Fundamentos de Qualidade de Software e Testes

- Guia Profissional de QA

Autora: Juciara E.Conceição
QA Analyst
Publicado em: 27/08/2025

Prefácio

Este livro fornece uma base sólida para quem está começando na área de QA. Ele aborda conceitos teóricos e práticos, exemplos de código, queries de banco de dados, requisições de API, técnicas de automação, pirâmide de testes, quadro Kanban e muito mais. O objetivo é capacitar o leitor a atuar como um guardião da qualidade do software, acompanhando o produto desde a concepção até a manutenção contínua.

Sumário

Etapa 0 – O que é ser um QA

Ser um **Quality Analyst (QA)** significa atuar como **guardião da qualidade do software** durante todo o ciclo de desenvolvimento.

Responsabilidades principais: - Planejar, documentar e executar testes com eficiência. - Aplicar práticas e processos de QA em diferentes fases do SDLC. - Detectar e registrar defeitos, garantindo

rastreabilidade e correção. - Colaborar com desenvolvedores, PO e outros stakeholders para garantir qualidade. - Automatizar testes e validar sistemas, APIs e banco de dados. - Garantir que o software entregue esteja de acordo com requisitos e expectativas do usuário.

Habilidades importantes: atenção aos detalhes, raciocínio lógico, comunicação clara, domínio de ferramentas de teste e conhecimento básico de linguagens de programação.

Etapa 1 – Planejamento e Concepção

1. Introdução à Qualidade de Software

Qualidade de software é a capacidade de um produto atender às expectativas do usuário e aos requisitos do negócio:-**Eficiência:** realiza funções rapidamente. - **Segurança:** protege dados do usuário. - **Desempenho:** responde de forma adequada sob carga. - **Usabilidade:** fácil de usar e entender. - **Manutenibilidade:** simples de atualizar e corrigir problemas.

1.1 Como Nasce o Software?

- Identificação de problemas ou oportunidades
- Ideação e concepção
- Análise de requisitos
- Desenvolvimento
- Testes
- Entrega de valor contínuo

2. Por que Testar Software?

- Identificação precoce de defeitos
- Redução de custos com retrabalho
- Aumento da confiabilidade
- Maior satisfação do cliente

3. Tipos de Testes: Estáticos e Dinâmicos

Tipo	O que é	Objetivo	Exemplos
Estático	Sem executar o sistema	Revisar e prevenir erros	Revisão de código, inspeção de documentos
Dinâmico	Com execução do software	Validar comportamento do sistema	Testes manuais, automatizados

4. Ciclo de Vida do Software (SDLC)

1. Ideação e concepção da ideia – validar a necessidade do software.
2. Levantamento e análise de requisitos – coletar informações de stakeholders, definir funcionalidades.
3. Planejamento e arquitetura técnica – definir tecnologias, frameworks, padrões, cronograma.
4. Desenvolvimento (codificação) – implementação das funcionalidades.
5. Testes e validação – execução de testes unitários, integração, sistema, aceitação, regressão.
6. Homologação e implantação – entrega do produto ao usuário final.
7. Suporte e manutenção – correções e evoluções contínuas

5. Product Discovery e Product Delivery

- **Product Discovery:** validar ideias, estudar viabilidade, entender problemas do usuário.
- **Product Delivery:** entrega contínua de valor, feedback constante, práticas de CI/CD

Etapa 2 – Análise e Design

6. O Papel do QA no Processo

O QA participa de todas as fases, garantindo que cada etapa do software esteja em conformidade com os requisitos: – Planejamento e análise de requisitos – Design de solução e preparação de massa de dados – Revisão de código e automação – Testes manuais, exploratórios e regressivos – Homologação e produção (UAT, beta) – Manutenção e testes contínuos

7. Casos de Teste – Estrutura, Conceito e Feature em Cucumber

Os casos de teste são organizados como **features** em Cucumber, permitindo uma leitura natural e execução automatizada.

Exemplo de Feature – Login com Sucesso:

```
Feature: Login do Usuário
  Como usuário autenticado
  Quero acessar minha
  conta Para visualizar o
  dashboard

  Scenario: Login com sucesso
    Given o usuário está na página de login
    And possui credenciais válidas
    When ele preenche o usuário e a senha
    corretos And clica em Entrar
    Then ele é redirecionado para o dashboard
```

Dados de Entrada: - Usuário: "usuario_teste" - Senha: "senha_teste" - Página de login acessível

Dados de Saída: - Redirecionamento para o dashboard - Sessão iniciada para o usuário - Elementos da interface visíveis e carregados corretamente

Explicação Dado/Quando/Então:
Dado: condição inicial ou pré-requisito (ex: usuário na página de login com credenciais válidas) -
Quando: ação executada pelo usuário (ex: preencher campos e clicar em Entrar)
Então: resultado esperado do teste (ex: redirecionamento para dashboard, sessão iniciada e interface carregada corretamente)

---

8. Técnicas e Abordagens de Teste

- **Caixa Preta:** baseada em requisitos e comportamento externo
- **BDD:** descrição de cenários em linguagem natural usando Gherkin/Cucumber

