitam a análise e visualização das tendências. Esses relatórios servem como base para a definição das próximas ações, garantindo um ciclo constante de melhorias.

As métricas desempenham um papel essencial, pois capturam informações relevantes sobre o software, como a qualidade do código, e também podem ser aplicadas à gestão do projeto, auxiliando no acompanhamento da evolução do trabalho (LEBLANC *et al.*, 2015). Elas promovem a transparência dos processos e fornecem uma base sólida para a tomada de decisões mais assertivas, além de possibilitarem um acompanhamento mais eficaz da qualidade.

Dentre as métricas mais relevantes para o cotidiano das equipes de desenvolvimento, destacam-se: a cobertura de teste de unidade, que contribui para a manutenção da qualidade do código; a velocidade da equipe, que mede a capacidade de entrega durante as sprints em um contexto Scrum; e a densidade de defeitos, que reflete a quantidade de defeitos identificados ao longo do trabalho (CESAR; CESAR; MONTEIRO, 2018). Outras métricas, como o tempo de correção de erros e a complexidade das entregas, também são importantes para a qualidade e gestão do desenvolvimento.

2.3.5 Gestão de Documentos

Como discutido por (SOMMERVILLE, 2011), embora as práticas ágeis enfatizem o foco em um código bem estruturado e em sua melhoria constante, em vez da documentação formal extensa, uma boa documentação de requisitos do sistema continua sendo essencial para que a equipe de desenvolvimento compreenda claramente o que deve ser entregue. A documentação é fundamental também para a manutenção dos softwares, pois registra informações relevantes sobre seu desenvolvimento e decisões técnicas tomadas ao longo do projeto.

Outro aspecto importante diz respeito à rotatividade de membros nas equipes ágeis. Caso ocorram alterações na composição da equipe, há o risco de perda de conhecimento se ele não estiver documentado ou compartilhado entre os demais integrantes (SOMMERVILLE, 2011). Assim, a documentação se mostra indispensável para o compartilhamento de conhecimento, a continuidade do projeto e a execução de manutenções futuras.

A documentação de projeto em contextos ágeis tende a ser mais concisa e objetiva,

podendo abranger aspectos como os objetivos do software, funcionalidades essenciais, fluxos e organização de dados, estrutura lógica, limitações, restrições, dependências, entre outras informações que a equipe considerar relevantes para o sucesso da entrega ([PRES-SMAN, 2016](#)). Além da sua produção, é fundamental que essa documentação esteja centralizada em um repositório acessível a todos os membros da equipe, garantindo seu fácil acesso e atualização durante o desenvolvimento.

O Confluence é uma ferramenta que pode ser utilizada para o gerenciamento de documentações em projetos de software. A plataforma permite a criação de anotações, o planejamento de projetos, a utilização de quadros brancos e a organização de espaços personalizados para os times. Por meio desses recursos, o Confluence viabiliza a colaboração entre os membros da equipe na criação, organização e compartilhamento das informações do projeto de forma estruturada, contribuindo para o alinhamento entre as partes interessadas e para a preservação do conhecimento ao longo do tempo. ([ATLASSIAN, a](#)).

2.4 Testes de Software

Os testes de software são um conjunto de atividades essenciais para avaliar a qualidade do software e reduzir os riscos de falhas durante sua operação. Eles desempenham um papel crucial na garantia da qualidade e no desenvolvimento, ajudando a identificar defeitos e garantir a qualidade do software.

A verificação e a validação são pilares fundamentais dos testes de software. A verificação consiste em assegurar que o software atenda aos requisitos definidos, enquanto a validação verifica se o software satisfaz as necessidades dos usuários e das demais partes interessadas ([ISTQB, 2023](#)).

Os testes de software podem ser classificados como dinâmicos ou estáticos. Os testes dinâmicos envolvem a execução do software utilizando diferentes tipos e técnicas de testes, enquanto os testes estáticos envolvem revisões de código e a análise estática para identificar possíveis defeitos.

O processo de testes é composto por um conjunto de atividades que incluem:

- O planejamento do teste, onde são definidos os objetivos dos testes e as estratégias para alcançá-los.
- O monitoramento de teste, que foca na contínua verificação das atividades de teste e o acompanhamento do que foi definido com o progresso atual.
- O controle do teste, envolvendo as ações necessárias para assegurar os objetivos do teste.

- A análise de teste que inclui a identificação dos itens testáveis, a avaliação dos riscos associados e a priorização dos testes.
- A modelagem e implementação do teste com a definição dos requisitos de dados, elaboração dos casos de testes, estruturação do ambiente e demais ferramentas necessárias para sua execução.
- A execução do teste, a documentação e a análise dos resultados. Caso haja intercorrências, elas são analisadas para identificar suas causas e serem repassadas para os responsáveis.

2.4.1 Testes Funcionais

Entre os tipos de testes de software, destacam-se os funcionais, que têm como objetivo avaliar se o software está funcionando de acordo com os requisitos especificados. Esses testes se concentram nas funcionalidades do sistema, testando o comportamento do software a partir da perspectiva do usuário final ou dos requisitos técnicos ([ISTQB, 2023](#)).

Diferentes níveis de validações podem ser realizados, manualmente ou de forma automatizada, dependendo da necessidade do projeto. Os testes funcionais são essenciais para garantir que o software entregue esteja livre de falhas que possam comprometer sua funcionalidade e experiência do usuário.

Alguns níveis de testes são descritos a seguir:

- **Teste unitário:** Tem como objetivo verificar módulos individuais de forma isolada. Ele avalia a menor unidade testável do software, buscando identificar falhas decorrentes de erros de lógica ou implementação, com base nas especificações definidas.
- **Teste de integração:** Tem como foco validar a interação entre duas ou mais unidades que precisam funcionar em conjunto. Nesse nível, são verificados os fluxos de dados e a comunicação entre os componentes integrados do sistema.
- **Teste de sistema:** Realiza a verificação do comportamento geral do sistema já completamente integrado. Esse teste avalia se todas as funcionalidades, recursos e processos operam conforme o esperado.
- **Teste de aceitação:** Tem como objetivo garantir que o sistema atenda aos requisitos de negócios definidos pelo cliente ou pelas partes interessadas, sendo a validação final antes da entrega ([UMAR, 2020](#)).
- **Teste End-to-End (E2E):** É semelhante ao de integração, porém adota a perspectiva do usuário final e considera fluxos completos. Esses testes são realizados do início ao fim, validando a interação entre todos os componentes do sistema em um cenário

realista. Normalmente, são executados após a aprovação dos testes de unidade e de integração (TSAI et al., 2001).

O teste de regressão, por sua vez, não é considerado um nível, mas sim uma estratégia de teste. Ele visa garantir que alterações realizadas no sistema, como correções de defeitos ou inclusão de novas funcionalidades, não tenham causado impactos negativos em funcionalidades que já estavam funcionando corretamente (NETO, 2007).

Diferentes níveis dos testes funcionais podem ser aplicados seguindo o modelo da Pirâmide de Testes, que apoia o time na automação, na alocação de esforço e na quantidade de testes esperados em cada camada (VOCKE, 2018). Na base da pirâmide estão os testes de unidade, menores e mais estáveis, que verificam o comportamento de componentes individuais do sistema (LIMA; LEAL; MESQUITA, 2023). Em seguida, encontram-se os testes de integração, que validam a interação entre os diferentes componentes do sistema. No topo da pirâmide estão os testes End-to-End (E2E). Esses testes avaliam o sistema de ponta a ponta, verificando as integrações e funcionalidades em conjunto (LIMA; LEAL; MESQUITA, 2023).

A Figura 3 ilustra a Pirâmide de Testes, com os testes de unidade posicionados na base, os de integração no meio e os End-to-End (E2E) no topo. Nas laterais da pirâmide, é possível observar a relação entre tempo e custos associados a cada tipo de teste, dependendo da etapa representada na estrutura.

Embora possam ser executados manualmente, a automação dos testes funcionais tem-se tornado fundamental para garantir eficiência e consistência na validação do software.

Os testes de API são um tipo de teste que pode ser enquadrado tanto como testes de integração quanto como testes End-to-End (E2E). Eles podem verificar se as interfaces estão conectadas e funcionando corretamente, além de testar o comportamento de ponta a ponta do sistema, passando por todas as camadas. As APIs funcionam como pontes entre diferentes componentes do software, permitindo a comunicação e a troca de informações entre eles. Devido à sua relevância, os testes de API podem se tornar exaustivos, sendo a automação um a grande aliada nesse cenário (MUDASSAR; KHAN, 2024). Ferramentas como o Postman podem ser utilizadas para a execução manual desses testes. Já a automação pode ser realizada com ferramentas como o RestAssured, que permite validar requisições e respostas HTTP, garantindo a funcionalidade e o desempenho das interfaces de uma aplicação (ASSURED, 2024).



Figura 3 – Pirâmide de Testes

Fonte: <https://www.alura.com.br/artigos/testes-automatizados>.

2.5 Testes Automatizados

Os testes automatizados referem-se à execução automática de testes utilizando ferramentas específicas para validar funcionalidades e outros aspectos do software. No contexto ágil e DevOps, essa prática tem se tornado essencial, pois permite a execução contínua de testes com rapidez e precisão, além de contribuir para a redução de custos, economia de tempo e melhoria da qualidade das entregas (GAROUSI; MäNTYLÄ, 2016).

Além de otimizar tempo e recursos, a automação proporciona um ciclo de feedback mais rápido, permitindo que os desenvolvedores e equipes de qualidade foquem na validação de novas funcionalidades e na melhoria contínua do sistema. Os testes automatizados são especialmente valiosos em softwares que passam por múltiplas versões, sendo fundamentais para a execução de testes de regressão (GAROUSI; MäNTYLÄ, 2016). Essa abordagem possibilita a validação de fluxos principais, críticos e altamente repetitivos sem a necessidade de reexecutar os testes manualmente a cada nova versão.

Em relação à Pirâmide de Testes, os testes localizados na base da pirâmide são mais aconselháveis para automação devido ao seu custo mais baixo, maior rapidez e menor esforço. À medida que se sobe na pirâmide, os testes tornam-se mais caros, complexos e exigem mais esforço para serem realizados (VOCKE, 2018).

