Descrição do Trabalho – Quality Assurance na Prática

Descrição do Trabalho – Quality Assurance na Prática

Este trabalho tem como objetivo aplicar, de forma prática e contextualizada, os principais conceitos e competências desenvolvidos na disciplina de Quality Assurance. A partir da ementa e dos objetos de aprendizagem estudados, foram reunidos exemplos reais que ilustram o uso de ferramentas, metodologias e estratégias fundamentais para garantir a qualidade de sistemas de software ao longo de seu ciclo de vida.

O conteúdo está estruturado com base nos seguintes pilares:

- Fundamentos de QA, destacando a importância da cultura de qualidade nas organizações e os diferentes tipos de testes: unitário, integração, sistema, aceitação, regressão e exploratório.
- Planejamento e Estratégias de Teste, com foco na criação de critérios de aceitação, planos de teste eficientes, priorização por risco e elaboração de métricas de qualidade.
- Automação de Testes, apresentando ferramentas como Selenium, Cypress, Appium e JMeter, além da integração com pipelines de CI/CD por meio do GitHub Actions. Também são exploradas aplicações de Inteligência Artificial para geração e análise de testes.
- Controle e Monitoramento da Qualidade, envolvendo práticas como auditorias, uso de métricas em tempo real, gestão de bugs e introdução à observabilidade com ferramentas modernas.

A proposta é oferecer uma visão integrada entre teoria e prática, capacitando o aluno a desenvolver soluções de QA alinhadas a metodologias ágeis e DevOps, promovendo a melhoria contínua e a confiabilidade do software. O trabalho apresenta códigos comentados que simulam cenários comuns de testes e validações em sistemas reais, demonstrando o papel estratégico da qualidade na entrega de produtos digitais eficientes e seguros.

2. Teste Unitário - Função calcularDesconto

Testa se a função retorna o valor correto para diferentes inputs.

```
function calcularDesconto(valor, percentual) {
  return valor - (valor * percentual / 100);
}

test('Desconto de 10% sobre 100 deve ser 90', () => {
  expect(calcularDesconto(100, 10)).toBe(90);
});
```

2. Teste de Integração - API de Produtos (Node.js + Supertest)

Verifica se a API GET /produtos responde corretamente após inserção no banco.

```
const request = require('supertest');
const app = require('../app');

describe('GET /produtos', () => {
  it('deve retornar lista de produtos', async () => {
    const response = await request(app).get('/produtos');
    expect(response.statusCode).toBe(200);
    expect(Array.isArray(response.body)).toBe(true);
  });
});
```

2. Teste de Sistema - Simulação de Pedido Completo (Cypress)

Simula um pedido completo em um e-commerce.

```
describe('Pedido Completo', () => {
  it('Deve permitir compra do início ao fim', () => {
    cy.visit('/');
    cy.get('[data-cy=produto-1]').click();
    cy.get('[data-cy=adicionar-carrinho]').click();
    cy.get('[data-cy=finalizar-compra]').click();
    cy.get('[data-cy=confirmar-pedido]').should('contain', 'Pedido
Confirmado');
  });
});
```

3. Planejamento - Critério de Aceitação com Gherkin

Define comportamento esperado usando linguagem natural.

```
Cenário: Aplicar cupom de desconto
Dado que o usuário tenha um cupom válido
Quando ele o aplicar no carrinho
Então o valor total deve ser reduzido conforme o cupom
```

6. Métrica de Qualidade com Cypress

Exemplo de script Cypress que pode ser usado para coleta de taxa de falhas.

```
describe('Teste de login válido', () => {
  it('Deve permitir login com credenciais corretas', () => {
    cy.visit('/login');
    cy.get('#email').type('usuario@teste.com');
    cy.get('#senha').type('123456');
    cy.get('form').submit();
```

```
    cy.url().should('include', '/dashboard');
  });
});
```

7. Automação com JMeter - Configuração XML básica

Simula 500 usuários acessando uma API simultaneamente.

```
<ThreadGroup guiclass="ThreadGroupGui" testclass="ThreadGroup"
testname="Carga 500 usuários" enabled="true">
  <stringProp name="ThreadGroup.num_threads">500</stringProp>
  <stringProp name="ThreadGroup.ramp_time">10</stringProp>
  <stringProp name="ThreadGroup.duration">30</stringProp>
</ThreadGroup>
```