QAs Rumo à Transformação:

Implementando Shift Left Testing

Vivian Lima

SUMÁRIO

INTRODUÇÃO

INTRODUÇÃO

Nos últimos anos, a indústria de desenvolvimento de software testemunhou uma mudança significativa na forma como as equipes abordam o processo de teste. A abordagem tradicional de testar no final do ciclo de desenvolvimento de software já não é suficiente para atender às demandas de velocidade, qualidade e inovação que os clientes e mercados atuais exigem. É aqui que entra o conceito de Shift Left Testing.

O que é Shift Left Testing?

Shift Left Testing é mais do que uma simples mudança de cronograma. É uma abordagem que enfatiza a realização de atividades de teste o mais cedo possível no ciclo de vida do desenvolvimento de software. Tradicionalmente, o teste era realizado apenas no final do processo de desenvolvimento, após a implementação do código. Com o Shift Left, as atividades de teste são movidas para a esquerda do ciclo de desenvolvimento, integrando o teste desde as fases iniciais do processo.

Por que é importante para os QAs?

Para os profissionais de garantia de qualidade (QAs), o Shift Left Testing representa uma mudança fundamental na forma como eles abordam seu trabalho. Não se trata apenas de realizar testes mais cedo no ciclo de desenvolvimento, mas sim de se tornar parte integrante do processo de desenvolvimento desde o início. Isso não apenas ajuda a identificar e corrigir problemas mais cedo, mas também promove uma cultura de qualidade em toda a equipe de desenvolvimento.

Por que o Shift Left é Importante para as Empresas?

O Shift Left Testing é vital para as empresas que buscam permanecer competitivas em um mercado em constante evolução. Ao antecipar os testes e integrar a qualidade desde o início do processo de desenvolvimento, as empresas podem acelerar a entrega de produtos de alta qualidade, reduzir os custos associados a correções tardias de defeitos e aumentar a satisfação do cliente. Além disso, o Shift Left promove uma cultura de colaboração e responsabilidade compartilhada entre as equipes de desenvolvimento e teste, resultando em uma melhor comunicação, eficiência e produtividade geral. Este e-book explora em detalhes como implementar com sucesso o Shift Left Testing e colher os muitos benefícios que ele oferece às empresas que adotam essa abordagem inovadora.

Objetivo do e-book

Este e-book foi projetado com um objetivo em mente: capacitar os QAs a implementar com sucesso o Shift Left Testing em suas organizações. Ao longo das próximas páginas, exploraremos os fundamentos do Shift Left Testing, os benefícios que ele oferece, as técnicas e práticas essenciais, as ferramentas necessárias para implementação e como adaptar essa abordagem para diferentes contextos de equipe e desenvolvimento. Nosso objetivo é fornecer um guia abrangente e prático que ajude os QAs a liderar a transformação rumo a uma abordagem de teste mais eficaz e colaborativa.

.

Capítulo 1

Fundamentos do Shift Left Testing

1.1 – O que é Shift Left Testing?

Antes de mergulharmos nos detalhes do que é o Shift Left Testing, é importante esclarecer o que não é. Shift Left Testing não se trata simplesmente de mover os testes para a esquerda do cronograma de desenvolvimento sem qualquer mudança significativa na abordagem ou no envolvimento dos QAs. Na verdade, é uma mudança de mentalidade e prática que vai além da simples reorganização do cronograma.

O que NÃO é Shift Left Testing?

Shift Left Testing não é apenas sobre adiantar o momento em que os testes são realizados. Não se trata de uma abordagem em que os QAs continuam a ser vistos como uma equipe separada e isolada do processo de desenvolvimento, apenas entrando em cena quando o código está pronto para ser testado. Essa mentalidade tradicional leva a uma série de desafios, incluindo a descoberta de problemas significativos apenas no final do ciclo de desenvolvimento, resultando em atrasos, retrabalho e, em última análise, em produtos de qualidade inferior.

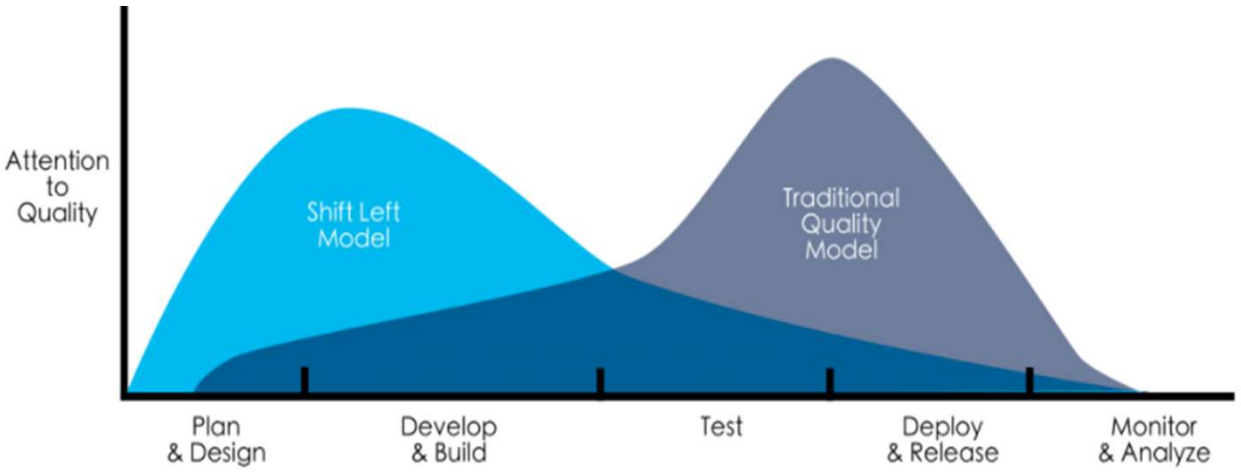
1.1 – O que é Shift Left Testing?

O Status Quo nos Ambientes Ágeis

Mesmo em equipes ágeis, onde a colaboração e a entrega contínua são incentivadas, a mentalidade de "*tá pronto, só falta testar!*" ainda é prevalente em muitos casos. As iterações curtas e o foco na entrega rápida muitas vezes levam as equipes a deixar os testes para o final de cada ciclo. Nesse cenário, a pessoa de QA é frequentemente vista como a última linha de defesa antes do lançamento, em vez de ser um membro integral e proativo da equipe desde o início.