Diversas ferramentas, como Selenium, JUnit e Cypress.io, são amplamente utili-

zadas para a automação de testes funcionais. Elas garantem que as funcionalidades do software sejam validadas continuamente, evitando que mudanças introduzam novos defeitos e comprometam seu funcionamento ([APIDOG, 2024](#)).

2.5.1 Testes Não Funcionais

Os testes não funcionais avaliam os aspectos não funcionais do sistema, ou seja, validam o comportamento do sistema ao invés das suas funcionalidades. Para isso, são verificados aspectos como desempenho, usabilidade, confiabilidade e segurança, sendo uma parte importante da garantia da qualidade ([ISTQB, 2023](#)).

2.6 DevOps

Com o aumento da concorrência no mercado de software, as empresas têm investido cada vez mais em estratégias que permitam o desenvolvimento e a entrega contínua de produtos de alta qualidade em um ritmo acelerado. Nesse contexto, surge o DevOps, um conjunto de práticas, cultura e ferramentas que visa integrar desenvolvimento e operações, promovendo automação, colaboração e um fluxo constante de entregas de funcionalidades. Além disso, a qualidade do produto é um foco central, com ênfase em práticas que reduzem riscos, como feedback rápido, testes contínuos e entregas incrementais, evitando a acumulação de falhas e facilitando a detecção precoce de problemas ([SHAHIN; BABAR; ZHU, 2017](#)).

Quando a mentalidade DevOps é adotada, a responsabilidade pela qualidade deixa de ser exclusiva da equipe de qualidade, tornando-se uma preocupação compartilhada por toda a equipe. A qualidade passa a ser monitorada de forma recorrente ao longo de todo o ciclo de vida do software. Nesse modelo, o papel dos analistas de qualidade se expande, tornando-se mais colaborativo e integrado ao fluxo de trabalho das equipes de desenvolvimento e operações. Práticas como a Integração Contínua (CI) e a Entrega Contínua (CD) ajudam a garantir que a qualidade seja aprimorada desde as fases iniciais até a produção, não sendo algo verificado apenas no final do ciclo ([SATO, 2014](#)).

Para garantir entregas ágeis e de qualidade, o DevOps foca na automação de processos essenciais ao ciclo de desenvolvimento e operação do software, incluindo a compilação do código, a execução de testes automatizados, o empacotamento, a criação e configuração de ambientes, a gestão da infraestrutura, a migração de dados, além do monitoramento e análise de desempenho ([SATO, 2014](#)).

2.6.1 Integração Contínua (Continuous Integration - CI)

A Integração Contínua, ou Continuous Integration (CI), é um dos pilares técnicos do DevOps, permitindo a incorporação frequente do código desenvolvido ao repositório principal, idealmente várias vezes ao dia. Essa prática permite identificar e resolver conflitos rapidamente, além de assegurar a qualidade do software desde as fases iniciais (SHAHIN; BABAR; ZHU, 2017).

A adoção da CI facilita ciclos de entrega mais curtos e frequentes, aumentando a produtividade das equipes e aprimorando a qualidade do software. A automação de testes, como os de unidade, integração e regressão, desempenha papel essencial na identificação e correção de defeitos ainda nas primeiras etapas do desenvolvimento (SATO, 2014). Ferramentas como Jenkins, GitHub Actions e GitLab CI são amplamente utilizadas para automatizar a integração e a execução de testes a cada alteração, garantindo que o código esteja sempre pronto para ser integrado.

Dentro da Integração Contínua, a prática de Shift-Left Testing, ou deslocamento dos testes para a esquerda, é essencial para identificar e prevenir defeitos o mais cedo possível. Os testes são realizados paralelamente ao desenvolvimento, melhorando a qualidade do produto desde as primeiras fases (KINSBRUNER, 2020).

Exemplos práticos de Shift-Left Testing incluem testes de requisitos, realizados nas fases iniciais para garantir que os requisitos sejam claros e implementáveis, e testes de unidade, que são desenvolvidos em paralelo à codificação, facilitando a correção precoce de erros.

2.6.2 Entrega Contínua (Continuous Delivery - CD)

O conceito de Continuous Delivery (CD) abrange tanto a Entrega Contínua quanto a Implementação Contínua. A Entrega Contínua refere-se à capacidade de disponibilizar uma nova versão de software de forma rápida e confiável para os clientes, mediante aprovação manual, priorizando a automação sempre que possível. Já a Implementação Contínua ou Continuous Deployment é a abordagem onde a implantação é completamente automatizada, sem necessidade de aprovação manual, liberando as versões de software automaticamente para produção (SHAHIN; BABAR; ZHU, 2017).

Entre os principais benefícios da Entrega Contínua, destacam-se o fortalecimento da colaboração entre as equipes de desenvolvimento e operações, a redução de tarefas manuais e a melhoria da eficiência do fluxo (SATO, 2014). A manutenção de testes automatizados ao longo de todo o ciclo de CD garante que apenas versões estáveis sejam entregues, por meio de testes funcionais, de desempenho e de homologação.

Além disso, a análise de métricas de qualidade e o monitoramento pós-implantação são fundamentais para identificar e corrigir problemas em produção de maneira ágil, mini-

mizando impactos nos usuários. Ferramentas como Prometheus e Grafana são amplamente usadas para monitorar a confiabilidade e o desempenho das aplicações (SATO, 2014).

A Figura 4 ilustra o ciclo contínuo do DevOps, destacando as etapas de Code, Plan, Monitor, Operate, Deploy, Release, Test, Build, e sua interconexão dentro do processo de desenvolvimento. A figura proporciona uma visão clara da iteratividade do fluxo, com uma sequência de ações que resultam em um ciclo contínuo de entregas, sempre com foco na qualidade.

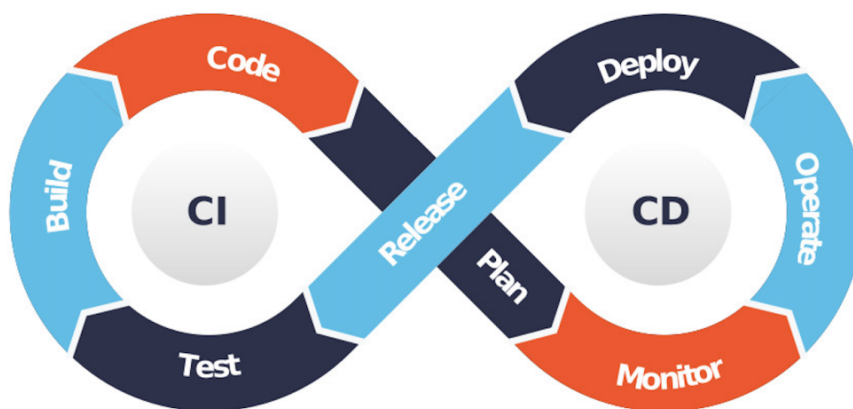


Figura 4 – Funcionamento do CI/CD em DevOps

Fonte: <https://blog.xpeducacao.com.br/ci-cd-e-devops/>

2.7 Trabalho Relacionados

O trabalho desenvolvido por (RIBEIRO, 2019) tem como objetivo apresentar de forma teórica e prática as atividades de testes de software, com foco na automação de testes e na cultura DevOps. O autor inicia abordando a relevância da área de Garantia da Qualidade, seu funcionamento e desafios. Ele também define testes automatizados e sua aplicação no cenário atual. Além disso, o conceito de DevOps é discutido, desde sua origem até seu funcionamento. Ao longo do projeto, o autor desenvolve uma aplicação web, define os testes necessários para garantir a qualidade desejada, automatiza esses testes e aplica a metodologia DevOps. O autor conclui que a automação de processos no desenvolvimento de software é altamente produtiva e sugere que outras tarefas do ciclo de desenvolvimento também poderiam ser automatizadas.

A dissertação "Qualidade de software no desenvolvimento com métodos ágeis", de Bruno Henrique de Oliveira (OLIVEIRA, 2014), oferece uma análise detalhada sobre a qualidade de software no contexto ágil, com ênfase nas métricas utilizadas nesse processo. O autor realizou um estudo de caso e uma enquete sobre as atividades de garantia e controle de qualidade, as métricas, os processos e as ferramentas utilizadas no desenvolvi-

mento de software. Como resultado, constatou-se que a execução constante de atividades, como a revisão de código, é fundamental para assegurar a qualidade em processos ágeis. O autor conclui que os métodos ágeis englobam diversas práticas focadas na garantia da qualidade das entregas, desde as fases iniciais do ciclo de desenvolvimento.

No artigo "Testing Strategies in an Agile Context" de Zornitsa Nikolova ([NIKOLOVA, 2019](#)), o teste no contexto ágil é destacado como extremamente importante para garantir a qualidade do produto e do processo. O artigo apresenta um quadrante com diferentes tipos de testes que podem ser aplicados, como de unidade, exploratórios, de interface, de aceitação, de performance e de segurança. Aborda o investimento necessário em automação de testes para garantir uma maior cobertura, feedback rápido e minimização de desperdícios de tempo. Discute tanto a pirâmide de testes tradicional, na qual os testes manuais ocupam uma maior parte, quanto a pirâmide de testes ágil, na qual os testes de unidade têm a maior fatia. A autora comenta sobre como a transição para testes ágeis pode ocorrer e destaca que, mesmo quando a equipe e a organização estão cientes e convencidas dos benefícios dos testes ágeis, o início pode ser uma tarefa difícil, pois muitas vezes as equipes ficam sem saber o que fazer e por onde começar. O autora conclui que o principal valor das abordagens ágeis está na ausência de uma separação rígida entre técnicas, papéis e responsabilidades nas equipes, destacando que o pensamento ágil envolve todo o desenvolvimento com foco no cliente. O artigo se relaciona com o trabalho em andamento, pois ambos abordam a qualidade e os testes no contexto ágil, o pensamento ágil e as dificuldades que a implementação dos processos ágeis pode enfrentar.

