# Pesquisa: Testes de Codificação Server-Side (Back-End)

Os testes de codificação server-side são um componente essencial da garantia da qualidade de software, com foco em validar as funcionalidades, a lógica de negócios, a integridade dos dados e a segurança de aplicações que operam no lado do servidor. Eles garantem que o servidor responda corretamente às requisições dos clientes, execute operações de forma consistente e mantenha os dados seguros e íntegros.

# 1. Conceito: O Que São Testes Server-Side?

Testes server-side (também chamados de testes de back-end) são voltados para o código que é executado no servidor de uma aplicação. Isso inclui:

- A lógica da aplicação (ex: regras de negócio)
- Operações com banco de dados
- Integração com APIs externas
- Processamento de autenticação/autorização
- Performance e escalabilidade

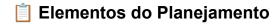
🔑 Diferente dos testes client-side, que avaliam a interface do usuário e interações em navegadores, os testes server-side focam em como o sistema processa os dados e responde às requisições.

# 2. Planejamento dos Testes



#### Conceito

O planejamento de testes é a primeira e uma das etapas mais críticas. Ele define a estratégia de teste, alinhada aos objetivos de qualidade do software.



- **Objetivos**: O que será validado? (ex: autenticação, consultas ao banco)
- Escopo dos testes: Quais funcionalidades do back-end serão testadas?
- Tipos de teste a serem usados:
  - Unitários
  - Integração
  - Funcionais
  - Performance
  - Segurança
- Ambiente de teste: Banco de dados mockado? Testes em homologação?
- Riscos: Possíveis falhas críticas, ambientes instáveis
- Critérios de entrada/saída: Quando começar e quando considerar o teste finalizado

## 📚 Referência:

- IEEE 829:2008 Standard for Software and System Test Documentation
- ISO/IEC 29119-3: Software Testing Test Documentation

# 💢 3. Implementação dos Testes



A implementação transforma os planos em **casos de teste reais**, que são scripts ou procedimentos capazes de verificar comportamentos específicos da aplicação.

# **₹** Tipos de Testes Server-Side

- 1. Teste Unitário
  - Verifica métodos ou funções isoladas
  - Ferramentas: JUnit (Java), Mocha (Node.js), PyTest (Python)

#### 2. Teste de Integração

Testa a comunicação entre módulos (ex: API ↔ banco de dados)

#### 3. Teste Funcional

Valida se uma funcionalidade entrega o resultado esperado

#### 4. Teste de Performance

- Avalia tempo de resposta, carga, stress
- Ferramentas: JMeter, k6, Artillery

### 5. Teste de Segurança

- Verifica falhas como SQL Injection, XSS
- Ferramentas: OWASP ZAP, Postman com scripts

## 📚 Referências:

- Freeman, S., & Pryce, N. (2010). Growing Object-Oriented Software, Guided by Tests
- Fowler, M. (2006). Continuous Integration

# 4. Execução dos Testes

# **★** Conceito

É a etapa em que os testes são executados no ambiente adequado, seja manualmente ou de forma automatizada, com os resultados sendo registrados.

# X Práticas Recomendadas:

- Automatizar testes com ferramentas de CI (GitHub Actions, Jenkins)
- Garantir ambiente isolado (Docker, mocks, bancos de dados temporários)
- Utilizar pipelines de integração contínua para executar testes automaticamente a cada push ou merge

#### Exemplo:

bash
CopiarEditar
npm run test
pytest tests/

## Referência:

• Humble, J., & Farley, D. (2010). Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation

# 1 5. Análise de Resultados dos Testes

# ★ Conceito

Após a execução, é necessário interpretar os resultados dos testes para tomar decisões:

## ✓ Indicadores Analisados

- Cobertura de testes: % do código testado (ex: Cobertura de 80%)
- Taxa de sucesso/falha: Quantos testes passaram vs. falharam
- Relatórios de erro: Logs, rastreamento de falhas
- Testes instáveis: Falhas intermitentes devem ser corrigidas ou removidas

#### **Ferramentas Auxiliares**

- SonarQube: Análise de qualidade e cobertura de código
- Allure, HTML Reports: Relatórios de execução com visualização gráfica

#### 📚 Referência:

- Myers, G. J. (2004). The Art of Software Testing
- Martin, R. C. (2008). Clean Code: A Handbook of Agile Software Craftsmanship

# 📚 Bibliografia Recomendada

- 1. Pressman, R. S. (2016). Engenharia de Software. McGraw-Hill.
- 2. **Sommerville, I.** (2011). *Software Engineering*. Pearson.
- 3. Myers, G. J., Sandler, C., & Badgett, T. (2011). The Art of Software Testing. Wiley.
- 4. **Freeman, S.**, & **Pryce, N.** (2010). *Growing Object-Oriented Software, Guided by Tests*. Addison-Wesley.
- 5. **ISO/IEC 29119**: Padrão internacional para processos e documentação de testes de software.