Gráfico: Comparação entre Abordagem Tradicional e Shift Left

A imagem abaixo ilustra a atenção que é dada à qualidade no modelo *Shift-Left* e no modelo tradicional



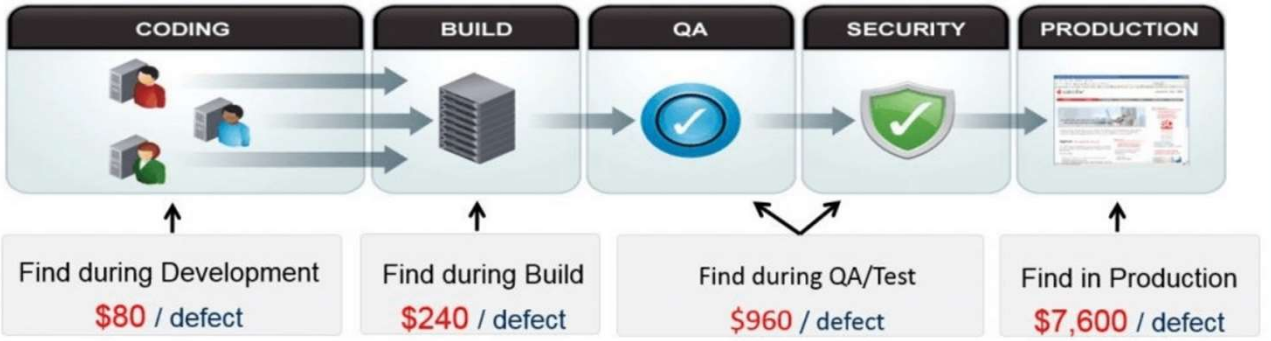
Fonte da imagem: https://blog.onedaytesting.com.br/shift-left-testing/

1.2 – Por que o Shift Left é Importante?

É fundamental compreender por que o Shift Left Testing desempenha um papel tão significativo no cenário atual de desenvolvimento de software, especialmente em um contexto de metodologias ágeis.

Custo do Bug e Redução de Defeitos

O custo de corrigir um defeito aumenta exponencialmente à medida que avançamos no ciclo de desenvolvimento. Corrigir um bug após o lançamento do produto pode ser até 100 vezes mais caro do que corrigi-lo durante a fase de design. Isso se deve não apenas ao tempo e aos recursos necessários para identificar e corrigir o problema, mas também aos custos associados à reputação da marca e à perda de confiança do cliente. O Shift Left Testing visa mitigar esse custo ao identificar e corrigir defeitos o mais cedo possível no processo de desenvolvimento. A imagem abaixo demonstra claramente como o custo de um bug aumenta exponencialmente à medida que avançamos no ciclo de desenvolvimento. Portanto, identificar e corrigir defeitos o mais cedo possível, antes que eles cheguem ao cliente, é crucial para reduzir custos e garantir a satisfação do cliente.

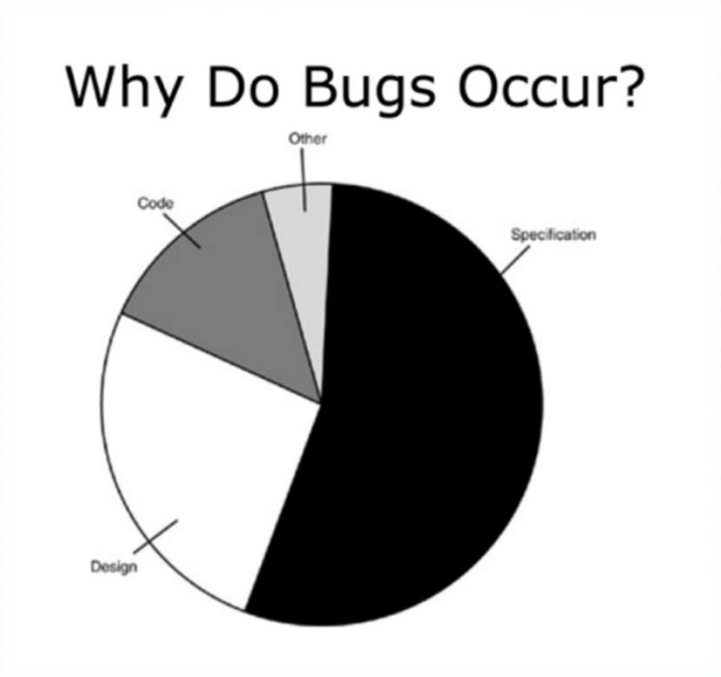


Fonte da imagem: https://www.bmc.com/blogs/what-is-shift-left-shift-left-testing-explained//

1.2 – Por que o Shift Left é Importante?

Entendimento do Requisito e Redução de Erros

Muitos defeitos em software surgem de uma falta de entendimento claro dos requisitos do cliente ou do usuário final. O Shift Left Testing incentiva uma abordagem colaborativa desde o início do processo de desenvolvimento, permitindo que os QAs trabalhem em estreita colaboração com os desenvolvedores e as partes interessadas para garantir que todos tenham uma compreensão clara dos requisitos e expectativas. Isso ajuda a reduzir significativamente o número de erros e retrabalhos associados a mal-entendidos.
O gráfico abaixo destaca a importância de uma comunicação clara e eficaz durante todo o processo de desenvolvimento para minimizar erros e garantir a entrega de um produto de alta qualidade.



Fonte da imagem: https://blog.onedaytesting.com.br/shift-left-testing/

1.2 – Por que o Shift Left é Importante?

Importância do Shift Left em uma Cultura Ágil

Em uma cultura ágil, onde a entrega rápida e a adaptação contínua são fundamentais, o Shift Left Testing desempenha um papel crucial. Ele permite que as equipes identifiquem e resolvam problemas rapidamente, iterando de forma eficiente e fornecendo feedback contínuo ao longo do ciclo de desenvolvimento. Isso ajuda as empresas a se manterem ágeis, respondendo rapidamente às mudanças nas necessidades do cliente e do mercado.

Um dos objetivos explícitos do manifesto ágil é o foco na entrega de software com qualidade e valor agregado. Para se beneficiar da metodologia ágil, os times devem seguir práticas e princípios durante todo o ciclo de vida do software.
A cultura de qualidade é um desses princípios.

1.2 – Por que o Shift Left é Importante?

Vantagens para o Desenvolvimento e a Qualidade do Produto

Implementar o Shift Left Testing não apenas melhora a eficácia dos testes, mas também traz uma série de benefícios significativos para o desenvolvimento de software e a qualidade do produto:

1. **Identificação Antecipada de Bugs e Facilidade na Resolução de Problemas :** Um dos principais benefícios do Shift Left é a capacidade de identificar bugs mais cedo no ciclo de desenvolvimento. Isso significa que os bugs são descobertos e corrigidos quando ainda estão frescos na mente dos desenvolvedores, tornando a correção mais fácil, rápida e econômica.
2. **Prevenção de Atrasos e Fracassos de Projetos, Antecipação de Melhorias e Geração de Previsibilidade e Agilidade:** Problemas descobertos tardiamente no ciclo de desenvolvimento podem causar atrasos significativos e até mesmo o fracasso de projetos. Ao antecipar a identificação de problemas e melhorias, o Shift Left Testing ajuda a mitigar esse risco e a manter o projeto no caminho certo para o sucesso. Além disso, uma abordagem de teste mais ágil e orientada para o início do ciclo de desenvolvimento gera mais previsibilidade em relação ao progresso do projeto e permite uma resposta mais ágil a mudanças de requisitos ou condições do mercado.
3. **Garantia de Qualidade e Aumento da Confiança no Time e no Negócio:** Ao identificar e corrigir bugs mais cedo, o Shift Left Testing ajuda a garantir que o produto final entregue aos clientes seja de alta qualidade e atenda às suas expectativas. Essa prática também aumenta a confiança no time de desenvolvimento e no negócio como um todo, demonstrando um compromisso com a excelência e a satisfação do cliente.

