PUC MINAS - 02/2025

TRABALHO PRÁTICO

TESTES DE SOFTWARE

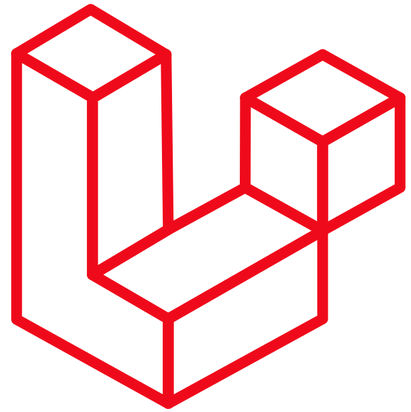
Engenharia de Software

CENÁRIO | FERRAMENTAS

INTRODUÇÃO

O objetivo deste trabalho é demonstrar a aplicação prática de diversas técnicas de teste de software de caixa-branca.

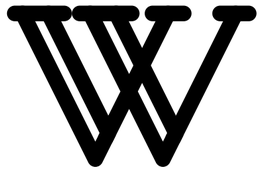
Para isso, desenvolvemos um **DiscountService**, uma classe simples em PHP que centraliza diferentes lógicas de cálculo de desconto.



PHPUnit

MERCADO

EMPRESAS QUE CONFIAM NO PHP E EM TESTES AUTOMATIZADOS:



CLASSIFICAÇÃO DE TESTES

i. Técnica: Teste de Caixa-Branca

A principal abordagem do PHPUnit.
Exige conhecimento profundo da estrutura e lógica interna do código-fonte.

Objetivo: Criar casos de teste que cobrem todos os cenários, incluindo caminhos lógicos, loops e exceções.

ii. Níveis de Teste

Teste de Unidade (Unit)
• **Foco:** Isolar e validar a menor parte do código (método/classe).
• **Rápido,** não inicializa o framework.
• Local (Laravel): /tests/Unit

Teste de Integração (Feature)
• **Foco:** Validar a interação entre componentes (Controller, Model, BD).
• Testa uma funcionalidade completa.
• Local (Laravel): /tests/Feature

iii. Tipos de Teste

Testes Funcionais
• O tipo predominante.
• Objetivo: Garantir que o software segue as regras de negócio e especificações.

Exemplo Prático:
Validar se um cupom de desconto é aplicado corretamente ao preço final de um produto.

TÉCNICAS DE TESTE

1. Particionamento de Equivalência e Análise do Valor Limite
2. Grafo de Causa e Efeito
3. Teste de Transição de Estado
4. Adivinhação de Erro (Error-Guessing)
5. Teste Funcional Sistêmico

CASE 1: DESCONTO POR FAIXA DE VALOR

- Técnica: Análise do Valor Limite
- Método Alvo: calculateTieredDiscount()
- Objetivo: Verificar o comportamento do sistema exatamente nas fronteiras onde a regra de negócio muda.
- Exemplo de Teste:

```php
public function it_applies_correct_discount_at_all_boundaries(): void
{
    // --- Teste da fronteira de R$100 ---
    // Imediatamente abaixo do limite (deve ter 0% de desconto)
    $this->assertEquals(0.0, $this->service->calculateTieredDiscount(99.99), "Falhou abaixo do limite de R$100");
    // No limite exato (deve ter 5% de desconto)
    $this->assertEquals(5.0, $this->service->calculateTieredDiscount(100.00), "Falhou no limite exato de R$100");

    // --- Teste da fronteira de R$500 ---
    // Imediatamente abaixo do limite (deve ter 5% de desconto)
    $this->assertEquals(24.9995, $this->service->calculateTieredDiscount(499.99), "Falhou abaixo do limite de R$500");
    // No limite exato (deve ter 10% de desconto)
    $this->assertEquals(50.0, $this->service->calculateTieredDiscount(500.00), "Falhou no limite exato de R$500");
}
```

CASE 1: DESCONTO POR FAIXA DE VALOR

```php
/**
 * MÉTODO 1: Alvo das técnicas de Particionamento de Equivalência e Análise do Valor Limite.
 * Calcula um desconto por faixas de valor.
 * - Compras < R$100: 0% de desconto.
 * - Compras ⩾ R$100 e < R$500: 5% de desconto.
 * - Compras ⩾ R$500: 10% de desconto.
 *
 * @param float $amount O valor da compra.
 * @return float O valor do desconto.
 */
public function calculateTieredDiscount(float $amount): float
{
    if ($amount < 0) {
        throw new InvalidArgumentException('O valor da compra não pode ser negativo.');
    }

    if ($amount ⩾ 500) {
        return $amount * 0.10;
    }

    if ($amount ⩾ 100) {
        return $amount * 0.05;
    }

    return 0.0;
}
```

