

# Teste como um Mestre

## A arte e a Ciência do Analista de Testes



Testing  
Software

MARIANA GOTTARDI

01

# A IMPORTÂNCIA DOS TESTES DE SOFTWARE

# A importância dos testes de software

## Benefícios de Testar um Software Antes da Produção

Realizar testes de software é essencial para garantir a qualidade e o sucesso de um sistema. Abaixo, destacamos os principais benefícios dessa prática:

### **1. Garantia da Qualidade**

Os testes asseguram que o software atende aos requisitos, evitando problemas que comprometam a funcionalidade ou a experiência do usuário.

### **2. Redução de Custos**

Corrigir erros nas fases iniciais é mais barato do que solucionar problemas em produção, prevenindo gastos com suporte e reparos emergenciais.

### **3. Confiabilidade e Estabilidade**

Testar o software antes do lançamento reduz falhas críticas, oferecendo aos usuários um sistema confiável e robusto.

### **4. Experiência do Usuário Melhorada**

Erros e interfaces confusas impactam a usabilidade. Testes identificam e eliminam esses problemas, aumentando a satisfação do cliente.

### **5. Segurança Reforçada**

Testes identificam vulnerabilidades, protegendo o sistema contra ataques e garantindo a integridade dos dados.

### **6. Cumprimento de Prazos**

Com menos erros para corrigir, os cronogramas são mais previsíveis, evitando atrasos e retrabalho.

### **7. Preservação da Reputação**

Um lançamento problemático pode prejudicar a imagem da empresa. Testes garantem um produto estável e confiável, fortalecendo a confiança do mercado.

02

# TIPOS DE TESTES DE SOFTWARE

# Tipos de testes de software

Abaixo, listamos os principais tipos de testes e suas finalidades:

## **1. Teste Funcional**

Foco em verificar se o software atende aos requisitos definidos. Testa funcionalidades específicas, simulando o comportamento do usuário.

## **2. Teste de Regressão**

Avalia se novas mudanças no código não afetam funcionalidades existentes. É essencial após atualizações ou correções de bugs.

## **3. Teste de Unidade**

Valida componentes individuais do software, como funções ou métodos, garantindo que cada parte funcione isoladamente.

## **4. Teste de Integração**

Verifica como diferentes módulos ou sistemas interagem entre si. Garante que a comunicação entre as partes do software funcione corretamente.

## **5. Teste de Aceitação**

Realizado pelos clientes ou usuários finais, valida se o software atende às expectativas e está pronto para ser lançado.

## **6. Teste de Desempenho**

Avalia a capacidade do sistema sob diferentes condições de carga. Inclui testes de carga (múltiplos usuários) e stress (situações extremas).

## **7. Teste de Segurança**

Identifica vulnerabilidades no sistema, protegendo contra ataques e garantindo a segurança dos dados.

## **8. Teste de Usabilidade**

Foco na experiência do usuário, avalia se o software é fácil de usar e atende às necessidades do público-alvo.

## **9. Teste Exploratório**

Executado sem roteiros fixos, ajuda a identificar falhas que testes estruturados podem não detectar.

## **10. Teste de Compatibilidade**

Garante que o software funcione corretamente em diferentes dispositivos, sistemas operacionais e navegadores.

03

# TÉCNICAS DE TESTES DE SOFTWARE

# Técnicas de Testes de softwares

## Diferentes Técnicas de Testes de software

Além dos tipos de testes, as técnicas utilizadas são fundamentais para cobrir diferentes aspectos do software de forma eficiente.

### **1. Caixa-Branca**

Avalia o funcionamento interno do código. Ideal para garantir que algoritmos, lógica e fluxos internos estão corretos. Geralmente usado em testes de unidade.

### **2. Caixa-Preta**

Foco nas entradas e saídas do sistema, sem considerar o funcionamento interno. Verifica se o software atende aos requisitos funcionais esperados.

### **3. Caixa-Cinza**

Combina aspectos das técnicas de caixa-branca e caixa-preta.

### **4. Particionamento de Equivalência**

Divide os dados de entrada em classes ou grupos semelhantes. Testa um valor de cada grupo, reduzindo o número de testes necessários sem comprometer a cobertura.

### **5. Análise de Valor Limite**

Foco nos valores extremos das entradas, como limites superior e inferior de um campo numérico. Identifica erros que geralmente ocorrem nessas bordas.

### **6. Teste Baseado em Cenários**

Cria casos de teste baseados em situações reais de uso. Ajuda a garantir que o software funcione conforme esperado em contextos do dia a dia.



# Técnicas de Testes de softwares

## Diferentes Técnicas de Testes de software

### **7. Tabelas de Decisão**

Usa tabelas para mapear diferentes combinações de entradas e saídas esperadas. Ideal para sistemas com muitas regras de negócio.