Capítulo 2

Aplicando o Shift-Left Testing no Dia a Dia

O Shift-Left Testing é mais do que uma simples prática de teste; é uma mudança de mentalidade que afeta diretamente as operações diárias das equipes de desenvolvimento e qualidade. Neste capítulo, vamos explorar como aplicar o Shift-Left Testing no dia a dia, desde a integração de diferentes papéis até a adoção de práticas e técnicas específicas.

2.1. Reunião dos 3 (ou 4) Amigos

A reunião dos 3 amigos é uma prática colaborativa originada do Behavior-Driven Development (BDD) que envolve o analista de negócio, um QA, um desenvolvedor e, muitas vezes, um representante de UX/UI. O objetivo principal desta reunião é garantir que todos os membros da equipe tenham uma compreensão clara e compartilhada dos requisitos e objetivos do projeto.

Durante a reunião dos 3 amigos, cada participante desempenha um papel específico e complementar:

- O analista de negócio traz a perspectiva do cliente e define os requisitos e funcionalidades desejadas.
- O QA contribui com insights sobre potenciais cenários de teste, critérios de aceitação e preocupações relacionadas à qualidade.
- O desenvolvedor fornece feedback sobre a viabilidade técnica das funcionalidades propostas e sugere soluções técnicas.
- O representante de UX/UI garante que as necessidades e expectativas do usuário final sejam consideradas no processo de desenvolvimento.

Juntos, esses amigos colaboram para melhorar o entendimento das histórias ou funcionalidades a serem desenvolvidas. Isso inclui discutir e esclarecer requisitos ambíguos, identificar suposições ocultas e antecipar potenciais desafios técnicos ou de qualidade.

2.1. Reunião dos 3 Amigos

Essa prática proporciona diversos benefícios, incluindo:

1. **Melhor Entendimento das Funcionalidades:** A discussão colaborativa ajuda a esclarecer requisitos e critérios de aceitação, reduzindo assim ambiguidades e mal-entendidos.
2. **Identificação Antecipada de Problemas**: Ao antecipar potenciais desafios técnicos ou de qualidade, a equipe pode tomar medidas proativas para resolvê-los antes que se tornem problemas significativos.
3. **Maior Envolvimento e Comprometimento da Equipe:** A participação ativa de diferentes membros da equipe promove um senso de propriedade compartilhada e comprometimento com o sucesso do projeto.
4. **Entrega de Valor ao Cliente:** Ao garantir que as funcionalidades desenvolvidas atendam às reais necessidades do cliente, a equipe aumenta a probabilidade de entregar um produto de alta qualidade e valor agregado.

Em resumo, a reunião dos 3 amigos é uma prática valiosa que promove a colaboração, o entendimento compartilhado e o alinhamento entre diferentes partes interessadas. Integrá-la ao processo de desenvolvimento pode contribuir significativamente para o sucesso do projeto e a satisfação do cliente.

2.2. Definição de Critérios de Pronto e Feito

- A definição de critérios de Pronto (DoR) e Feito (DoD) desempenha um papel crucial na garantia de que as tarefas sejam iniciadas e concluídas de maneira adequada. O DoR estabelece os requisitos mínimos que uma tarefa deve atender antes de ser considerada pronta para entrar no fluxo de trabalho, enquanto o DoD define os critérios que devem ser cumpridos para que a tarefa seja considerada concluída.

  Definition of Ready (DoR): O DoR é uma lista de critérios que determinam quando uma tarefa está pronta para entrar no fluxo de trabalho. Isso pode incluir requisitos como clareza dos objetivos, disponibilidade de recursos necessários e definição de critérios de aceitação. O DoR garante que as tarefas sejam iniciadas de forma adequada, evitando retrabalho e promovendo eficiência.
- **Definition of Done (DoD):** O DoD é uma lista de critérios que definem quando uma tarefa está concluída. Isso pode incluir requisitos como testes completos, documentação atualizada e revisão por pares. O DoD garante que o trabalho seja considerado finalizado apenas quando atender aos requisitos e padrões de qualidade estabelecidos.

A definição de critérios de Pronto e Feito ao processo de desenvolvimento permite que a equipe mantenha um alto padrão de qualidade desde o início do projeto até a entrega final. Essas práticas promovem a transparência, eficiência e qualidade contínua do código desenvolvido.

2.3. Colaboração e Integração entre QAs e Desenvolvedores

Uma das características essenciais do Shift-Left Testing é a colaboração e integração estreita entre os membros da equipe de desenvolvimento e qualidade. Em vez de trabalharem de forma isolada, QAs e desenvolvedores devem colaborar desde o início do projeto, compartilhando responsabilidades e conhecimentos para garantir a entrega de um produto de alta qualidade.

Vamos explorar mais detalhadamente como essa colaboração pode ser promovida:

- **Mais Colaboração e Menos Isolamento**: Em equipes que adotam o Shift-Left Testing, é fundamental promover uma cultura de colaboração e compartilhamento de responsabilidades. QAs e desenvolvedores devem trabalhar juntos, no mesmo time, desde o início do projeto, em vez de formarem grupos isolados. Essa abordagem permite que os membros da equipe troquem conhecimentos, identifiquem problemas mais cedo e contribuam para a resolução de desafios de forma colaborativa.
- **Uso do Dev-Box Testing:** Uma prática cada vez mais adotada para promover a colaboração entre QAs e desenvolvedores é o uso do Dev-Box Testing. Nessa abordagem, QAs e desenvolvedores revisam e testam o código em um ambiente local de desenvolvimento, compartilhando responsabilidades e conhecimentos durante todo o processo. Isso não apenas facilita a identificação de problemas e a troca de informações, mas também promove uma maior eficiência e agilidade nos testes bem como reduz drasticamente o número de retrabalhos

2.3. Colaboração e Integração entre QAs e Desenvolvedores

A colaboração e integração entre QAs e desenvolvedores promovem diversos benefícios, incluindo:

- **Identificação Antecipada de Problemas:** Ao trabalharem juntos desde o início do projeto, QAs e desenvolvedores podem identificar problemas mais cedo e tomar medidas proativas para resolvê-los. Isso contribui para a redução de bugs e problemas de qualidade ao longo do ciclo de desenvolvimento.
- **Maior Eficiência e Produtividade:** A colaboração entre QAs e desenvolvedores permite que a equipe trabalhe de forma mais eficiente e produtiva, evitando atrasos e retrabalho. Compartilhando responsabilidades e conhecimentos, a equipe pode alcançar seus objetivos de forma mais rápida e eficaz.
- **Maior Qualidade do Produto Final:** Ao integrarem testes e qualidade ao longo de todo o ciclo de desenvolvimento, QAs e desenvolvedores garantem que o produto final atenda aos mais altos padrões de qualidade. Isso resulta em um produto mais confiável, estável e livre de defeitos, aumentando a satisfação do cliente e a reputação da empresa.