O artigo científico "Agile Software Development" de JinJin Li ([LI, 2017](#)) discute os métodos ágeis na criação de software. Inicialmente, o autor contextualiza as abordagens tradicionais, como o modelo cascata, discutindo seu funcionamento, vantagens e desvantagens. Em seguida, explora o modelo ágil, detalhando cada um dos 12 princípios por trás do Manifesto Ágil e a diferença entre os modelos tradicionais e ágeis. Destaca o método ágil Scrum, descrevendo seus processos, papéis, sprints, vantagens e limitações. O artigo também apresenta um exemplo prático real de aplicação de métodos ágeis. Conclui ressaltando a melhoria na qualidade dos produtos finais, a eficiência dos desenvolvedores e a redução de erros, proporcionadas por essa abordagem. O artigo se conecta diretamente com o trabalho em questão, abordando temas como os modelos tradicionais e ágeis, além de um exemplo prático de aplicação.

3 Análise do processo de Garantia da Qualidade na Empresa Omega

Este capítulo tem como objetivo apresentar as principais características da empresa de tecnologia, cujo processo de qualidade no ciclo de desenvolvimento de software será analisado. Para isso, serão explorados seu contexto e fluxo de trabalho na entrega de produtos e, posteriormente, será apresentada a proposta de implantação de melhorias. A empresa não terá o seu nome real revelado para preservar seu sigilo, sendo chamada de Empresa Omega neste trabalho.

3.1 Caracterização da empresa e do projeto

A Empresa Omega faz parte de um conglomerado de uma multinacional. Fundada em 2016, ela é uma plataforma global de comércio digital B2B, que permite aos varejistas realizar a compra de produtos para seus negócios, gerenciar a compra, acompanhar o status das transações e acessar dados relevantes sobre o desempenho de seus negócios.

Com sua plataforma, a Omega busca criar novas oportunidades de negócios para pequenos e médios varejistas, ajudando-os a tornar seus empreendimentos mais lucrativos. A empresa conta com equipes de desenvolvimento de software internas, além da terceirização de profissionais por meio de consultorias especializadas em tecnologia.

Este estudo de caso foca em um dos projetos da empresa: a criação de uma plataforma digital Web, voltada para uso interno e de parceiros, com o objetivo de facilitar o pagamento de compras realizadas pela plataforma da empresa. O desenvolvimento do projeto já ultrapassa um ano, mas ainda não há previsão de sua disponibilização para uso em produção. No entanto, várias entregas já foram realizadas ao longo desse período.

O detalhamento dos processos realizados pela empresa Omega descritos nesse estudo de caso se deu através de entrevistas de caráter exploratório realizadas com o analista de qualidade e desenvolvedores do projeto. Nessas conversas, foram detalhados aspectos como a dinâmica do time ao longo do ciclo de desenvolvimento, os principais desafios enfrentados e o impacto das práticas adotadas na entrega do produto com foco no processo atual de garantia da qualidade.

O projeto é dividido em duas equipes de desenvolvimento: uma de frontend e outra de backend. Este estudo se concentra nos processos adotados pela equipe de backend, especialmente nas práticas de garantia da qualidade aplicadas durante as sprints.

A equipe é composta por 10 membros, com a seguinte divisão:

- Um Scrum Master;
- Um Product Owner;
- Um Arquiteto;
- Um Analista de Qualidade;
- Seis Desenvolvedores Backend.

A equipe segue a metodologia Scrum, com sprints de duas semanas. No início de cada sprint, a equipe participa da reunião de planning, onde as atividades do backlog são escolhidas, priorizadas e os objetivos da sprint são definidos. Durante a sprint, as reuniões diárias (dailies) são realizadas para acompanhar o progresso e resolver impedimentos.

No final da sprint, ocorre a reunião de revisão, onde as entregas são avaliadas em relação aos objetivos, e a retrospectiva, que serve para identificar pontos de melhoria no processo. O time usa o Kanban para acompanhamento das tarefas da sprint.

3.2 Processo de garantia da qualidade

O analista de qualidade é responsável por garantir a qualidade das entregas junto ao time, participando ativamente desde a definição dos requisitos até os testes finais das funcionalidades.

O papel do analista começa nos refinamentos técnicos e de negócios, onde ele colabora com os desenvolvedores e o Product Owner (PO). No refinamento de negócios, ele participa da definição dos critérios de aceitação das histórias de usuário, garantindo que estejam bem definidos, claros e testáveis. Esses critérios servirão como base no planejamento de testes. Já no refinamento técnico, o analista acompanha os desenvolvedores na divisão e pontuação das atividades, auxiliando na análise do que será desenvolvido e levantando questionamentos sobre como será feito, com foco no planejamento de testes que serão realizados posteriormente.

Durante as sprints, o profissional de qualidade participa das reuniões de planejamento, definindo as atividades de testes que farão parte do escopo. Também está presente nas dailies, acompanhando o andamento das atividades, reportando o status dos testes e discutindo problemas ou impedimentos.

Para garantir a qualidade das entregas, os desenvolvimentos seguem padrões de código e boas práticas. Além disso, é realizada a revisão de código, na plataforma da Azure, onde, após a codificação de uma funcionalidade, pelo menos outros dois desenvolvedores revisam o código para garantir que esteja correto e aderente aos padrões estabelecidos. O código deve abranger tudo o que for necessário para a execução da nova funcionalidade

e incluir testes de unidade, que são a base da Pirâmide de Testes e fundamentais para garantir a estabilidade do sistema. Após a aprovação no code review, o código é promovido do ambiente local para o ambiente de desenvolvimento, onde os desenvolvedores realizam os testes iniciais da funcionalidade, verificando sua integração com o restante do sistema. O analista de qualidade não tem participação na revisão de código e na codificação de testes de unidade, que são de responsabilidade dos desenvolvedores.

Paralelamente, o profissional de qualidade acompanha o progresso das atividades da sprint atual, preparando os testes à medida que as implementações avançam.

O deploy para o ambiente de homologação, onde o analista realiza os testes, é realizado após a aprovação no ambiente de desenvolvimento de cada funcionalidade pelos desenvolvedores. Então, o analista busca identificar falhas e garantir que os requisitos e critérios de aceitação sejam atendidos antes das funcionalidades serem consideradas prontas para produção.

O analista de qualidade utiliza diversas estratégias e técnicas de testes para validação das funcionalidades no ambiente de homologação. Entre essas estratégias, destaca-se a Pirâmide de Testes, onde a base, composta pelos testes de unidade, é coberta pelos desenvolvedores, sendo responsabilidade do analista atuar na camada intermediária, nos testes de integração, e na camada final, com testes E2E. Como a equipe de backend trabalha com uma grande quantidade de APIs, é ainda mais importante o foco nos testes de integração e, posteriormente, nos E2E.

Caso o analista de qualidade identifique defeitos durante os testes no ambiente de homologação, inicia-se as etapas de gestão de bugs. Os defeitos identificados são registrados como atividades do tipo Bug no quadro de atividades do time na plataforma Jira, onde o time acompanha todas as atividades da sprint e do backlog. O bug é detalhado, incluindo todas as informações coletadas, o fluxo de reprodução e possíveis soluções ou melhorias. Em seguida, o analista informa os desenvolvedores sobre o problema, permanecendo à disposição do desenvolvedor responsável para apoiar em quaisquer dúvidas ou na reprodução do problema.

Após a implementação das correções, o analista de qualidade realiza uma nova rodada de testes para garantir que as falhas foram corrigidas, além de realizar testes regressivos para assegurar que as funcionalidades já validadas continuem funcionando corretamente.

O time já implantou, durante a atuação de um analista de qualidade anterior, um projeto de automação de testes. Porém, devido à falta de manutenção e à mudança de contexto, o projeto foi descontinuado.

Embora o foco principal do analista de qualidade esteja nos testes funcionais, existem algumas iniciativas pontuais de testes não funcionais, como de performance, que

podem ser feitas em colaboração com um time especializado dentro da empresa.

O analista utiliza ferramentas como o Jira para gerenciar bugs e atividades. A documentação dos casos de teste, incluindo cenários a serem validados, passos de execução, status e evidências, também é registrada na ferramenta, permitindo que qualquer membro do time acompanhe o andamento dos testes e sua cobertura.

Além disso, são utilizadas ferramentas específicas para testes de APIs e fluxos backend E2E, como o Postman. Quando há necessidade de consultas na base de dados, o analista recorre ao DBeaver para acessar o PostgreSQL.

A equipe possui uma quantidade considerável de documentação relacionada às atividades do projeto, além de registros técnicos, como a estrutura do código, descrição das funcionalidades e detalhamento das APIs. No entanto, cada documento está na plataforma em que foi originalmente criado, por exemplo, o LucidChart é utilizado para diagramas arquiteturais, o Swagger concentra as informações das APIs e o Confluence abriga a maior parte dos documentos descritivos do projeto. Não há, contudo, um repositório centralizado nem uma organização unificada de links que facilite o acesso a essas informações.

Ao final da sprint, na review, o analista destaca as validações realizadas e se algo ficará pendente para o próximo ciclo. Na retrospectiva, ele apoia o time e o Scrum Master na identificação de pontos positivos, pontos de melhoria e na definição de ações para a próxima sprint.

Com a conclusão das validações, é definido o conjunto de funcionalidades que será promovido para produção no próximo deploy, assegurando que apenas funcionalidades validadas e aprovadas sejam entregues aos usuários.

O fluxo atual de garantia da qualidade pode ser visualizado na Figura 5, que apresenta as principais etapas e responsabilidades ao longo do ciclo de desenvolvimento.

