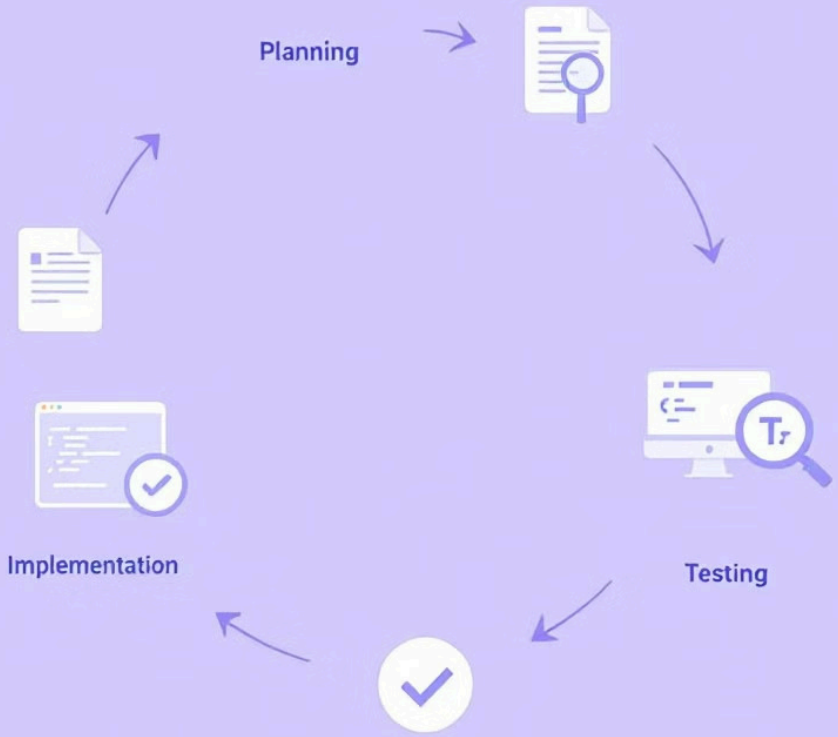


Testes no Ciclo de Vida de Desenvolvimento de Software

Abordaremos desde os fundamentos do SDLC até conceitos avançados como DevOps, testes de confirmação e regressão, e a importância da manutenção contínua.

O SDLC e seu Impacto nos Testes

Planejamento

Definição de requisitos e escopo

Design

Arquitetura e especificações

Codificação

Implementação das funcionalidades

Testes

Verificação da qualidade

Entrega

Implantação e manutenção

O SDLC funciona como um guia com etapas para criar um software do zero, mostrando o que precisa ser feito e em que ordem. Este ciclo impacta diretamente o planejamento e a execução dos testes, determinando como e com que intensidade serão realizados, os prazos, os custos e até mesmo a colaboração entre os membros da equipe.

Boas Práticas de Teste no SDLC

Mesmo com diferentes modelos de SDLC, algumas boas práticas de teste são fundamentais para garantir a qualidade do software. A integração contínua dos testes ao longo do ciclo de desenvolvimento permite identificar problemas mais cedo, reduzindo custos e melhorando a qualidade final do produto.

Correspondência de Atividades

Para cada atividade de desenvolvimento, deve haver uma atividade de teste correspondente, garantindo cobertura e controle de qualidade em todas as etapas.

Níveis Distintos

Diferentes níveis de teste devem ser aplicados com objetivos distintos, como testes unitários, de integração e sistema, evitando redundância e aumentando a eficiência.

Participação Precoce

Os testadores devem participar desde os primeiros rascunhos dos documentos, contribuindo para a identificação precoce de falhas.

Testes como Motivadores do Desenvolvimento

TDD (Test-Driven Development)

Escrever testes antes do código, guiando o desenvolvimento

ATDD (Acceptance Test-Driven Development)

