*Santo André, 16 de setembro de 2025.*

**Ana Lauren Dourado Pereira.**

**Laura Araujo Dutra dos Santos.**

# **Testes de Software (Explorando Testes de Software em Projetos de Desenvolvimento).**

**🔹 Tipos de Testes e Exemplos na Calculadora**

## **1. Teste Unitário**

* **O que é: Verifica uma unidade isolada do código, geralmente uma função ou método.**
* **Objetivo: Garantir que cada parte pequena do sistema funciona de forma correta e previsível.**
* **Como executar: Criar testes para as funções matemáticas (soma, subtrai, multiplica, divide).**

**Exemplo:**

**import unittest**

**from calculadora\_funcoes import soma**

**class TestUnitario(unittest.TestCase):**

**def test\_soma(self):**

**self.assertEqual(soma(2, 3), 5) # verifica se 2 + 3 = 5**

**✅ Importância: detecta erros cedo (mais barato corrigir).**

**2. Teste de Integração**

* **O que é: Verifica se diferentes partes do sistema funcionam juntas.**
* **Objetivo: Detectar falhas na comunicação entre módulos.**
* **Como executar: Simular um fluxo que use mais de uma função em sequência.**

**Exemplo:**

**from calculadora\_funcoes import soma, subtrai**

**def test\_fluxo\_integracao():**

**resultado = soma(10, 5) # 15**

**resultado = subtrai(resultado, 3) # 12**

**assert resultado == 12**

**✅ Importância: garante que módulos independentes (soma e subtração) se integram corretamente.**

**3. Teste de Sistema**

* **O que é: Avalia o sistema como um todo, verificando se todos os módulos integrados funcionam juntos.**
* **Objetivo: Simular o uso real do programa.**
* **Como executar: Criar um teste que combine várias operações, como se o usuário estivesse usando a calculadora.**

**Exemplo:**

**from calculadora\_funcoes import soma, multiplica**

**def test\_calculadora\_completa():**

**# Simulação de (2 + 3) \* 4 = 20**

**resultado = multiplica(soma(2, 3), 4)**

**assert resultado == 20**

**✅ Importância: garante que o sistema entregue o resultado final esperado.**

**4. Teste de Aceitação**

* **O que é: Verifica se o software atende ao critério de aceitação do cliente (ou seja, requisitos definidos).**
* **Objetivo: Validar que o sistema resolve o problema do usuário.**
* **Como executar: Definir um critério (exemplo: *“o cliente quer que 7 + 8 = 15”*) e testá-lo.**

**Exemplo:**

**from calculadora\_funcoes import soma**

**def test\_aceitacao():**

**# Critério do cliente: 7 + 8 deve ser 15**

**assert soma(7, 8) == 15**

**✅ Importância: mostra que o sistema realmente resolve o que o cliente pediu.5. Testes Funcionais**

* **O que é: Verificam o que o software faz, ou seja, se cada funcionalidade entrega o resultado esperado.**
* **Objetivo: Validar o comportamento externo de cada funcionalidade.**
* **Como executar: Testar diretamente uma operação da calculadora.**

**Exemplo:**

**from calculadora\_funcoes import divide**

**def test\_divisao\_funcional():**

**assert divide(10, 2) == 5**

**✅ Diferença: aqui você testa a funcionalidade final (ex: “dividir dois números”), sem se preocupar com a lógica interna.**

**6. Testes Não Funcionais**

* **O que é: Avaliam atributos de qualidade do sistema que não são ligados a funcionalidades específicas (ex: desempenho, usabilidade, segurança).**
* **Objetivo: Garantir que o software não apenas funcione, mas funcione bem.**
* **Como executar: Simular, por exemplo, desempenho sob carga.**

**Exemplo de teste de desempenho:**

**import time**

**from calculadora\_funcoes import soma**

**def test\_performance():**

**inicio = time.time()**

**for i in range(100000):**

**soma(i, i+1)**

**fim = time.time()**

**assert (fim - inicio) < 1 # deve rodar em menos de 1 segundo**

**✅ Importância: garante que o sistema seja rápido, seguro, escalável e agradável de usar.**

# **📌 Resumo**

* **Unitário: Testa uma função isolada.**
* **Integração: Testa se funções/módulos funcionam juntas.**
* **Sistema: Testa o sistema completo em cenário real.**
* **Aceitação: Testa se atende ao requisito do cliente.**
* **Funcional: Testa se uma funcionalidade específica funciona.**
* **Não Funcional: Testa desempenho, segurança, usabilidade, etc.**