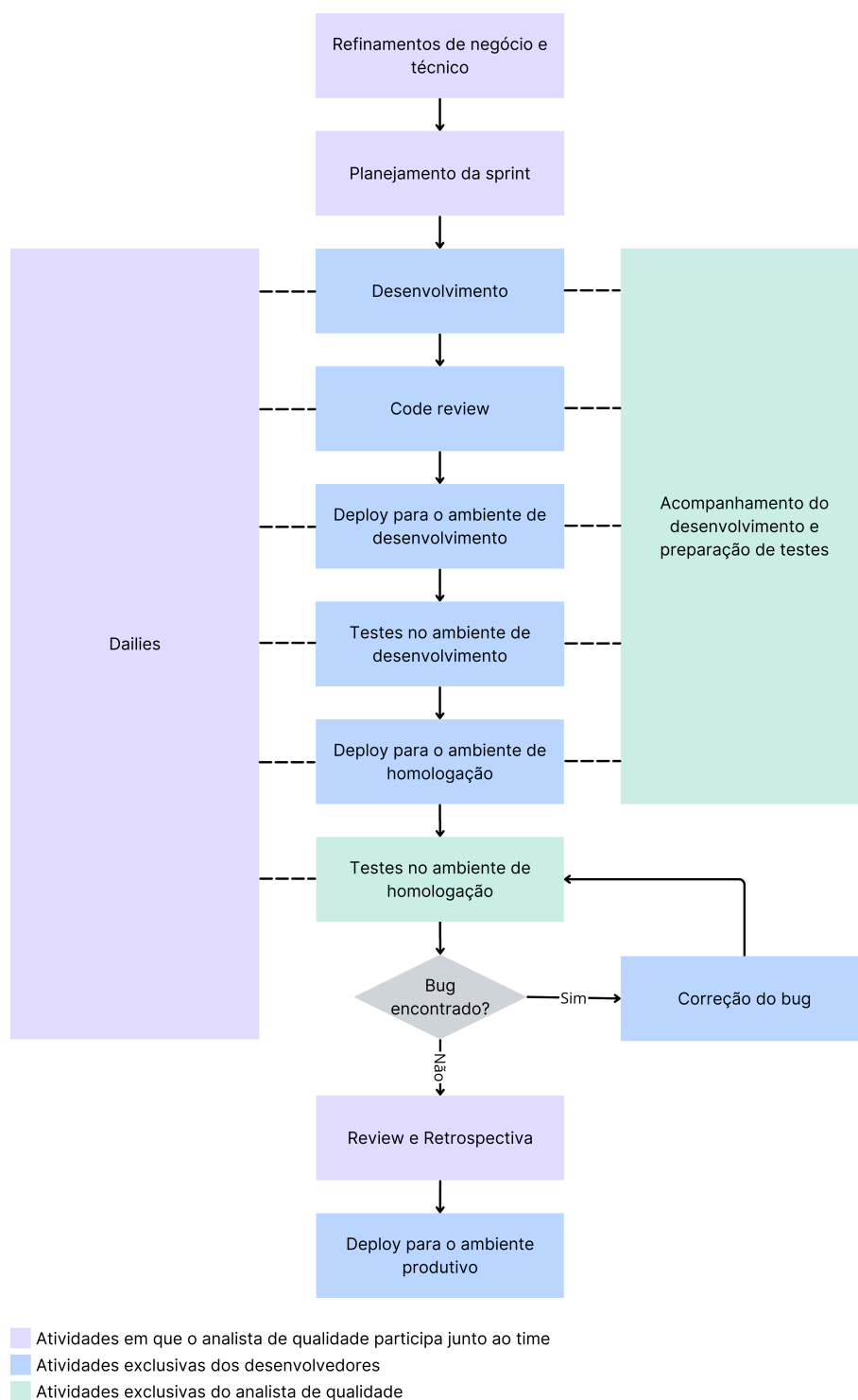


Figura 5 – Diagrama do fluxo atual de Garantia da Qualidade

Fonte: Elaborado pelo autor

O fluxograma representa o processo atual, destacando a atuação do analista de qualidade e sua colaboração com o time. A legenda na imagem indica os responsáveis. As setas mostram a sequência das etapas, enquanto os traços paralelos indicam atividades realizadas simultaneamente.

3.3 Avaliação do impacto das práticas de garantia da qualidade no projeto

O profissional de qualidade desempenha um papel essencial nas entregas do time, sendo fundamental para garantir que elas atendam aos requisitos e critérios de aceitação definidos. Sua atuação integrada à metodologia ágil contribui para a eficiência das entregas, facilitando a detecção precoce de erros, a melhoria contínua das práticas de trabalho e, consequentemente, a qualidade do produto final.

Apesar das práticas adotadas pelo time para assegurar a qualidade do produto, ainda existem desafios que afetam a eficácia das atividades realizadas, impactando diretamente a qualidade das entregas.

A falta de automação nos testes compromete tanto a cobertura quanto a agilidade na execução. Quando realizados manualmente, os testes apresentam uma cobertura limitada, especialmente conforme o número de funcionalidades aumenta. Essa limitação se agrava em testes regressivos, onde a ausência de automação dificulta a validação, já que depende de um único analista, criando um gargalo no fluxo de trabalho. Além disso, o analista acaba repetindo cenários de teste já validados, algo que poderia ser resolvido com a automação desses cenários.

A ausência de integração com Integração Contínua (CI) e Entrega Contínua (CD) agrava ainda mais esse problema. Sem a automação dos testes, as pipelines de CI e CD não podem realizar testes prévios automatizados antes de implementar novas funcionalidades, o que compromete a continuidade dos fluxos validados e prejudica a detecção precoce de erros, já que a execução de testes nas pipelines poderia identificar problemas antes que fossem introduzidos no ambiente.

Outro ponto importante é que o analista de qualidade não participa da revisão de código, realizado exclusivamente entre desenvolvedores. Com isso, a identificação de problemas iniciais, como a ausência de implementação de algum requisito estabelecido, fica limitada. A inclusão do analista de qualidade poderia contribuir para a prevenção de falhas desde as etapas iniciais.

A documentação do projeto não é centralizada em um único repositório, utilizando diversas plataformas como Confluence, Swagger e LucidChart, para detalhar diferentes aspectos. Além disso, não há um local único que concentre todos os links para acessar essas informações, o que dificulta a localização delas durante o desenvolvimento e os testes, prejudicando também a integração de novos profissionais ao time.

Embora o desempenho e o progresso sejam acompanhados por meio de ferramentas específicas, o analista de qualidade não está envolvido nesse monitoramento, e os dados ainda não são avaliados no projeto. Além disso, não são gerados relatórios com métri-

cas de testes e histórico de falhas, o que dificulta a identificação de padrões recorrentes de problemas e prejudica a análise de tendências, impactando a tomada de decisões e, consequentemente, a qualidade do produto.

Por mais que o time já adote diversas ações para garantir a qualidade das entregas, as limitações e dificuldades mencionadas ainda afetam o desempenho da equipe. Esses desafios devem ser considerados na busca por melhorias, principalmente em relação à automação de testes, à integração CI/CD e à criação de métricas que contribuam para a evolução desses processos.

4 Proposta de melhorias para o processo de Garantia da Qualidade na Empresa Omega

Conforme analisado no capítulo anterior, a equipe da empresa Omega já adota práticas relacionadas à garantia da qualidade em suas entregas, havendo por parte da empresa uma busca por melhorias e espaço para que isso ocorra. Para isso, será apresentado um plano de ação com melhorias para o processo atual, visando aperfeiçoar e ampliar as práticas atuais, com a adoção de novas ações, juntamente com a redefinição de algumas responsabilidades.

4.1 Pontos focais do processo de garantia da qualidade

4.1.1 Automação de testes funcionais

A dependência de testes manuais na Empresa Omega tem resultado em uma cobertura limitada e em gargalos no processo de validação, especialmente em cenários de testes regressivos.

A automação de testes é amplamente reconhecida como uma prática essencial no desenvolvimento ágil e em ambientes DevOps, pois proporciona uma execução rápida e contínua de validações, otimizando tempo e recursos, além de promover um feedback ágil para a equipe ([GAROUSI; MäNTYLÄ, 2016](#)). Com isso, o time pode focar em novas funcionalidades e em testes exploratórios, que requerem validações manuais.

No contexto analisado, a adoção de testes automatizados de API, utilizando ferramentas como o RestAssured, pode ampliar a cobertura dos testes, principalmente em cenários regressivos e de maior complexidade. A automatização desses casos permitirá que o analista de qualidade reduza o seu retrabalho em testes repetitivos e dedique-se à criação de novos testes e outras atividades relacionadas à qualidade.

4.1.2 Integração contínua e Entrega Contínua (CI/CD)

Com base na abordagem DevOps, que conecta o desenvolvimento e as operações, a integração contínua (CI), prática de integrar frequentemente o código em um repositório compartilhado, e a entrega contínua (CD), que dá continuidade à CI ao automatizar a liberação de código para os ambientes, favorecem a entrega contínua de valor ao cliente. No cenário analisado, a inexistência dessas práticas limita a validação, pois ela só é iniciada pelo analista de qualidade após a realização do deploy no ambiente. Com a adoção

de pipelines mais avançadas, essas validações poderiam ser iniciadas previamente, ainda durante a integração.

A integração da automação aos fluxos de CI/CD também é uma prática recomendada pela literatura, por proporcionar um feedback mais rápido sobre a qualidade do código e a integração entre os componentes do sistema. Essa prática está alinhada aos princípios do shift-left testing (KINSBRUNER, 2020), contribuindo para a detecção precoce de falhas e evitando a introdução de defeitos em ambientes posteriores.

Portanto, a implementação de fluxos de CI/CD na Empresa Omega, com a execução de testes automatizados, representa uma estratégia alinhada às práticas de DevOps, permitindo que a equipe crie, teste e disponibilize o código com mais qualidade desde os primeiros estágios do desenvolvimento.

4.1.3 Revisão de código (Code Review)

A revisão de código assegura que o código implementado siga as boas práticas e os padrões de qualidade estabelecidos pela equipe, além de identificar problemas estruturais e de lógica (SILVA et al., 2023). A abordagem adotada pela equipe da Empresa Omega, conforme discutido anteriormente, no qual pelo menos dois desenvolvedores realizam a revisão do código, está alinhada com o esperado para essa fase.

A partir da abordagem já existente, o analista de qualidade da equipe pode integrá-la, realizando validações pontuais nas mudanças que serão incorporadas ao repositório principal e aos demais ambientes. Essas validações podem ser mais específicas, considerando que, em alguns casos, o analista pode não dominar a linguagem utilizada no desenvolvimento. Dessa forma, sua análise pode se concentrar em aspectos como o cumprimento dos requisitos definidos e a verificação da implementação completa das funcionalidades previstas.