Promover a colaboração e integração entre QAs e desenvolvedores é essencial para o sucesso do Shift-Left Testing. Ao trabalharem juntos como uma equipe unificada, QAs e desenvolvedores podem alcançar resultados significativamente melhores do que trabalhando isoladamente.

2.4. Utilização de Cenários de Testes Mapeados Antes do Desenvolvimento

Ao adotar o Shift-Left Testing, é fundamental mapear cenários de testes antes mesmo do início do desenvolvimento. Essa prática permite que a equipe tenha uma visão clara dos fluxos que serão testados e das expectativas de qualidade desde o início do projeto. Vamos explorar mais detalhadamente como essa abordagem pode ser implementada:

- **Definição de Cenários de Testes Antes do Desenvolvimento:** Em vez de esperar até que o código esteja pronto para iniciar os testes, a equipe deve colaborar na definição de cenários de testes antes mesmo do início do desenvolvimento. Isso pode ser feito durante as reuniões de refinamento ou planejamento, onde QAs, desenvolvedores e outros membros da equipe podem contribuir com suas perspectivas e conhecimentos.
- **Benefícios dos Cenários de Testes Mapeados Antes do Desenvolvimento:** Ao mapear cenários de testes no início do projeto, a equipe pode:
  - Identificar requisitos e critérios de aceitação claros desde o início.
  - Prevenir bugs e problemas de qualidade antes mesmo de começarem a desenvolver.
  - Guiar o desenvolvimento, fornecendo uma visão clara dos fluxos que precisam ser implementados e testados.
  - Facilitar a criação de testes automatizados, uma vez que os cenários de testes já estão definidos e documentados.

Ao adotar a prática de mapear cenários de testes antes do desenvolvimento, a equipe pode garantir uma abordagem proativa para a garantia da qualidade, prevenindo problemas e bugs antes mesmo de começarem a desenvolver. Isso contribui para um processo de desenvolvimento mais eficiente, ágil e de alta qualidade.

2.4. Utilização de Cenários de Testes Mapeados Antes do Desenvolvimento

Ao adotar o Shift-Left Testing, é fundamental mapear cenários de testes antes mesmo do início do desenvolvimento. Essa prática permite que a equipe tenha uma visão clara dos fluxos que serão testados e das expectativas de qualidade desde o início do projeto. Vamos explorar mais detalhadamente como essa abordagem pode ser implementada:

- **Definição de Cenários de Testes Antes do Desenvolvimento:** Em vez de esperar até que o código esteja pronto para iniciar os testes, a equipe deve colaborar na definição de cenários de testes antes mesmo do início do desenvolvimento. Isso pode ser feito durante as reuniões de refinamento ou planejamento, onde QAs, desenvolvedores e outros membros da equipe podem contribuir com suas perspectivas e conhecimentos.
- **Benefícios dos Cenários de Testes Mapeados Antes do Desenvolvimento:** Ao mapear cenários de testes no início do projeto, a equipe pode:
  - Identificar requisitos e critérios de aceitação claros desde o início.
  - Prevenir bugs e problemas de qualidade antes mesmo de começarem a desenvolver.
  - Guiar o desenvolvimento, fornecendo uma visão clara dos fluxos que precisam ser implementados e testados.
  - Facilitar a criação de testes automatizados, uma vez que os cenários de testes já estão definidos e documentados.

Ao adotar a prática de mapear cenários de testes antes do desenvolvimento, a equipe pode garantir uma abordagem proativa para a garantia da qualidade, prevenindo problemas e bugs antes mesmo de começarem a desenvolver. Isso contribui para um processo de desenvolvimento mais eficiente, ágil e de alta qualidade.

2.5. Envolvimento do Cliente

No contexto do Shift-Left Testing, o envolvimento do cliente desde as fases iniciais do projeto é crucial para garantir que as necessidades e expectativas do usuário final sejam compreendidas e atendidas de forma eficaz. Vamos explorar como o envolvimento do cliente pode ser integrado ao processo de desenvolvimento:

- **Importância do Envolvimento do Cliente:** O envolvimento do cliente é essencial para garantir que o produto final atenda às expectativas e necessidades do usuário final. Ao colaborar de perto com os clientes desde o início do projeto, a equipe pode obter insights valiosos sobre os requisitos do produto, prioridades de desenvolvimento e expectativas de qualidade.
- **Benefícios do Envolvimento do Cliente:** Ao integrar o cliente ao processo de desenvolvimento, a equipe pode beneficiar-se de vários aspectos:
    - **Melhor Compreensão dos Requisitos:** O envolvimento do cliente permite uma compreensão mais clara dos requisitos e objetivos do projeto, reduzindo o risco de mal-entendidos e retrabalho.
    - **Feedback Contínuo:** Com a participação ativa do cliente, a equipe pode receber feedback contínuo sobre as funcionalidades desenvolvidas, permitindo ajustes e melhorias ao longo do tempo.
    - **Validação Antecipada:** O envolvimento do cliente possibilita a validação antecipada das funcionalidades desenvolvidas, garantindo que o produto atenda às expectativas e necessidades do usuário final desde o início do projeto.

2.5. Envolvimento do Cliente

- **Métodos de Envolvimento do Cliente:** Existem várias práticas eficazes para integrar o cliente ao processo de desenvolvimento, incluindo:
  - **Reuniões de Planejamento e Revisão:** Realizar reuniões de planejamento e revisão com o cliente para discutir requisitos, prioridades e expectativas.
  - **Demonstrações Regulares do Produto:** Realizar demonstrações regulares do produto em desenvolvimento para obter feedback direto do cliente sobre as funcionalidades implementadas.
  - **Sessões de Feedback:** Promover sessões regulares de feedback com o cliente para discutir progresso, desafios e oportunidades de melhoria.

Ao integrar o cliente ao processo de desenvolvimento, a equipe pode garantir que o produto final atenda às expectativas e necessidades do usuário final, promovendo assim o sucesso do projeto e a satisfação do cliente.