### **8. Teste Exploratório**

Executado sem um plano rígido. O testador utiliza sua experiência para explorar o software em busca de comportamentos inesperados.

### **9. Teste Baseado em Riscos**

Prioriza testes nas áreas mais críticas ou vulneráveis do sistema. Ajuda a concentrar esforços onde falhas podem ter maior impacto.

### **10. Técnicas de Modelagem de Estado**

Testa como o software se comporta em diferentes estados (ex.: logado, deslogado) e as transições entre eles. Ideal para sistemas com fluxos complexos.



04

# ATIVIDADES DE TESTES

# ATIVIDADES DE TESTES DE SOFTWARE

## Diferentes Atividades de Testes de software

O processo de testes de software é composto por várias atividades organizadas para garantir a qualidade do sistema. Essas atividades abrangem desde o planejamento até a execução e análise dos resultados. Confira as principais etapas e suas explicações simples:.

### 1. Planejamento dos Testes

**Descrição:** Define a estratégia, objetivos, escopo e recursos necessários para os testes.

**Exemplo:** Criar um plano para testar um aplicativo móvel, incluindo tipos de testes, cronograma e equipe envolvida.

### 2. Análise de Requisitos

**Descrição:** Compreende e revisa os requisitos do sistema para identificar cenários de teste.

**Exemplo:** Avaliar as especificações de uma funcionalidade de login e listar as condições a serem verificadas.

### 3. Design de Casos de Teste

**Descrição:** Desenvolve casos de teste com base nos requisitos e cenários identificados.

**Exemplo:** Criar um caso de teste para validar se o sistema aceita apenas senhas com mais de 8 caracteres.

### 4. Configuração do Ambiente de Teste

**Descrição:** Configura os sistemas, ferramentas e dados necessários para executar os testes.

**Exemplo:** Preparar uma base de dados e instalar ferramentas de automação para testar uma API.

# ATIVIDADES DE TESTES DE SOFTWARE

## Diferentes Atividades de Testes de software

### 5. Execução dos Testes

**Descrição:** Realiza os testes definidos, executando os casos criados e registrando os resultados.

**Exemplo:** Testar a funcionalidade de "Esqueci minha senha" para verificar se o e-mail de recuperação é enviado corretamente.

### 6. Registro e Análise de Defeitos

**Descrição:** Documenta e analisa os problemas encontrados durante a execução dos testes.

**Exemplo:** Reportar um bug onde o botão de "Enviar" não funciona em um formulário específico.

### 7. Avaliação de Resultados

**Descrição:** Compara os resultados esperados com os obtidos e decide se os requisitos foram atendidos.

**Exemplo:** Validar que todas as entradas de um formulário são aceitas corretamente conforme esperado.

### 8. Relatório Final de Testes

**Descrição:** Resumo dos resultados dos testes, incluindo métricas, defeitos encontrados e a qualidade geral do software.

**Exemplo:** Criar um relatório mostrando que 95% dos casos de teste foram aprovados, mas 5% ainda precisam de correção.

### 9. Automação de Testes (Opcional)

**Descrição:** Usa ferramentas para automatizar casos de teste repetitivos ou demorados.

**Exemplo:** Configurar scripts para testar automaticamente a compatibilidade do site em diferentes navegadores.

05

# PRODUTOS DE TRABALHO DE TESTES

# PRODUTOS DE TRABALHO DE TESTES

## Testware

No contexto do **Certified Tester Foundation Level (CTFL)**, os produtos de trabalho de testes são entregáveis criados durante o processo de testes para documentar, acompanhar e garantir a qualidade do software. Abaixo estão os principais produtos de trabalho, com explicações rápidas e simples:

### 1. Plano de Testes

**Descrição:** Documento que define a estratégia, o escopo, os objetivos e os recursos necessários para os testes.

**Exemplo:** Um plano detalhando quais funcionalidades de um aplicativo de e-commerce serão testadas, o cronograma e as ferramentas utilizadas.

### 2. Casos de Teste

**Descrição:** Descrevem as condições de teste, os passos a serem seguidos, as entradas e os resultados esperados.

**Exemplo:** Um caso de teste para verificar se o sistema impede login com senha incorreta.

### 3. Procedimentos de Teste (Scripts)

**Descrição:** Conjunto de passos necessários para executar um ou mais casos de teste.

**Exemplo:** Um script automatizado para testar a funcionalidade de busca em um site.

### Dados de Teste

**Descrição:** Conjunto de informações usadas para testar o sistema. Inclui dados válidos, inválidos e limites.

**Exemplo:** Usuários fictícios com diferentes níveis de acesso para validar permissões no sistema.

# PRODUTOS DE TRABALHO DE TESTES

## Testware

### 5. Relatórios de Defeitos

**Descrição:** Documentam detalhes sobre os problemas encontrados durante os testes, como comportamento inesperado e passos para reproduzir o erro.