A inclusão do analista de qualidade na revisão tende a elevar a confiabilidade das entregas, pois adiciona mais uma etapa de verificação antes do deploy do código, reduzindo a probabilidade de introdução de problemas nos ambientes subsequentes e, por fim, em produção. Essa integração também contribui para a melhoria da comunicação no time e para o compartilhamento de conhecimento entre seus integrantes.

4.1.4 Gestão do conhecimento e documentação

Apesar de seguir uma abordagem ágil, na qual a documentação do projeto nem sempre é priorizada, a equipe da Omega já registra uma quantidade significativa de informações nas diversas ferramentas utilizadas, como o Jira e o Confluence. O fato de o time valorizar a documentação é um comportamento valioso para o projeto, mas é necessário

que o acesso a essas informações seja facilitado, garantindo que o conhecimento gerado seja efetivamente aproveitado.

A consulta à documentação não se limita apenas ao desenvolvimento das funcionalidades, mas também é crucial em diversos outros momentos, como durante os testes, na manutenção de funcionalidades anteriores e na adaptação de novos membros à equipe (SOMMERVILLE, 2011).

Embora o time já possua uma quantidade considerável de documentação, surge a necessidade de centralizá-la em um repositório único, como o Confluence. Esse repositório centralizado pode ser organizado por temas, facilitando a navegação e a consulta rápida. Além disso, é importante que esse repositório contenha links para documentos que ainda precisam ser mantidos em outras ferramentas. Em casos onde a documentação é atualizada com frequência, como nas ferramentas específicas de integração e desenvolvimento, pode ser mais prático manter essas informações apenas nesses locais, evitando a necessidade de atualizações redundantes.

A centralização da documentação é um passo importante, mas não é suficiente. O time deve continuar criando e atualizando os materiais, garantindo que o conhecimento registrado seja sempre o mais completo e atualizado possível. Para isso, é preciso definir responsáveis e momentos específicos para a atualização dessas informações. Essas definições podem ser discutidas e estabelecidas dentro do time, garantindo que a documentação seja atualizada de forma recorrente.

4.1.5 Análise de métricas

Durante o ciclo de desenvolvimento da equipe da Empresa Omega, a análise de métricas e dos relatórios gerados pelas ferramentas já utilizadas não é realizada de forma eficaz, muitas vezes nem sendo considerada. Essa ausência compromete a capacidade da equipe de identificar áreas e situações que necessitam de atenção. Embora haja uma grande quantidade de dados disponíveis, eles ainda não são explorados com o objetivo de identificar tendências e promover melhorias, essencial para atingir altos padrões de qualidade.

Conforme discutido na seção 2.3.4, sobre Métricas e Relatórios, a adoção dessas práticas é fundamental para entender melhor a evolução do projeto e o desempenho da equipe. As métricas podem fornecer uma visão clara de como o time está atuando em relação às entregas, qual o ritmo de trabalho, os principais problemas enfrentados e os pontos que demandam maior atenção.

A definição dos indicadores a serem acompanhados deve envolver os responsáveis por cada função. Por exemplo, os desenvolvedores podem focar em métricas relacionadas ao código, como a cobertura de testes de unidade; o analista de qualidade pode acom-

panhar indicadores como a cobertura de testes automatizados e a quantidade de defeitos identificados, buscando a redução de falhas recorrentes; e a liderança do time pode se concentrar em aspectos como a capacidade de entrega por sprint, o que possibilita avaliar se o time está cumprindo os compromissos assumidos ou se ajustes no planejamento são necessários.

Além da definição das métricas, os relatórios gerados pelas ferramentas utilizadas pela equipe, como o Jira, podem ser extremamente valiosos. Esses relatórios concentram e facilitam a visualização das informações mais relevantes. Com base neles, o time pode reservar momentos específicos a cada sprint para analisar os dados coletados e propor ações corretivas de forma mais eficiente.

4.2 Proposta de melhorias do processo de garantia da qualidade para a Omega

O time de backend da empresa Omega já adota práticas de garantia da qualidade em suas atividades. No entanto, diante dos desafios identificados na análise dessas práticas no projeto, há oportunidades de aperfeiçoamento, especialmente na automação de testes, na ampliação do uso de CI/CD e na análise de métricas. Assim, este trabalho propõe um conjunto de ações para aprimorar as práticas da equipe, tornando-as mais eficientes e confiáveis.

1. Automação de testes

- **Objetivo:** Ampliação da cobertura e da agilidade na realização dos testes pelo analista de qualidade.
- **Ações sugeridas:**
 - Introdução de ferramentas adequadas para automação de testes de API, considerando frameworks como RestAssured, em linguagens de programação como Java ou Kotlin. Isso inclui a capacitação do analista de qualidade, caso necessário, para a escrita e manutenção dos scripts de testes.
 - Criação de um projeto de automação ou a inclusão de uma seção dentro do repositório do projeto para a implementação dos testes automatizados.
 - Implementação de testes automatizados, inicialmente, para cenários críticos e fluxos principais, que também serão utilizados durante os testes regressivos.
 - Substituição gradual de testes manuais repetitivos por automação, com foco naqueles de maior complexidade ou maior frequência de execução.

- Definição de um processo de revisão e atualização contínua dos testes automatizados.
- **Desafios e considerações:** A capacitação do analista e sua adaptação às novas ferramentas podem representar um desafio inicial, exigindo tempo e dedicação para o aprendizado das tecnologias envolvidas. Podem ser gerados impactos temporários no fluxo atual de testes. Após essa fase de adaptação e com o início da execução automatizada, tende-se a observar ganhos significativos em agilidade, consistência e reaproveitamento de esforço ao longo das sprints.

2. Integração Contínua (CI) e Entrega Contínua (CD)

- **Objetivo:** Integração dos testes automatizados nas pipelines de CI/CD.
- **Ações sugeridas:**
 - Configuração de pipelines de CI/CD para execução automatizada de testes antes de cada deploy.
 - Estabelecimento de gatilhos para a execução dos testes sempre que uma nova funcionalidade for integrada ao repositório.
 - Monitoramento constante dos testes dentro do fluxo de CI/CD, utilizando dashboards e alertas para garantir a estabilidade das pipelines e agir rapidamente em caso de falhas.
 - Criação de relatórios automáticos sobre os resultados dos testes, com detalhes sobre falhas e cenários de erro.
 - Integração dos relatórios de testes com ferramentas de comunicação do time para que o analista de qualidade receba notificações em tempo real sobre falhas nos testes.
- **Desafios e considerações:** Será necessário disseminar o conhecimento sobre a estrutura das pipelines entre todos os membros do time, incluindo o analista de qualidade, para que ele possa atuar na construção e no acompanhamento das execuções a cada deploy. As automações desenvolvidas na etapa anterior, principalmente os testes regressivos, poderão ser integradas aqui, contribuindo para um ganho importante de tempo. É importante destacar que o acompanhamento das primeiras execuções automatizadas será essencial para garantir que as configurações estejam corretas e que funcionem conforme o esperado.

3. Participação do analista de qualidade no processo de revisão de código

- **Objetivo:** Integração do analista de qualidade no processo de revisão de código (code review).
- **Ações sugeridas:**

- Inclusão do analista de qualidade na revisão das alterações de código submetidas por meio de solicitações formais, chamadas de Pull Requests (PRs), especialmente aquelas que envolvam mudanças consideradas críticas, novas funcionalidades complexas ou correções de bugs com alto impacto, com foco na verificação dos critérios de aceitação e impacto nos testes planejados.
- Utilização de checklists padronizados nos PRs para facilitar a validação dos critérios de qualidade durante o code review.
- Capacitação para o analista de qualidade sobre linguagens de programação e práticas de revisão de código adotadas pelo time.
- **Desafios e considerações:** A inclusão do analista de qualidade na revisão de código exige que os desenvolvedores esteja disposto a compartilhar informações sobre o projeto e os padrões de qualidade adotados. Isso permitirá que o analista tenha uma visão geral durante as validações. O analista também poderá acompanhar os desenvolvedores durante a revisão para entender como ela é feita e quais pontos de verificação são priorizados. Assim como para a automação, a capacitação do analista será fundamental, e o período de adaptação precisa ser considerado.

4. Centralização da documentação

- **Objetivo:** Estruturação de um repositório central para as informações do projeto.
- **Ações sugeridas:**
 - Utilização do Confluence, que já é utilizado no projeto, para centralizar e organizar a documentação do projeto, como descrições detalhadas, requisitos, design e arquitetura, garantindo fácil acesso a todos os membros do time.
 - Definição de uma estrutura clara e consistente para organizar as informações gerais do projeto, incluindo links úteis para ferramentas existentes, como as que documentam os diagramas e as APIs do sistema.
 - Definição de diretrizes para atualização periódica da documentação.
- **Desafios e considerações:** Como o time não precisará criar uma grande quantidade de documentação do zero inicialmente, com foco na organização da documentação existente, pode ser interessante designar responsáveis por realizar essa atividade. Posteriormente, a equipe também deverá definir os momentos e os responsáveis pela criação e atualização das documentações sobre as novas funcionalidades. Além disso, é fundamental estabelecer padrões para

as documentações, garantindo que, independente do profissional envolvido, a estrutura seja mantida.