2.5. Envolvimento do Cliente

- **Métodos de Envolvimento do Cliente:** Existem várias práticas eficazes para integrar o cliente ao processo de desenvolvimento, incluindo:
  - **Reuniões de Planejamento e Revisão:** Realizar reuniões de planejamento e revisão com o cliente para discutir requisitos, prioridades e expectativas.
  - **Demonstrações Regulares do Produto:** Realizar demonstrações regulares do produto em desenvolvimento para obter feedback direto do cliente sobre as funcionalidades implementadas.
  - **Sessões de Feedback:** Promover sessões regulares de feedback com o cliente para discutir progresso, desafios e oportunidades de melhoria.

Ao integrar o cliente ao processo de desenvolvimento, a equipe pode garantir que o produto final atenda às expectativas e necessidades do usuário final, promovendo assim o sucesso do projeto e a satisfação do cliente.

2.6. Feedback Rápido e Frequente

No contexto do Shift-Left Testing, a obtenção de feedback durante o processo de desenvolvimento é crucial para identificar problemas, mitigar riscos e garantir a qualidade do produto final. Nesta seção, exploraremos práticas e ferramentas que promovem o feedback contínuo e seus benefícios:

- **Reuniões Diárias (Dailies):** As reuniões diárias, também conhecidas como Dailies ou Daily Stand-ups, são uma prática ágil que permite que a equipe compartilhe atualizações de progresso, identifique bloqueios e discuta soluções em um ambiente colaborativo. Essas reuniões fornecem feedback instantâneo sobre o andamento do projeto e promovem a comunicação eficaz entre os membros da equipe.
    - **Participação Proativa:** O QA deve participar das reuniões diárias para garantir que questões de qualidade sejam discutidas desde o início do projeto. Isso inclui compartilhar preocupações sobre a integridade do código, possíveis cenários de teste e riscos de qualidade.
    - **Foco na Prevenção de Problemas:** Durante as reuniões diárias, o QA pode destacar áreas do código ou funcionalidades que podem exigir testes adicionais ou revisões de qualidade. Ao antecipar possíveis problemas, a equipe pode tomar medidas proativas para corrigir ou mitigar riscos antes que afetem a entrega do produto.

2.6. Feedback Rápido e Frequente

- **Integração Contínua/Entrega Contínua (CI/CD):** A adoção de práticas de CI/CD automatiza o processo de integração, teste e entrega de código, permitindo que a equipe receba feedback rápido sobre a qualidade e estabilidade das alterações realizadas. Isso facilita a identificação precoce de problemas e garante uma entrega mais rápida e confiável do produto final.
    - **Automação de Testes:** O QA desempenha um papel central na automação de testes para suportar práticas de CI/CD. Isso envolve o desenvolvimento e execução de testes automatizados que validam continuamente as alterações de código, garantindo que novas funcionalidades não causem regressões ou quebras no sistema.
    - **Validação Contínua da Qualidade:** Os testes automatizados integrados ao pipeline de CI/CD permitem uma validação contínua da qualidade do código em cada etapa do desenvolvimento. O QA pode monitorar esses testes e garantir que eles cubram adequadamente os requisitos de qualidade, identificando e corrigindo falhas de teste conforme necessário.
- **Identificação de Riscos desde o Início do Desenvolvimento:** A equipe deve estar constantemente atenta à identificação e mitigação de riscos durante o processo de desenvolvimento. Ao antecipar e abordar proativamente os riscos, a equipe pode evitar surpresas indesejadas e garantir a qualidade e a estabilidade do produto final.
    - **Análise Proativa de Riscos:** O QA deve estar envolvido desde o início do projeto na identificação e análise de riscos relacionados à qualidade do software. Isso inclui avaliar os requisitos, arquitetura e tecnologias utilizadas para identificar possíveis áreas de preocupação e definir estratégias de mitigação de riscos.

Code Review: A prática de code review envolve a revisão do código por pares para identificar problemas de qualidade, padrões de codificação inconsistentes e oportunidades de melhoria. Essa prática

2.6. Feedback Rápido e Frequente

- **Code Review:** A prática de code review envolve a revisão do código por pares para identificar problemas de qualidade, padrões de codificação inconsistentes e oportunidades de melhoria. Essa prática promove o aprendizado colaborativo, garante a consistência do código e ajuda a evitar bugs e vulnerabilidades de segurança.
    - **Revisão de Qualidade do Código:** Durante as revisões de código, o QA pode focar na qualidade e integridade do código em relação aos requisitos de qualidade definidos. Isso envolve identificar padrões de codificação inconsistentes, potenciais vulnerabilidades de segurança e áreas que requerem testes adicionais ou revisões de qualidade.
    - **Garantia de Conformidade com Padrões:** O QA desempenha um papel importante na garantia de que o código revisado esteja em conformidade com os padrões de qualidade e melhores práticas estabelecidos pela equipe. Isso ajuda a manter a consistência, legibilidade e manutenibilidade do código ao longo do tempo.

- **Utilização de Ferramentas como Linters e Análise Estática de Código**: O uso de ferramentas automatizadas, como linters e análise estática de código, permite que a equipe identifique problemas de código, padrões de codificação inconsistentes e possíveis vulnerabilidades de segurança de forma rápida e eficiente. Essas ferramentas fornecem feedback instantâneo sobre a qualidade do código e ajudam a manter um alto padrão de qualidade ao longo do processo de desenvolvimento.

Ao integrar práticas e ferramentas que promovem o feedback contínuo durante o processo de desenvolvimento, a equipe pode garantir a qualidade, estabilidade e segurança do produto final, além de promover uma cultura de melhoria contínua e aprendizado colaborativo.

2.6. Feedback Rápido e Frequente

- **Utilização de Ferramentas como Linters e Análise Estática de Código**: O uso de ferramentas automatizadas, como linters e análise estática de código, permite que a equipe identifique problemas de código, padrões de codificação inconsistentes e possíveis vulnerabilidades de segurança de forma rápida e eficiente. Essas ferramentas fornecem feedback instantâneo sobre a qualidade do código e ajudam a manter um alto padrão de qualidade ao longo do processo de desenvolvimento.
    - **Configuração e Monitoramento de Ferramentas de Qualidade:** O QA é responsável por configurar e monitorar ferramentas como linters e análise estática de código para identificar problemas de qualidade de forma automática e contínua. Isso permite uma abordagem proativa para identificar e corrigir problemas de código antes que afetem a qualidade do produto final.
    - **Ação Pronta sobre Resultados:** Ao identificar problemas de qualidade por meio das ferramentas de análise estática, o QA pode tomar medidas imediatas para corrigir ou mitigar os problemas. Isso pode incluir a revisão do código, a implementação de correções e a coordenação com a equipe de desenvolvimento para garantir que os problemas sejam resolvidos de forma eficaz e oportuna.