**Exemplo:** Um relatório sobre um bug onde o botão "Adicionar ao Carrinho" não funciona corretamente em um navegador específico.

### 6. Relatório de Progresso

**Descrição:** Documento ou painel que monitora o andamento dos testes, destacando casos executados, aprovados e pendentes.

**Exemplo:** Um gráfico mostrando que 70% dos testes foram concluídos, com 15% ainda em execução e 15% com falhas.

### 7. Relatório Final de Testes

**Descrição:** Fornece um resumo geral dos testes realizados, destacando resultados, métricas, defeitos críticos e recomendações.

**Exemplo:** Um relatório indicando que todas as funcionalidades principais foram aprovadas, exceto por pequenas falhas em recursos secundários.

### 8. Logs de Teste

**Descrição:** Registros detalhados das ações realizadas durante a execução dos testes.

**Exemplo:** Um log detalhando cada etapa de um teste automatizado, incluindo tempo de execução e resultados intermediários.

### 9. Checklist de Testes

**Descrição:** Lista de verificação para garantir que todas as atividades e itens de teste foram concluídos.

**Exemplo:** Uma checklist que confirma se todos os testes funcionais e de segurança foram realizados antes do lançamento.

06

# REVISÕES E INSPEÇÕES



# REVISÕES E INSPEÇÕES DE TESTES

No âmbito do **Certified Tester Foundation Level (CTFL)**, as revisões e inspeções são práticas fundamentais para identificar defeitos e melhorar a qualidade do software antes e durante os testes. Estas práticas permitem detectar problemas em documentos, requisitos ou código, garantindo uma base sólida para o processo de desenvolvimento e testes. A seguir, listamos os principais tipos de revisões e inspeções com explicações rápidas e simples:

## 1. Revisão Informal

**Descrição:** Uma revisão sem procedimentos formais, geralmente realizada entre colegas de forma casual.

**Objetivo:** Identificar erros óbvios de forma rápida e econômica.

**Exemplo:** Um desenvolvedor revisa o código de um colega antes de enviá-lo para o repositório.

## 2. Revisão Técnica

**Descrição:** Uma análise mais estruturada, focada em aspectos técnicos do documento ou código.

**Objetivo:** Validar soluções técnicas, identificar erros e melhorar a implementação.

**Exemplo:** Engenheiros revisam a arquitetura de um sistema para garantir que ela atenda aos requisitos de desempenho.

## 3. Revisão Baseada em Checklists

**Descrição:** Revisão guiada por uma lista de verificação, garantindo que todos os pontos críticos sejam analisados.

**Objetivo:** Cobrir sistematicamente os aspectos relevantes de um documento ou código.

**Exemplo:** Uso de um checklist para validar se todos os cenários de teste foram cobertos nos casos de teste.

# REVISÕES E INSPEÇÕES DE TESTES

## 4. Walkthrough (Caminhada Guiada)

**Descrição:** Uma apresentação informal onde o autor guia os revisores pelo documento ou código, explicando os detalhes.

**Objetivo:** Obter feedback inicial e melhorar a compreensão entre as partes interessadas.

**Exemplo:** O autor de um plano de teste apresenta o documento à equipe para receber sugestões.

## 5. Inspeção

**Descrição:** Revisão formal e detalhada com papéis e procedimentos definidos, como moderador, leitor e escritor.

**Objetivo:** Detectar defeitos, coletar métricas e assegurar conformidade com padrões.

**Exemplo:** Uma inspeção de código para verificar a aderência a padrões de segurança.

## 6. Revisão Gerenciada

**Descrição:** Revisão formal conduzida por um moderador com foco em identificar riscos e dependências.

**Objetivo:** Melhorar a qualidade do trabalho e mitigar problemas antes que afetem o projeto.

**Exemplo:** Revisão de requisitos para garantir que não haja ambiguidades antes do início do desenvolvimento.

## 7. Revisão de Requisitos

**Descrição:** Análise específica dos requisitos para garantir clareza, completude e viabilidade.

**Objetivo:** Detectar problemas que poderiam resultar em erros de implementação.

**Exemplo:** Validar se os requisitos de um aplicativo estão claros e abrangem cenários negativos.

# Agradecimientos

# Obrigada por ler até aqui!

**Este Ebook foi gerado por IA e diagramado por humano.**

**Este conteúdo foi gerado para fins didáticos de construção, não foi realizada uma validação Cuidadosa.**

**O passo a passo se encontra no meu GitHub.**



Link GitHub: [MarianaGottardi/Ebook-Testes-de-Software](https://github.com/MarianaGottardi/Ebook-Testes-de-Software)