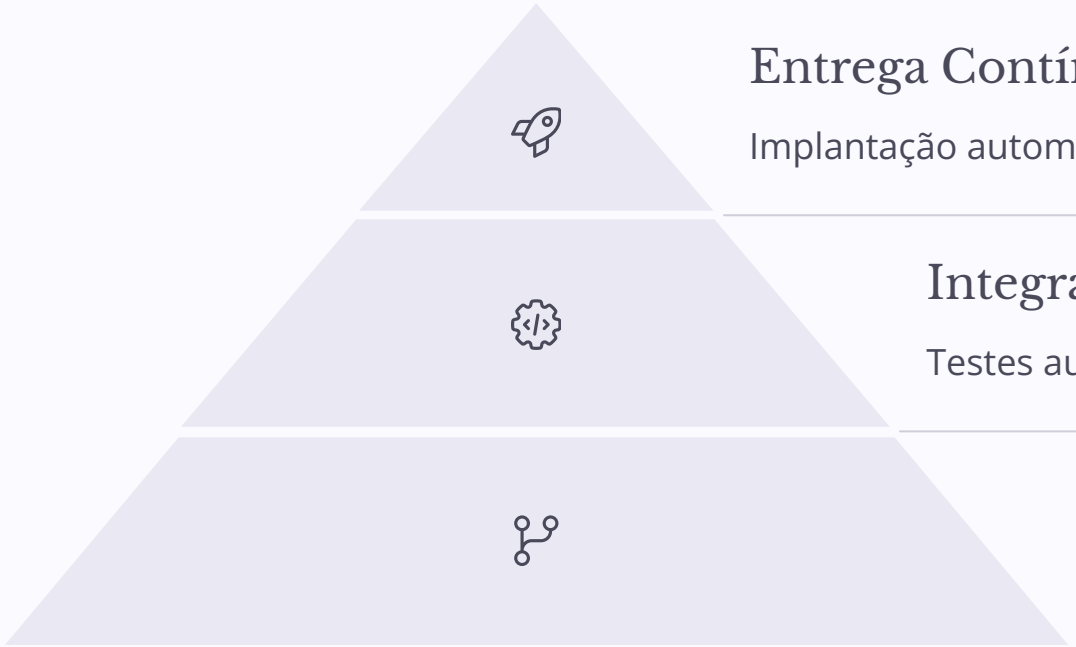
Foco nos critérios de aceitação do cliente

BDD (Behavior-Driven Development)

Colaboração entre negócios e tecnologia com linguagem comum

Usar os testes para guiar o desenvolvimento significa escrever código mais claro, seguro e focado nos requisitos desde o início. Estas abordagens se aplicam nas fases iniciais do desenvolvimento, logo após o levantamento de requisitos e durante o planejamento e codificação, seguindo o princípio do teste antecipado e da estratégia shift-left.

DevOps e a Abordagem Shift-Left



Entrega Contínua (CD)

Implantação automatizada

Integração Contínua (CI)

Testes automatizados frequentes

DevOps

Junção de desenvolvimento e operações

DevOps é uma abordagem organizacional que promove a junção de desenvolvimento e operações para alcançar objetivos comuns. A automatização dos testes com CI (Implementação Contínua) e CD (Entrega Contínua) é fundamental neste contexto.

A abordagem Shift-Left sugere que os testes devem ser feitos desde o início do SDLC - quanto antes melhor. Isso não significa que os testes posteriores devem ser negligenciados, mas que a qualidade deve ser construída desde o começo.

Níveis de Teste

Teste de Componente (Unidade)

Testa partes isoladas do código

Teste de Integração

Valida a comunicação entre componentes

Teste de Sistema

Verifica o sistema completo (end-to-end)

Teste de Aceite

Valida o sistema com foco no uso real

Os níveis de teste são conjuntos organizados de atividades, cada um com foco em diferentes estágios do desenvolvimento. O teste de componente é rápido, automatizado e feito por desenvolvedores. O teste de integração é realizado por devs ou testadores. O teste de sistema é executado por testadores, enquanto o teste de aceite é realizado por usuários, clientes e stakeholders.

Tipos de Teste

Testes Funcionais

Verificam **o que** o sistema faz (funções, regras de negócio). Focam no comportamento esperado do software de acordo com os requisitos.

Exemplos: testes de funcionalidades específicas, fluxos de trabalho, regras de negócio.

Testes Não Funcionais

Avaliam **como** o sistema se comporta (desempenho, segurança, usabilidade). Focam nas características de qualidade além das funcionalidades.

Exemplos: testes de carga, segurança, acessibilidade, compatibilidade.

Abordagens

Caixa Preta: Foca nas entradas e saídas com base nas especificações. Não considera o código.

Caixa Branca: Baseado na estrutura interna (código, lógica). Requer conhecimento técnico.

Testes de Confirmação

Situação comum:

1	2	3	4
Encontro um defeito	*Reporto o problema*	*O time dev corrige*	*É gerado uma nova versão*

“ O que faço depois? “

Reexecuto os testes!

Etapas:

- Após encontrar e corrigir um defeito
- Reexecutar o teste que falhou
- Validar se o erro foi mesmo resolvido
- Verificar se não surgiram efeitos colaterais

Reteste focado no erro que foi reportado.