Integrando práticas e ferramentas que promovem feedback constante, a equipe assegura a qualidade e estabilidade do produto final. O QA desempenha um papel-chave na identificação precoce de problemas e na melhoria contínua do processo. Ao adotar o Shift-Left, o QA prioriza a qualidade desde o início do projeto, garantindo produtos finais mais confiáveis e satisfatórios para os usuários.

2.7. Cultura de Qualidade em Toda a Equipe

Neste subcapítulo, vamos destacar como a mentalidade de qualidade permeia toda a equipe e como cada indivíduo se torna um representante ativo da qualidade do produto.

- **Responsabilidade Compartilhada:** No contexto do Shift-Left Testing, a qualidade não é apenas responsabilidade do QA, mas de todos os membros da equipe. Cada desenvolvedor, analista de negócios, designer e gerente de projeto compartilha a responsabilidade de garantir que o produto atenda aos mais altos padrões de qualidade.
- **Mentalidade de Quem Testa:** Em uma cultura de qualidade, todos os membros da equipe devem adotar uma mentalidade de quem testa. Isso significa ser proativo na identificação e resolução de problemas, mesmo antes que eles se manifestem como bugs. Todos devem estar constantemente questionando e validando o trabalho realizado, buscando garantir que atenda às expectativas e requisitos do usuário final.
- Análise de Riscos: Além de testar o produto, cada membro da equipe deve ser capaz de analisar riscos potenciais em seu trabalho e no projeto como um todo. Isso envolve antecipar possíveis problemas e tomar medidas preventivas para evitá-los ou mitigá-los. Ao identificar e abordar proativamente os riscos, a equipe pode evitar a ocorrência de problemas no futuro.
- **Busca Constante pela Qualidade:** Uma cultura de qualidade também é caracterizada pela busca constante pela excelência. Isso significa que cada indivíduo na equipe está comprometido em melhorar continuamente seus processos, práticas e produtos. A qualidade não é vista como um destino final, mas sim como um caminho contínuo de aprimoramento e aprendizado.

Conclusão:

Ao adotar uma cultura de qualidade em toda a equipe, estamos alinhados aos princípios do manifesto ágil, que valoriza indivíduos e interações mais do que processos e ferramentas. Reconhecemos que a qualidade não pode ser terceirizada para o departamento de

2.7. Cultura de Qualidade em Toda a Equipe

Conclusão:

Ao adotar uma cultura de qualidade em toda a equipe, estamos alinhados aos princípios do manifesto ágil, que valoriza indivíduos e interações mais do que processos e ferramentas. Reconhecemos que a qualidade não pode ser terceirizada para o departamento de QA, mas deve ser incorporada em cada etapa do desenvolvimento de software. Ao desenvolver uma mentalidade de quem testa, analisa riscos e busca constantemente a qualidade, cada membro da equipe contribui para o sucesso do projeto e a satisfação do cliente.

Capítulo 3

Artefatos e Práticas para Aplicação do Shift-Left Testing

3.1. Kick Off:

O Kick Off, na tradução literal, seria 'ponta-pé inicial'.
Ele acontece quando as pessoas desenvolvedoras estão prontas para começar um novo card (história de usuário, tarefa ou débito técnico).
Trata-se de uma cerimônia rápida - entre 5 e 15 minutos - com o objetivo de entender o que deve ser feito.
É uma excelente oportunidade para realizar verificações e validações acerca do que se pretende entregar.

Abaixo, 10 perguntas para guiar um Kick Off eficaz:

1. O refinamento do card foi feito?
2. Está completo com detalhes e todas as informações relevantes?
3. O valor de negócio está claro?
4. Há dependência com outros cards?
5. Existe alguma dívida técnica relacionada a ele?
6. Há algum protótipo?
7. Há mensagens de erro previstas?
8. O escopo está num bom tamanho ou é melhor quebrar em mais cards?
9. O "caminho feliz" e os fluxos alternativos estão claros?
10. Como pretende testar?

Recomendado para: Equipes multidisciplinares que estão iniciando um novo projeto ou uma nova fase do projeto, onde é essencial alinhar objetivos, esclarecer expectativas e definir critérios de sucesso desde o início.

Não recomendado para: Projetos muito pequenos ou equipes já consolidadas que estão apenas fazendo pequenas alterações, onde o tempo e os recursos necessários para conduzir um kick-off podem não ser justificados.

3.1. Kick Off:

Prós:

Alinhamento de Objetivos: O kick-off proporciona uma oportunidade para todas as partes interessadas se reunirem e alinharem seus objetivos e expectativas em relação ao projeto. Isso ajuda a garantir que todos estejam na mesma página desde o início e trabalhem em direção aos mesmos objetivos.

Esclarecimento de Expectativas: Durante o kick-off, as expectativas em relação ao projeto, como escopo, prazos e recursos disponíveis, podem ser discutidas e esclarecidas. Isso ajuda a evitar mal-entendidos e conflitos no futuro.

Definição de Critérios de Sucesso: O kick-off é uma oportunidade para definir claramente os critérios de sucesso do projeto, ou seja, quais são os resultados esperados e como serão medidos. Isso fornece uma estrutura para avaliar o progresso e o desempenho ao longo do tempo.

Contras:

Consumo de Tempo e Recursos: Realizar um kick-off pode consumir tempo e recursos significativos, especialmente se houver muitas partes interessadas envolvidas. Isso pode ser considerado desnecessário em projetos menores ou em equipes já consolidadas.

Possível Sobrecarga de Informações: Se não for conduzido de forma eficiente, um kick-off pode resultar em uma sobrecarga de informações, tornando difícil para os participantes absorver e processar tudo o que foi discutido. Isso pode levar à falta de clareza e compreensão em relação aos objetivos do projeto.

3.2. Teste Contínuo:

O teste contínuo é um tipo de teste de software no qual o produto é avaliado antes, durante e depois de todo o processo de Entrega Contínua.
Com isso os desenvolvedores realizam os testes de codificação e unitários e os QA's ficam responsáveis pelos testes de integração, de sistemas e aceitação em todo o processo.
O objetivo desse tipo de teste é dar um feedback rápido e contínuo de todo o fluxo desenvolvido na tarefa.

Recomendado para: Todos os tipos de equipes de desenvolvimento, especialmente aquelas que adotam metodologias ágeis, onde a integração de testes ao ciclo de desenvolvimento é fundamental para garantir a qualidade do produto final.

Não recomendado para: Equipes com infraestrutura de testes inadequada ou que não valorizam a automação de testes, pois a implementação de testes contínuos pode exigir investimento em ferramentas e recursos de automação.