5. Análise de métricas e relatórios

- **Objetivo:** Criação de métricas que auxiliem na avaliação do processo de desenvolvimento e da qualidade do produto.
- **Ações sugeridas:**
 - Coleta recorrente dos relatórios já gerados pelas ferramentas utilizadas pelo projeto.
 - Utilização dos dados armazenados no Jira para gerar relatórios sobre a execução dos testes, permitindo a análise do histórico de falhas, identificação de padrões de problemas e monitoramento da evolução da cobertura de testes.
 - Análise detalhada do projeto para identificar novas ferramentas de coleta de dados e geração de métricas, que possam ser utilizadas para aprimorar a avaliação da qualidade.
 - Estabelecimento das métricas prioritárias como cobertura de testes, métricas da qualidade do código, velocidade e estabilidade da pipeline de CI/CD e tempo médio para resolução de bugs.
 - Definição de reuniões periódicas para análise dos relatórios coletados, com foco nas tendências e padrões que exigem melhorias. Caso a análise não seja feita por todo o time, os resultados podem ser trazidos para o time durante as reuniões de review em cada sprint.
 - Utilização de planilhas ou outro tipo de repositório centralizado com as informações coletadas, para facilitar a compreensão e acompanhamento.
 - Elaboração e acompanhamento de planos de ação para endereçar as melhorias identificadas durante as análises.
- **Desafios e considerações:** A definição de quais métricas acompanhar e como interpretá-las para gerar ações pode representar um desafio inicial. O Scrum Master e o Product Owner, acostumados a lidar com esse tipo de informação, podem guiar o time. Contudo, é importante que toda a equipe se engaje em aprender sobre essas e outras métricas, para aplicá-las em suas atividades diárias. A adoção dessas práticas é crucial para garantir que as mudanças implementadas pela equipe não se limitem às propostas iniciais, mas evoluam junto ao amadurecimento do time.

Na Figura 6, é apresentado o novo fluxo de garantia da qualidade, com a inclusão de novas etapas e a redefinição de algumas responsabilidades ao longo das fases de desenvolvimento.

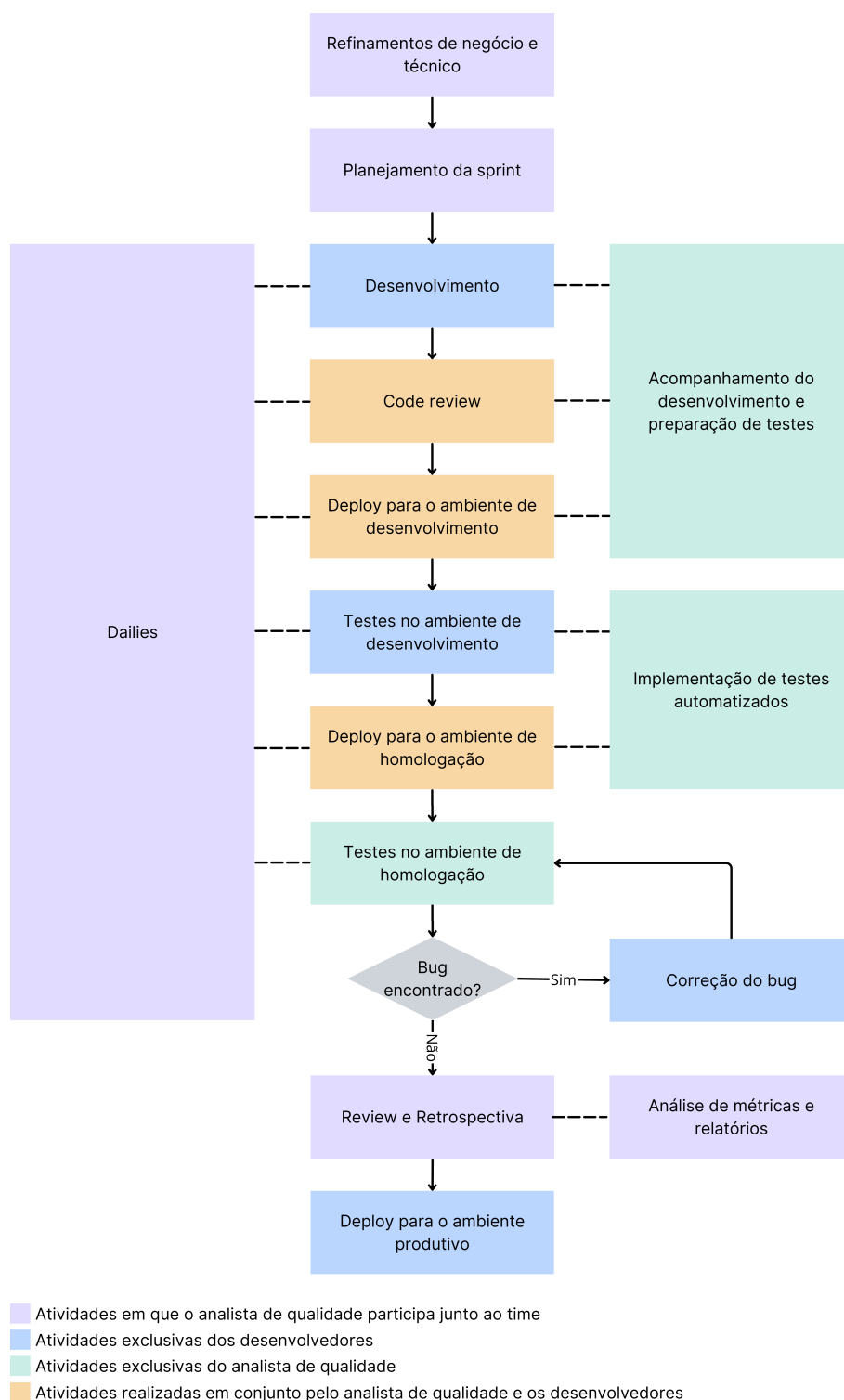


Figura 6 – Diagrama do fluxo proposto para a Garantia da Qualidade

Fonte: Elaborado pelo autor

Com base nas atividades propostas, o novo fluxograma reforça a atuação do analista de qualidade em colaboração com o time. Foram incluídas novas atividades e redefinidas algumas responsabilidades, incluindo uma nova cor para representar atividades que agora contam com a participação do analista de qualidade. Embora não esteja explicita-

mente representada no diagrama, a gestão e a utilização da documentação são práticas ativas e relevantes em todas as etapas do projeto.

4.3 Plano de ação sugerido para implementação da proposta

A partir das melhorias propostas, o plano de ação a seguir tem como objetivo orientar a implementação gradual das mudanças no processo de garantia da qualidade. Este plano visa garantir que as ações sejam adotadas respeitando as necessidades e o ritmo do time.

1. Alinhamento inicial

Para iniciar a implementação, é essencial que todo o time participe de uma reunião para entender as propostas, discutir as ações e alinhar expectativas. Neste momento, será possível definir os prazos e os responsáveis por cada ação. A participação ativa de todos os membros é crucial para garantir o comprometimento das mudanças.

2. Análise de ferramentas e tecnologia

O analista de qualidade deverá iniciar o processo identificando as ferramentas e tecnologias mais adequadas para a automação de testes no contexto do time, além de avaliar as habilidades necessárias para sua implementação. A liderança do time deve acompanhar, identificando possíveis necessidades de licenças ou treinamentos. Com essa base, será possível realizar uma análise das prioridades de testes para automação, com a definição de uma implementação gradual, priorizando as áreas mais críticas.

3. Integração contínua

Com os primeiros cenários de testes automatizados estruturados, será o momento de integrar esses testes nas pipelines de CI/CD. Os desenvolvedores e o analista de qualidade devem se reunir para definir uma estratégia de integração contínua. Essa estratégia pode envolver configurações e ajustes nas pipelines existentes e, caso haja um profissional de DevOps na empresa, ele poderá fornecer suporte técnico.

4. Participação nas revisões de código

O momento de introduzir a atuação do analista de qualidade nas revisões de código precisa ser cuidadosamente planejado. Considerando as novas responsabilidades do analista, será necessário definir o momento mais adequado para essa participação, a fim de evitar a sobrecarga de tarefas. Após essa definição, os desenvolvedores podem ajudar a estabelecer critérios claros e organizar reuniões de compartilhamento de conhecimento, garantindo que o analista esteja bem preparado para validar os critérios de qualidade.

5. Centralização da documentação

O time de desenvolvimento será responsável por levantar toda a documentação

existente do projeto, revisar sua estrutura e garantir que as informações sejam organizadas de maneira eficiente no repositório central. A plataforma escolhida para centralizar os documentos, será configurada conforme os padrões estabelecidos para garantir fácil acesso e atualização constante.

6. Definição e acompanhamento de métricas

Durante a implementação, é fundamental que a equipe já comece a definir as métricas que serão acompanhadas. À medida que as automações de teste forem implementadas, métricas como o número de testes automatizados e a quantidade de defeitos identificados podem ser monitoradas. Da mesma forma, ao automatizar o deploy nas pipelines, a integração de relatórios automáticos facilitará o acompanhamento em tempo real.

7. Apoio do Scrum Master

O Scrum Master desempenhará um papel fundamental durante a implementação das novas práticas, garantindo que o time se adapte às mudanças de forma fluida. Ele será responsável por monitorar o progresso da equipe e fornecer suporte nas dificuldades que surgirem. Todos os membros do time devem estar cientes de que ajustes nos prazos e nas ações podem ser necessários.

4.4 Resultados esperados e possíveis lições aprendidas

A implementação das ações sugeridas deverá contribuir para a melhoria contínua das atividades da equipe, com foco na automação, na agilidade das validações e no aumento da qualidade do produto final. O analista de qualidade terá um papel mais integrado ao desenvolvimento, ampliando suas atividades para agregar mais valor ao trabalho da equipe.

A automação tornará a execução dos testes mais ágil e eficiente, aumentando a cobertura de cenários, reduzindo o tempo de execução e o esforço manual em testes repetitivos. No entanto, esse processo demanda um período de aprendizado, um investimento inicial de tempo e esforço, além de uma manutenção recorrente dos testes implementados. A capacitação do analista de qualidade será essencial para garantir a eficácia da automação ao longo do tempo. O time deve reconhecer a necessidade dessa manutenção e da capacitação constante, tanto do analista de qualidade quanto dos desenvolvedores.

Com a automação integrada ao desenvolvimento e ao deploy, espera-se um fluxo de trabalho mais ágil e confiável. A execução automatizada dos testes permitirá validar rapidamente se alterações no código afetam fluxos críticos, facilitando a identificação de falhas precocemente. Com isso, será possível ter feedbacks rápidos e eficazes, além de melhorar a colaboração entre os desenvolvedores e o analista de qualidade. Conforme a cobertura da automação for aumentando, o time perceberá cada vez mais como a automação melhora

a estabilidade das entregas e como a colaboração entre os membros é fundamental.