Testes de Regressão e Manutenção

Teste de Regressão

- Garante que funcionalidades antigas continuam funcionando
- Roda após correções ou novas features
- Ideal para automação, por serem repetitivos
  Toda mudança pode quebrar algo que já funcionava.

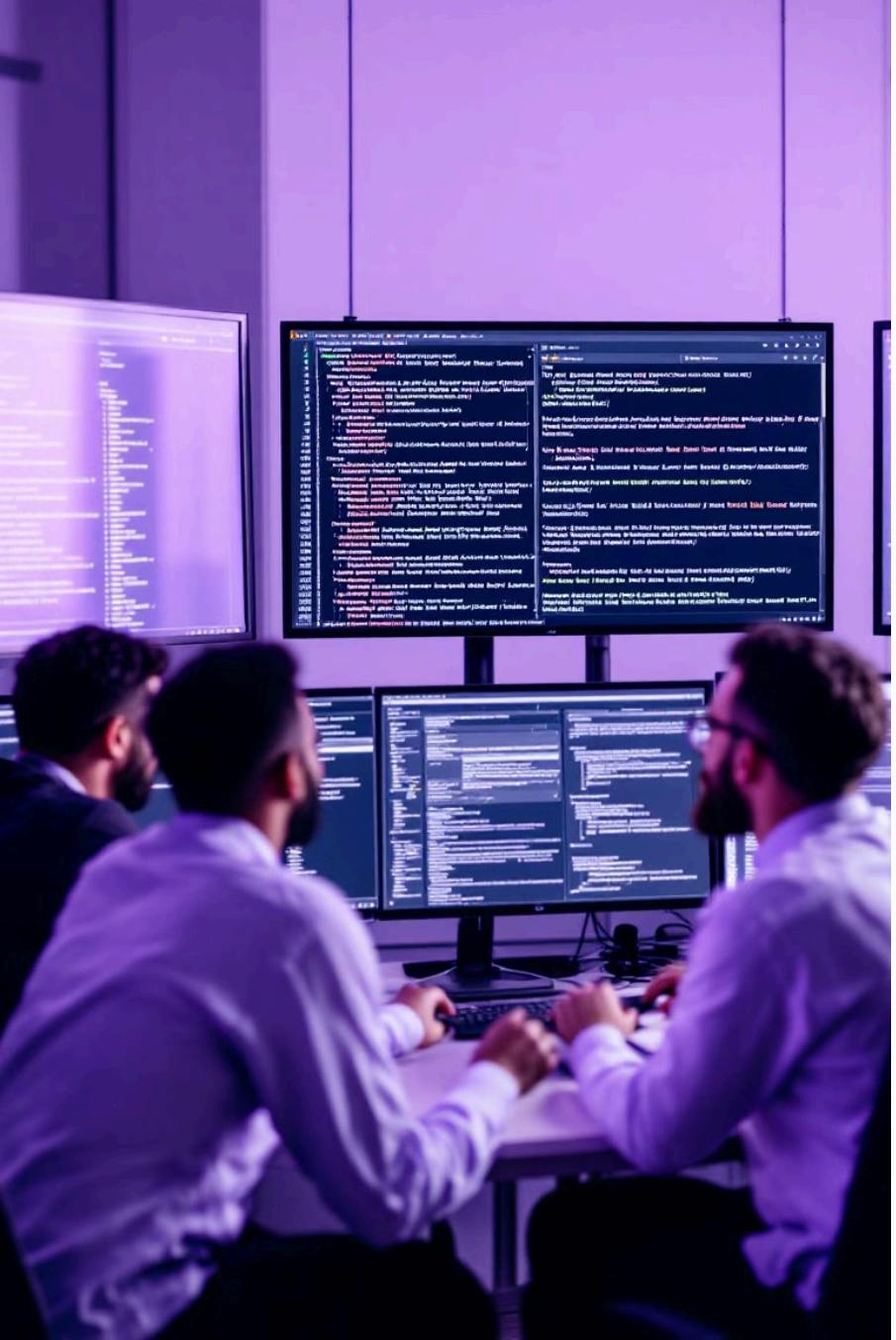
Teste de Manutenção

Feito com o sistema já em executação

Tipos:

- **Corretiva**: corrigir defeitos pós-produção
- **Adaptativa**: ajustes por mudanças no ambiente
- **Perfectiva**: melhorias de desempenho ou manutenção

É importante sempre fazer análise de impacto antes de alterar algo



De modo resumido...

Confirmação

Garante que o erro foi realmente corrigido

Regressão

Garante que o que já funcionava não foi quebrado

Manutenção

Ajudam a manter o sistema funcionando bem, mesmo com o tempo e as mudanças