3.3. Testes Regressivos:

Também é um tipo de teste de software no qual o produto é avaliado após melhorias e alterações realizadas no código (correções) durante o desenvolvimento, cujo objetivo é detectar efeitos colaterais indesejados, garantindo que nenhum defeito foi acrescentado no sistema após modificações e trazer segurança quanto a publicação e/ou entrega final.

Recomendado para: Equipes que realizam alterações frequentes no código ou que têm uma base de código grande e complexa, onde a garantia de que as alterações não introduzem regressões é crucial para manter a estabilidade do produto.

Não recomendado para: Projetos com alterações mínimas no código ou onde a manutenção de testes regressivos seria excessivamente dispendiosa, especialmente se os riscos de regressão são baixos.

Prós:

Garantia de Estabilidade: Os testes regressivos garantem que as alterações no código não introduzam regressões, mantendo a estabilidade do produto.
Redução de Riscos: Ao identificar rapidamente possíveis regressões, os testes regressivos ajudam a reduzir os riscos associados a alterações no código.
Manutenção da Qualidade: A execução regular de testes regressivos ajuda a manter a qualidade do produto ao longo do tempo, mesmo após várias iterações de desenvolvimento.

3.3. Testes Regressivos:

Contras:

Custo de Manutenção: Manter uma suíte abrangente de testes regressivos pode exigir recursos significativos em termos de desenvolvimento, execução e manutenção.
Complexidade da Configuração: Configurar e manter testes regressivos pode ser complexo, especialmente em projetos com uma base de código grande e complexa.
Tempo de Execução: A execução de testes regressivos pode levar tempo, especialmente se a suíte de testes for extensa, o que pode afetar o fluxo de desenvolvimento.

3.4. Testes Baseados em Riscos:

Testes baseados em riscos é uma abordagem importante para prever e mitigar inconsistências na aplicação. Uma de suas vantagens é que as features mais críticas são desenvolvidas e testadas primeiro, de modo a reduzir o risco geral no projeto.

Recomendado para: Projetos com requisitos complexos ou onde a mitigação de certos riscos é crucial para o sucesso do projeto.

Não recomendado para: Projetos simples ou onde os riscos são bem compreendidos e facilmente gerenciados.

Prós:
Priorização de Testes: Os testes baseados em riscos permitem priorizar os testes com base na probabilidade e impacto dos riscos identificados, otimizando o uso de recursos de teste.
Foco nos Pontos Críticos: Identifica e concentra esforços nos pontos críticos do sistema ou funcionalidades que têm maior potencial de impacto negativo no produto final.
Antecipação de Problemas: Ajuda a antecipar problemas potenciais e a tomar medidas proativas para mitigar riscos, reduzindo assim a possibilidade de problemas graves no futuro.

Contras:
Requer Habilidades Específicas: Identificar e avaliar riscos requer habilidades específicas e conhecimento detalhado do domínio do projeto, o que nem sempre está disponível na equipe.
Possível Sobrecarga de Testes: Se não for feito de forma eficaz, o processo de identificação e priorização de riscos pode resultar em uma sobrecarga de testes, levando a uma alocação inadequada de recursos e esforços.

3.5. Automação de Testes:

A automação reduz os erros humanos, aumenta a confiança na entrega e permite que sobre mais tempo para as pessoas focarem em tarefas mais desafiadoras/inspiradoras do que passar dias fazendo testes exaustivos!.

Recomendado para: Equipes que buscam aumentar a eficiência dos testes, reduzir o tempo de execução e garantir a consistência nos resultados dos testes.

Não recomendado para: Projetos onde a automação de testes não é viável devido à sua natureza ou recursos limitados.

Prós:
Eficiência e Rapidez: Os testes automatizados são executados de forma rápida e eficiente, permitindo testar o software em várias plataformas e configurações de forma mais ágil.
Repetibilidade: Os testes automatizados garantem a repetibilidade dos testes, fornecendo resultados consistentes em todas as execuções.
Liberação de Recursos Humanos: A automação de testes libera recursos humanos para atividades mais criativas e estratégicas, ao mesmo tempo que reduz o tempo gasto em testes manuais repetitivos.

Contras:
Custo Inicial Elevado: O desenvolvimento e implementação de testes automatizados podem exigir um investimento inicial significativo em termos de tempo e recursos.
Manutenção Contínua: Os testes automatizados requerem manutenção contínua para acompanhar as mudanças no software, o que pode aumentar os custos a longo prazo.
Limitações da Automação: Nem todos os tipos de testes podem ser completamente automatizados, e há casos em que os testes manuais são necessários para garantir uma cobertura adequada.

3.6. Desk Check:

O Desk Check é uma cerimônia rápida (entre 5 a 10 minutos) que ocorre logo após o desenvolvimento de um card.

As pessoas desenvolvedoras que trabalharam no card se reúnem com as pessoas analistas de negócios, designers, e QAs com o objetivo de:

- Verificar o que foi desenvolvido;
- Realizar testes rápidos e verificações para confirmar que os critérios de aceitação foram cumpridos;
- Encontrar defeitos.

Abaixo, 10 perguntas para guiar um Desk Check:

- A nova feature faz o que é esperado?
- Há algum comportamento não previsto?
- Cobre cenários alternativos e o "caminho feliz"?
- Qual a cobertura de testes para esta feature?
- Passou pelo code review?
- Foi verificado se a implementação pode ter impactado outra parte do sistema?
- Todos os critérios de aceitação estão cobertos?
- É necessário algum tipo de melhoria ou correção?
- A implementação está conforme o design da interface?
- Foram verificados se havia erros gramaticais nos textos?

3.6. Desk Check:

Estas perguntas ajudam a garantir que se desenvolveu a coisa certa do jeito certo.

O Desk Check deve ser conduzido no ambiente de testes (não confunda com ambiente local) e, se forem encontrados defeitos, eles não precisam ser registrados formalmente.

O card volta para o desenvolvimento para serem feitos ajustes, ou se o valor de negócio do card tiver sido atingido, pode-se abrir débitos técnicos e dar o card como concluído.

Recomendado para: Equipes que buscam garantir a qualidade do código e a conformidade com os critérios de aceitação logo após o desenvolvimento de um card, de forma rápida e eficiente.

Não recomendado para: Projetos onde o tempo e os recursos não permitem a realização de verificações rápidas após o desenvolvimento de cada card.

3.6. Desk Check:

Prós:

Verificação Imediata do Desenvolvimento: O Desk Check permite verificar imediatamente o que foi desenvolvido, garantindo que atenda aos requisitos e expectativas do usuário.
Identificação Precoce de Defeitos: A realização de testes rápidos e verificações durante o Desk Check ajuda a identificar defeitos logo após o desenvolvimento, permitindo correções rápidas e reduzindo o risco de problemas futuros.
Colaboração entre as Equipes: A participação de diferentes membros da equipe, como desenvolvedores, analistas de negócios, designers e QAs, promove a colaboração e a troca de conhecimentos, garantindo uma visão holística do trabalho realizado.