A participação do analista de qualidade na revisão de código tornará mais eficiente a comunicação e a identificação antecipada de problemas no código, evitando retrabalho e aprimorando a qualidade das entregas. Para isso, será necessário um período de adaptação e treinamento, além de ajustes no fluxo de revisão. Após implementado, o time deverá perceber como essa prática conjunta melhora a qualidade do código e oferece uma visão mais completa do produto, unindo a visão do analista com a dos desenvolvedores.

A centralização da documentação facilitará o acesso a informações essenciais do projeto, reduzindo o tempo gasto na busca por dados e agilizando a integração de novos membros ao time. No entanto, será necessário um esforço inicial para organizar a documentação existente e estabelecer um processo contínuo de atualização, com responsabilidades bem definidas entre os integrantes do time. Ao implementar esse processo, ficará clara a importância de manter a documentação atualizada, tornando o trabalho mais prático e reduzindo retrabalho.

O uso de métricas permitirá uma análise mais precisa, possibilitando a identificação de padrões e áreas de melhoria com base em dados. A escolha de métricas relevantes, a coleta de informações e a análise periódica dos relatórios serão fundamentais para garantir uma otimização contínua da qualidade e da capacidade do time. A equipe desenvolverá um olhar mais analítico, compreendendo melhor o papel das métricas na tomada de decisões.

Essas mudanças também exigem um ajuste cultural dentro do time, pois afetam rotinas, responsabilidades e a mentalidade de qualidade de todos os envolvidos. A adesão e o engajamento do time serão muito importantes para o sucesso da implementação. Além disso, é essencial que a equipe esteja aberta a mudanças, assegurando que as práticas adotadas se ajustem à realidade do projeto.

O sucesso da implementação deve ser avaliado por meio de indicadores definidos para cada iniciativa, como a redução no tempo de execução dos testes regressivos e o aumento da cobertura dos testes automatizados. A proposta apresentada aplica o conceito de shift-left, integrando a qualidade desde as fases iniciais. Essa integração não apenas reduz custos de correção de defeitos, mas também promove uma cultura de responsabilidade coletiva pela qualidade do produto, evitando que essa responsabilidade seja atribuída somente ao analista de qualidade, resultando em entregas mais completas e eficazes.

As melhorias propostas neste capítulo estão alinhadas às práticas recomendadas de garantia da qualidade em ambientes ágeis. Espera-se que sua implementação contribua para a otimização dos processos internos da empresa e gere benefícios consistentes ao longo do tempo.

5 Conclusão

Este trabalho dedicou-se a uma análise detalhada de diversos aspectos que podem ser implementados na garantia da qualidade em times ágeis. O estudo de caso apresentado retrata um cenário que pode inclusive ser comum em outros times ou empresas, onde algumas ações de qualidade já são adotadas, mas ainda podem ser ampliadas e devem ser constantemente analisadas em busca de melhorias. A análise demonstrou que, embora diversas ações em prol da qualidade já estejam em prática, há espaço para expandi-las e fortalecê-las.

O foco dado ao papel do analista de qualidade justifica-se por sua relevância nas práticas ágeis, com atuação em diversas frentes que influenciam diretamente na qualidade das entregas. Contudo, é importante frisar que a qualidade é uma responsabilidade que deve ser compartilhada por todos os membros da equipe.

Este estudo analisou práticas, metodologias e papéis envolvidos no desenvolvimento ágil, propondo melhorias para tornar essas abordagens mais eficazes ao longo do ciclo de vida do projeto. A proposta, detalhada no capítulo anterior, concentrou-se em cinco pilares principais: automação de testes, integração e entrega contínuas, participação do analista de qualidade na revisão de código, centralização da documentação e análise de métricas.

Os pontos selecionados consideram o estágio atual da equipe e foram priorizados por apresentarem maior viabilidade de aplicação no curto e médio prazo, de acordo com a disposição do time e da empresa em seguir investindo na evolução de suas práticas e na qualidade do produto entregue. Espera-se que a equipe da Empresa Omega possa aplicar o plano apresentado e, mesmo diante dos desafios esperados, alcance os resultados desejados após o período de adaptação.

Para os demais projetos da Empresa Omega, assim como para outras empresas do setor, este trabalho pode servir como ponto de partida para o estudo e a adoção de práticas ágeis no cotidiano das equipes, com foco na colaboração e na melhoria contínua como caminho para a entrega de softwares com cada vez mais qualidade. A consolidação de uma cultura de qualidade representa, então, um importante diferencial competitivo.

Em um contexto ágil e em um mercado cada vez mais dinâmico, os projetos devem estar abertos a transformações eficazes, adotando a análise constante e a otimização de processos como parte da rotina.

Para futuras pesquisas e iniciativas, destaca-se a ampliação da utilização de testes não funcionais pela equipe da Empresa Omega. Apesar de já existirem iniciativas pontuais,

esses testes ainda não são aplicados de forma ampla. A implementação de testes, como os de segurança, pode representar um próximo passo importante rumo a uma abordagem mais abrangente da garantia da qualidade, com impactos diretos na qualidade técnica do produto final e na satisfação dos usuários.

Outra possibilidade seria a realização de estudos focados na aplicação prática de um plano de melhorias como o proposto neste trabalho, considerando sua implementação em equipes já em operação. Esse tipo de investigação poderia explorar com maior profundidade a definição de cronogramas, a distribuição de responsabilidades e as estratégias de acompanhamento, oferecendo direcionamentos para adoções mais eficientes em contextos reais.

Referências

- AGILE, S. of. **The 16th State of Agile Report**. 2022. Acesso em: 20 mar. 2024. Disponível em: <<https://info.digital.ai/rs/981-LQX-968/images/SOA16.pdf>>. Citado na página 15.
- ALESSIO, S.; SABADIN, N.; ZANCHETT, P. **Processos de Software**. 2017. Acesso em: 20 jan. 2025. Disponível em: <https://edisciplinas.usp.br/pluginfile.php/7876207/mod_resource/content/1/processo-produto.pdf>. Citado na página 14.
- APIDOG. **As 20 Principais Ferramentas de Teste Automatizado de Código Aberto e Gratuitas**. 2024. Acesso em: 10 fev. 2025. Disponível em: <<https://apidog.com/pt/blog/automated-testing-tools/>>. Citado na página 27.
- ASSURED, J. R. **REST Assured**. 2024. <<https://rest-assured.io/>>. Acessado em: 06 abr. 2025. Citado na página 25.
- ATLASSIAN. **Confluence Documentation**. Acessado em: 29 abr. 2025. Disponível em: <<https://support.atlassian.com/confluence-cloud/resources/>>. Citado na página 23.
- _____. **Jira Documentation**. Acessado em: 29 abr. 2025. Disponível em: <<https://confluence.atlassian.com/jira>>. Citado na página 20.
- AZZAM, F. Quality assurance in agile software development - literature review. **ResearchGate**, 2022. Acesso em: 05 fev. 2025. Disponível em: <<https://www.researchgate.net/publication/362554542>>. Citado na página 21.
- BARBOSA, F. B.; TORRES, I. V. O teste de software no mercado de trabalho. **Revista Tecnologias em Projeção**, 2011. Acesso em: 26 fev. 2024. Disponível em: <<https://revista.faculdadeprojecao.edu.br/index.php/Projecao4/article/view/82>>. Citado na página 10.
- BECK, K.; ANDRES, C. **Extreme Programming Explained**. 2012. Acesso em: 25 ago. 2024. Disponível em: <<https://ptgmedia.pearsoncmg.com/images/9780321278654/samplepages/9780321278654.pdf>>. Citado na página 19.
- CESAR, W. H. C. A.; CESAR, F. F.; MONTEIRO, L. A. **Pesquisa em Métricas para melhoria, medição e avaliação de software**. 2018. Acessado em: 12 abr. 2025. Disponível em: <<https://www.researchgate.net/publication/334684548>>. Citado na página 22.
- DUDZIAK, T. **eXtreme Programming An Overview**. 2000. Acesso em: 06 set. 2024. Disponível em: <https://csis.pace.edu/~marchese/CS616/Agile/XP/XP_Overview.pdf>. Citado na página 19.
- GAEA. Afinal, o que é quality assurance? 2018. Acesso em: 26 fev. 2024. Disponível em: <<https://gaea.com.br/afinal-o-que-e-quality-assurance/>>. Citado na página 10.

GANTAYAT, S. **Everything You Need to Know About Quality Management in Agile**. 2022. Acesso em: 26 fev. 2024. Disponível em: <<https://agilemania.com/quality-management-in-agile>>. Citado na página 10.

GAROUSI, V.; MÄNTYLÄ, M. V. **When and what to automate in software testing? A multi-vocal literature review**. [S.l.]: Elsevier B.V., 2016. 92-117 p. Acesso em: 10 fev. 2025. Citado 2 vezes nas páginas 26 e 38.

HALVORSEN, H.-P. **Software Development A Practical Approach!** [s.n.], 2020. Acesso em: 20 mar. 2024. ISBN 9788269110609. Disponível em: <<https://www.halvorsen.blog>>. Citado na página 14.

INMETRICS. **O Futuro da Qualidade de Software**. 2023. Acesso em: 25 fev. 2024. Disponível em: <https://lp.inmetrics.com.br/inscricao-estudo-tgt-pesquisa-qualidade?utm_source=Social&utm_medium=Organic&utm_campaign=Ofuturodaqualidade>. Citado na página 10.

INVILLIA. **O que é QA Ágil e como se diferencia do tradicional**. 2025. Acesso em: 05 fev. 2025. Disponível em: <<https://insights.invillia.ai/pt/o-que-e-qa-agil-como-se-diferencia-do-tradicional/>>. Citado 2 vezes nas páginas 20 e 21.

ISTQB, I. S. T. Q. B. **Certified Tester Foundation Level Syllabus CTFL**. 2023. Acesso em: 26 fev. 2024. Disponível em: <https://bcr.bstqb.org.br/docs/syllabus_ctfl_4.0br.pdf>. Citado 5 vezes nas páginas 10, 20, 23, 24 e 27.