---

9. Gestão de Defeitos e Jira

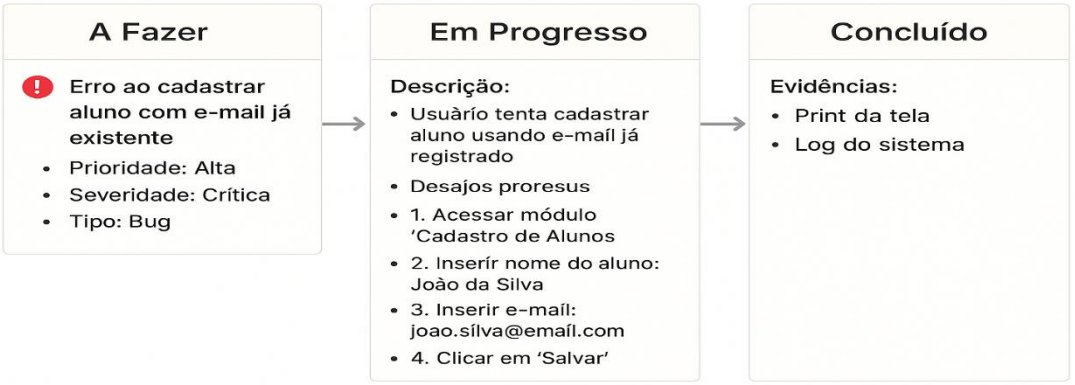
O que é o Jira: Jira é uma ferramenta de gestão de projetos e acompanhamento de tarefas muito utilizada para registro, controle e monitoramento de defeitos, facilitando a comunicação entre equipes de desenvolvimento e QA.

Para que serve: - Registrar bugs e tarefas - Gerenciar prioridades e severidades - Monitorar progresso e status de correções - Facilitar colaboração entre desenvolvedores, QA e PO

Exemplo prático de uso do Jira: - Criar um ticket de bug detalhando título, descrição, prioridade, severidade - Anexar evidências como prints ou logs - Acompanhar status no quadro Kanban

Passos para abrir um bug no Jira:
1. Abrir Jira e selecionar projeto
2. Criar ticket → Tipo: Bug
3. Preencher título: resumo claro e objetivo
4.Preencher descrição: detalhar cenário, passos, resultado esperado e observado
5.Definir prioridade: alta, média ou baixa
6.Definir severidade: crítica, maior, menor, trivial
7.Anexar evidências: prints, logs ou vídeos
8.Salvar e acompanhar o status

Desenvolvimento: Quadro Kanban



Etapa 3 – Desenvolvimento

10. Automação de Testes: Java (Selenium + JUnit + Cucumber)

```
import org.junit.Test;
import org.openqa.selenium.By;
import
org.openqa.selenium.WebDriver; import
org.openqa.selenium.WebElement;
import  org.openqa.selenium.chrome.ChromeDriver;

public class LoginTest {
  @Test
  public void loginSucesso() {
    WebDriver driver = new ChromeDriver();
    driver.get("https://meusite.com/login");
    WebElement usuario =
    driver.findElement(By.id("usuario")); WebElement senha
    = driver.findElement(By.id("senha"));
    usuario.sendKeys("usuario_teste");
    senha.sendKeys("senha_teste");
    driver.findElement(By.id("entrar")).click();
    driver.quit();
  }
}
```

11.Automação de Testes: Python

```python
import unittest
from selenium import webdriver

class TestLogin(unittest.TestCase):
    def test_sucesso(self):
        self.assertEqual(1+1, 2)

if __name__ == "__main__":
    unittest.main()

# Filtro e inspeção Selenium
driver = webdriver.Chrome()
driver.get("https://meusite.com")
element = driver.find_element("id", "campo_busca")
driver.quit()
```

12.Banco de Dados: SQL Avançado

```sql
-- Seleciona usuários ativos
SELECT id, nome, email FROM usuarios WHERE status='ativo';

-- Insere novo usuário
INSERT INTO usuarios (nome,email,status) VALUES ('João
Silva','joao@email.com','ativo');

-- Atualiza usuário
UPDATE usuarios SET status='inativo' WHERE id=10;

-- Deleta usuário
DELETE FROM usuarios WHERE id=15;

-- Consulta com JOIN
SELECT u.nome, p.id_pedido, p.valor
FROM usuarios u
JOIN pedidos p ON u.id=p.id_usuario
WHERE p.valor>100
ORDER BY p.valor DESC;
```

Etapa 4 – Testes e Validação

13.Principais Tipos de Testes e Pirâmide

Tipo de Teste	O que Faz	Objetivo
Unitário	Valida métodos ou funções individuais do código	Garantir que cada unidade funcione corretamente isoladamente; rápido e barato
Integração	Verifica a comunicação entre módulos ou sistemas	Assegurar que componentes diferentes funcionem juntos corretamente, incluindo APIs e serviços integrados
Sistema	Testa o sistema como um todo	Validar se o software completo atende aos requisitos e funciona em diferentes cenários e fluxos
Aceitação (UAT)	Teste realizado pelo usuário final ou PO	Garantir que o software atende às expectativas do cliente e requisitos de negócio
Regressão	Reexecuta testes existentes após mudanças	Garantir que alterações no sistema não quebrem funcionalidades já existentes
Funcional	Verifica funcionalidades específicas do software	Validar requisitos e funcionalidades específicas
Não Funcional	Avalia desempenho, segurança, usabilidade e confiabilidade	Garantir qualidade além das funcionalidades, como tempo de resposta, carga e experiência do usuário

Pirâmide de Testes:

- **Base:** Testes Unitários – rápido, barato, valida métodos
- **Meio:** Testes de Integração – valida comunicação entre módulos/APIs
- **Topo:** Testes de Sistema – valida funcionalidades completas

---

14.API: Conceitos e Exemplos

- GET /usuarios → retorna lista de usuários
- POST /usuarios → cria usuário
- PUT /usuarios/{id} → atualiza usuário
- DELETE /usuarios/{id} → remove usuário

15.Resumo Final para Estudo

- SDLC, papéis do QA, tipos de testes, técnicas, casos de teste, registro de defeitos, automação (Java/Python), SQL, API, filtros, pirâmide de testes, Kanban, visão estratégica e prática.