Contras:

Possível Sobrecarga de Tempo: Se não for conduzido de forma eficiente, o Desk Check pode se tornar uma sobrecarga de tempo para a equipe, especialmente em projetos com muitos cards sendo desenvolvidos simultaneamente.
Limitação na Cobertura de Testes: O Desk Check é uma verificação rápida e pode não cobrir todos os aspectos do desenvolvimento, especialmente se houver uma grande quantidade de cards a serem verificados.
Dependência da Cultura da Equipe: O sucesso do Desk Check depende da cultura da equipe e da disposição dos membros em participar ativamente do processo. Se a equipe não valoriza a qualidade e a colaboração, o Desk Check pode não ser eficaz.

Capítulo 4

O Papel do QA para o Sucesso do Shift-Left Testing

No contexto do Shift-Left Testing, o papel do QA é fundamental para garantir a qualidade do produto em todas as etapas do ciclo de desenvolvimento. Neste capítulo, exploraremos a atuação efetiva do QA e estratégias para lidar com resistências por parte da equipe em relação à implementação dos pontos do Shift-Left Testing.

4.1. Atuação do QA no Shift-Left Testing:

- **Integração com a Equipe:** O QA deve integrar-se totalmente à equipe de desenvolvimento desde o início do projeto, participando ativamente de todas as fases do ciclo de vida do software.
- **Participação nas Reuniões de Planejamento:** O QA deve participar das reuniões de planejamento para entender os requisitos do cliente e colaborar na definição dos critérios de aceitação e cenários de teste.
- **Cobrança de Antecipação das Histórias:** O QA deve cobrar a antecipação das histórias antes do refinamento, garantindo que os requisitos sejam discutidos e definidos de forma clara e completa antes do início do desenvolvimento bem como a criação antecipada dos cenários de testes.
- **Participação Ativa em Todas as Cerimônias:** O QA deve participar ativamente de todas as cerimônias, fazendo perguntas pertinentes e trazendo o pessimismo profissional para evitar que defeitos ocorram durante o desenvolvimento.
- **Análise de Histórias, Requisitos e Layouts:** O QA é responsável por analisar cuidadosamente as histórias, requisitos e layouts do projeto, levantando dúvidas sobre regras de negócio e impactos nos sistemas para garantir uma melhor tomada de decisão técnica.
- **Cobrança do Teste Contínuo:** O QA deve cobrar o teste contínuo, solicitando a participação ativa na quebra de tarefas e na execução de testes em todas as etapas do processo de desenvolvimento.

4.1. Atuação do QA no Shift-Left Testing:

- **Prática do DevBox Testing:** O QA deve chamar os desenvolvedores para praticar o DevBox Testing, repassando os principais cenários e identificando defeitos de forma ágil para correção imediata.
- **Feedback Contínuo e Melhoria:** O QA deve fornecer feedback contínuo à equipe de desenvolvimento, identificando áreas de melhoria e buscando constantemente maneiras de otimizar o processo de desenvolvimento e garantir a qualidade do produto final.

4.2. Lidando com Resistências:

- **Educação e Treinamento:** É importante educar a equipe sobre os benefícios do Shift-Left Testing e fornecer treinamento adequado sobre as práticas e ferramentas envolvidas.
- **Demonstração de Valor:** O QA pode demonstrar o valor do Shift-Left Testing através de exemplos concretos de como a abordagem pode ajudar a identificar e evitar problemas no desenvolvimento do software.
- **Envolvimento da Liderança:** A liderança da equipe deve apoiar ativamente a implementação do Shift-Left Testing e ajudar a superar quaisquer resistências ou objeções por parte da equipe.
- **Adaptação Gradual:** Em vez de tentar implementar todas as práticas do Shift-Left Testing de uma só vez, é melhor introduzi-las gradualmente, permitindo que a equipe se acostume e se familiarize com as novas abordagens.
- **Feedback e Melhoria Contínua:** O QA deve incentivar a equipe a fornecer feedback sobre o processo de implementação do Shift-Left Testing e buscar constantemente maneiras de melhorar e otimizar as práticas adotadas.

Capítulo 5

Conclusão

Conclusão

O Shift-Left Testing representa uma mudança fundamental na abordagem de garantia de qualidade de software, deslocando o foco dos testes para as fases iniciais do ciclo de desenvolvimento. Ao longo deste e-book, exploramos os conceitos-chave do Shift-Left Testing, suas práticas, benefícios e desafios, além do papel fundamental do QA na sua implementação bem-sucedida.

O objetivo principal do Shift-Left Testing é identificar e corrigir defeitos o mais cedo possível no processo de desenvolvimento, reduzindo custos, aumentando a eficiência e melhorando a qualidade do produto final. Ao integrar práticas como reuniões de refinamento, teste contínuo, participação ativa do QA em todas as etapas do ciclo de desenvolvimento e práticas de automação, as equipes podem garantir uma abordagem mais eficaz e ágil para o desenvolvimento de software.

No entanto, implementar o Shift-Left Testing não é uma tarefa trivial e pode encontrar resistência dentro da equipe. É importante educar, treinar e envolver toda a equipe no processo, demonstrando os benefícios e promovendo uma cultura de qualidade e colaboração.

Ao finalizar este e-book, esperamos que você tenha adquirido uma compreensão mais profunda do Shift-Left Testing e esteja motivado a implementar suas práticas em seu próprio ambiente de trabalho. Lembre-se sempre de que a busca pela qualidade é uma jornada contínua e que o Shift-Left Testing é uma ferramenta poderosa para ajudá-lo a alcançar seus objetivos de desenvolvimento de software de forma mais eficiente e eficaz.

Este e-book foi baseado na minha palestra *"Shift-Left Testing - Por que testar cedo e frequentemente é tão importante?"* e foi gerado com o auxílio de inteligência artificial. Você pode encontrar o passo a passo desse processo no meu perfil do Github

É importante salientar que, apesar de ter havido uma revisão humana minuciosa do conteúdo, ainda podem existir erros gerados pela inteligência artificial.

Sobre a autora:

Sou uma pessoa apaixonada por tecnologia que decidiu fazer uma transição de carreira aos 37 anos e com três filhos. Em setembro de 2021, tive minha primeira oportunidade como QA e me apaixonei pela área. Desde então, tenho me dedicado intensamente, buscando aprimorar minhas habilidades e fortalecer a comunidade. Sou fundadora do 'Cantinho das Mulheres QAs' uma comunidade acolhedora e segura, comprometida em fortalecer principalmente as mulheres em transição de carreira para a área de QA. também sou uma entusiasta de Agilidade e de Inteligência artificial.

http://bit.ly/portifolio-Vivian