KINSBRUNER, E. **ESSENTIAL GUIDE TO SHIFT YOUR TESTING LEFT EBOOK**. 2020. Acesso em: 20 set. 2024. Disponível em: <<https://em360tech.com/sites/default/files/2020-10/Essential-Guide-to-Shift-Your-Testing-Left.pdf>>. Citado 2 vezes nas páginas 28 e 39.

KOSCHEVIC, M. T. **GERENCIAMENTO DE PROCESSOS COM METODOLOGIAS ÁGEIS**. 2011. Acesso em: 20 mar. 2024. Disponível em: <https://repositorio.utfpr.edu.br/jspui/bitstream/1/23527/2/MD_ENGESS_I_2012_15.pdf>. Citado na página 15.

LEBLANC, S. E. R.; HAGAR, J. D.; HOEK, A. V. D.; PETRE, M.; KUHN, D. R.; KACKER, R. N.; LEI, Y. **Software Metrics A Rigorous and Practical Approach** Chapman Hall/CRC Innovations in Software Engineering and Software Development Software Designers in Action: A Human-Centric Look at Design Work Introduction to Combinatorial Testing. [S.l.]: CRC Press, 2015. Acessado em: 10 abr. 2025. Citado na página 22.

LI, J. **Agile Software Development**. 2017. Acesso em: 03 abr. 2024. Disponível em: <https://www.inf.ufpr.br/lmpres/2017_2/ci090/aula3/MET_article_agile-software-development_li.pdf>. Citado na página 30.

LIMA, A. **Métodos Ágeis - O que são e como aplicar?** 2022. Acesso em: 20 mar. 2024. Disponível em: <<https://irislab.ce.gov.br/wp-content/uploads/2022/12/Guia-de-Metologias-Ageis.pdf>>. Citado na página 15.

LIMA, D. P. de C.; LEAL, E. D.; MESQUITA, L. J. da S. Benefícios da utilização da pirâmide de teste em uma aplicação mobile. **Revista Interface Tecnológica**, Interface Tecnológica, v. 20, p. 63–75, 10 2023. Acesso em: 18 set. 2024. Disponível em: <https://www.researchgate.net/publication/374908489_BENEFICIOS_DA_UTILIZACAO_DA_PIRAMIDE_DE_TESTE_EM_UMA_APLICACAO_MOBILE>. Citado na página 25.

MAGOWAN, K. **SHIFT LEFT TESTING: WHAT, WHY HOW TO SHIFT LEFT**. 2020. Acesso em: 20 set. 2024. Disponível em: <<https://blogs.bmc.com/what-is-shift-left-shift-left-testing-explained/?print-posts=pdf>>. Citado na página 20.

MOUSAEI, M. **REVIEW ON ROLE OF QUALITY ASSURANCE IN WATERFALL AND AGILE SOFTWARE DEVELOPMENT MAHDI MOUSAEI**. [S.l.], 2020. v. 5. Acesso em: 07 fev. 2025. Disponível em: <www.jseis.org>. Citado na página 20.

MUDASSAR, S.; KHAN, A. **Mastering API Testing A Comprehensive Guide to Modern API Testing Techniques and Best Practices**. 2024. Acessado em: 06 abr. 2025. Disponível em: <<https://www.researchgate.net/publication/378768605>>. Citado na página 25.

NETO, A. C. D. **Introdução a Teste de Software**. [S.l.], 2007. Acesso em: 02 abr. 2025. Disponível em: <<http://www.projectcartoon.com/cartoon/611>>. Citado na página 25.

NIDHRA, S. Black box and white box testing techniques - a literature review. **International Journal of Embedded Systems and Applications**, Academy and Industry Research Collaboration Center (AIRCC), v. 2, p. 29–50, 6 2012. Acesso em: 08 fev. 2025. Citado na página 20.

NIKOLOVA, Z. Testing strategies in an agile context. In: _____. [S.l.]: Springer International Publishing, 2019. p. 111–121. ISBN 9783030295097. Acesso em: 02 abr. 2024. Citado 2 vezes nas páginas 21 e 30.

NOVAES, J. Após aplicativo fora do ar, itaú é punido e pode ter prejuízo financeiro. **FDR, Economia Simplificada**, 2023. Acesso em: 26 fev. 2024. Disponível em: <<https://fdr.com.br/2023/08/11/apos-aplicativo-fora-do-ar-itaue-punido-e-pode-ter-prejuizo-financeiro/>>. Citado na página 11.

OLIVEIRA, B. H. **Qualidade de software no desenvolvimento com métodos ágeis**. [S.l.], 2014. Acessado em: 10 abr. 2025. Citado na página 29.

PRESSMAN, R. S. **Engenharia de Software, Uma Abordagem Profissional**. [S.l.]: AMGH, 2016. Acessado em: 12 abr. 2025. Citado na página 23.

RAUT, L. P.; WAKODE, R. B.; TALMALE, P.; LTD, B. S. P. **Overview on Kanban Methodology and its Implementation Overview on Kanban Methodology and its Implementation Student 2 Assistant Professor 3 Production Manager 1,2 Department of Mechanical Engineering**. 2015. 2321-0613 p. Acesso em: 20 jul. 2024. Disponível em: <<https://www.researchgate.net/publication/280865949>>. Citado na página 18.

RIBEIRO, G. D. **Automação de testes aplicados ao DevOps**. 2019. Acesso em: 22 fev. 2024. Disponível em: <<https://repositorio.ufu.br/bitstream/123456789/26515/1/Automa%c3%a7%c3%a3oTestesAplicados.pdf>>. Citado na página 29.

ROGER, G.; LIMA, C. Benefícios das metodologias ágeis no gerenciamento de projetos de tecnologia da informação (ti). 2015. ISSN 2179-5568. Acesso em: 24 fev. 2024. Disponível em: <<https://docplayer.com.br/16101207-Beneficios-das-metodologias-ageis-no-gerenciamento-de-projetos-de-tecnologia-da-informacao-ti.html>>. Citado na página 10.

RUSSELL, R. J. **Chapter 4*: QUALITY ASSURANCE**. 2011. Acesso em: 07 set. 2024. Disponível em: <<https://www.researchgate.net/publication/237287165>>. Citado na página 20.

SANTOS, E. W. dos; NUNES, I. Investigating the effectiveness of peer code review in distributed software development based on objective and subjective data. **Journal of Software Engineering Research and Development**, Sociedade Brasileira de Computacao - SB, v. 6, 12 2018. Acessado em: 08 abr. 2025. Citado na página 21.

SARATKAR, D. A. N. An article on importance of software technologies in business and management science. **International Journal of Engineering Applied Sciences and Technology**, 2019. Acesso em: 24 fev. 2024. Disponível em: <<https://www.ijeast.com/papers/291-294,Tesma404,IJEAST.pdf>>. Citado na página 10.

SATO, D. Devops - na prática: entrega de software confiável e automatizada. **Casa do Código**, 2014. Acesso em: 11 fev. 2025. Citado 3 vezes nas páginas 27, 28 e 29.

SCHNOELLER, G.; SANTOS, D. **EXTREME PROGRAMMING: METODOLOGIA ÁGIL**. 2013. Acesso em: 25 ago. 2024. Citado na página 19.

SCHWABER, K.; SUTHERLAND, J. **Guia do Scrum**. 2013. Acesso em: 30 abr. 2024. Disponível em: <<https://scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf>>. Citado 3 vezes nas páginas 15, 16 e 17.

SHAHIN, M.; BABAR, M. A.; ZHU, L. **Continuous Integration, Delivery and Deployment: A Systematic Review on Approaches, Tools, Challenges and Practices**. [S.l.]: Institute of Electrical and Electronics Engineers Inc., 2017. 3909-3943 p. Acesso em: 10 fev. 2025. Citado 2 vezes nas páginas 27 e 28.

SILVA, D. D. I. D.; PABASARA, W. A. C.; SANGKAVI, S. V.; L.G.A.T.D, W.; W.M.K.H, W.; S.A, R. The effectiveness of code reviews on improving software quality: An empirical study. **International Journal of Recent Technology and Engineering (IJRTE)**, v. 12, p. 1–10, 7 2023. ISSN 22773878. Acessado em: 08 abr. 2025. Disponível em: <<https://www.ijrte.org/portfolio-item/B76660712223/>>. Citado 2 vezes nas páginas 21 e 39.

SOFTWARE, A. A. B. das Empresas de. **Mercado Brasileiro de Software: panorama e tendências, 2023**. [s.n.], 2023. Acesso em: 25 fev. 2024. Disponível em: <<https://abes.com.br/dados-do-setor/>>. Citado na página 10.

- SOMMERVILLE, I. **Engenharia de Software**. Pearson Prentice Hall, 2011. Acesso em: 09 mar. 2024. ISBN 9788579361081. Disponível em: <<https://www.facom.ufu.br/~william/Disciplinas%202018-2/BSI-GSI030-EngenhariaSoftware/Livro/engenhariaSoftwareSommerville.pdf>>. Citado 4 vezes nas páginas 14, 15, 22 e 40.
- TSAI, W. T.; BAI, X.; PAUL, R.; SHAO, W.; AGARWAL, V. End-to-end integration testing design. **Proceedings - IEEE Computer Society's International Computer Software and Applications Conference**, p. 166–171, 2001. ISSN 07303157. Acesso em: 06 abr. 2025. Citado na página 25.
- ULLAH, M. I.; WAQAR, .; ZAIDI, A. **Quality Assurance Activities in Agile-Philosophy to Practice**. 2009. Acesso em: 03 abr. 2024. Disponível em: <https://www.cin.ufpe.br/~scls/Claudia/QA_Activities_in_Agile.pdf>. Citado na página 11.
- UMAR, M. A. **Comprehensive Study of Software Testing: Categories, Levels, Techniques, and Types**. 2020. Acesso em: 04 abr. 2025. Disponível em: <https://www.researchgate.net/publication/342579919_A_Study_of_Software_Testing_Categories_Levels_Techniques_and_Types>. Citado na página 24.
- VOCKE, H. **The Practical Test Pyramid**. 2018. Acesso em: 18 set. 2024. Disponível em: <<https://martinfowler.com/articles/practical-test-pyramid.html>>. Citado 2 vezes nas páginas 25 e 26.