CASE 2: DESCONTO POR FIDELIDADE

- Técnica: Grafo de Causa e Efeito
- Método Alvo: calculateLoyaltyDiscountRate()
- Objetivo: Validar se a combinação específica de causas (é membro?, é primeira compra?) produz o efeito esperado.
- Exemplo de Teste:

```php
public function it_covers_all_paths_of_the_loyalty_discount_cause_effect_graph(): void
{
    // Cenário 1: Membro E Primeira Compra → Efeito: 15%
    $this->assertEquals(0.15, $this->service->calculateLoyaltyDiscountRate(true, true), "Falhou para Membro + Primeira Compra");

    // Cenário 2: Membro E NÃO Primeira Compra → Efeito: 10%
    $this->assertEquals(0.10, $this->service->calculateLoyaltyDiscountRate(true, false), "Falhou para Membro + Compra Recorrente");

    // Cenário 3: NÃO Membro E Primeira Compra → Efeito: 5%
    $this->assertEquals(0.05, $this->service->calculateLoyaltyDiscountRate(false, true), "Falhou para Não Membro + Primeira Compra");

    // Cenário 4: NÃO Membro E NÃO Primeira Compra → Efeito: 0%
    $this->assertEquals(0.0, $this->service->calculateLoyaltyDiscountRate(false, false), "Falhou para Não Membro + Compra Recorrente");
}
```

CASE 2: DESCONTO POR FIDELIDADE

```php
/**
 * MÉTODO 2: Alvo da técnica Grafo de Causa e Efeito.
 * Calcula um desconto com base em duas condições (causas) que geram um efeito.
 * - Causa 1: O utilizador é um membro do clube de fidelidade.
 * - Causa 2: É a primeira compra do utilizador.
 *
 * Efeitos (Taxas de desconto):
 * - Membro e Primeira Compra: 15%
 * - Membro, mas não é Primeira Compra: 10%
 * - Não Membro e Primeira Compra: 5%
 * - Não Membro e não é Primeira Compra: 0%
 *
 * @param bool $isMember Se o utilizador é membro.
 * @param bool $isFirstPurchase Se é a primeira compra.
 * @return float A taxa de desconto (ex: 0.15 para 15%).
 */
public function calculateLoyaltyDiscountRate(bool $isMember, bool $isFirstPurchase): float
{
    if ($isMember && $isFirstPurchase) {
        return 0.15;
    }

    if ($isMember && !$isFirstPurchase) {
        return 0.10;
    }

    if (!$isMember && $isFirstPurchase) {
        return 0.05;
    }

    return 0.0;
}
```

CASE 3: DESCONTO POR VOLUME

- Técnica: Teste de Transição de Estado
- Método Alvo: calculateBulkDiscountRate()
- Objetivo: Garantir que a transição de estado (mudança na taxa de desconto) ocorra precisamente no valor correto (de 9 para 10 itens).
- Exemplo de Teste:

```php
public function it_transitions_discount_rate_correctly_across_all_states(): void
{
    // Estado 1: Sem desconto
    $this->assertEquals(0.0, $this->service->calculateBulkDiscountRate(4), "Falhou no estado 'sem desconto'");

    // Transição para o Estado 2 (5-9 itens)
    $this->assertEquals(0.10, $this->service->calculateBulkDiscountRate(5), "Falhou na transição para 5 itens");
    $this->assertEquals(0.10, $this->service->calculateBulkDiscountRate(9), "Falhou no final do estado de 10%");

    // Transição para o Estado 3 (10+ itens)
    $this->assertEquals(0.15, $this->service->calculateBulkDiscountRate(10), "Falhou na transição para 10 itens");
}
```

CASE 3: DESCONTO POR VOLUME

```php
/**
 * MÉTODO 3: Alvo da técnica de Teste de Transição de Estado.
 * Calcula um desconto progressivo com base na quantidade de itens comprados.
 * O "estado" do sistema muda conforme a quantidade de itens atravessa certos limiares.
 * - Estado 1 (1-4 itens): 5% de desconto.
 * - Estado 2 (5-9 itens): 10% de desconto.
 * - Estado 3 (10+ itens): 15% de desconto.
 *
 * @param int $itemCount O número de itens.
 * @return float A taxa de desconto por volume.
 */
public function calculateBulkDiscountRate(int $itemCount): float
{
    if ($itemCount >= 10) {
        return 0.15;
    }

    if ($itemCount >= 5) {
        return 0.10;
    }

    if ($itemCount >= 1) {
        return 0.05;
    }

    return 0.0;
}
```

CASE 4: CUPOM DE DESCONTO FIXO

- Técnica: Adivinhação de Erro (Error-Guessing)
- Método Alvo: applyFixedCoupon()
- Objetivo: Antecipar erros comuns do usuário (ex: espaços extras no cupom) para ver como o sistema reage.
- Exemplo de Teste:

```php
public function it_handles_common_user_errors_when_applying_coupon(): void
{
    // Erro Adivinhado 1: Espaços em branco acidentais.
    $this->assertEquals(10.0, $this->service->applyFixedCoupon(' PROMO10 '), "Falhou ao tratar espaços em branco");

    // Erro Adivinhado 2: Caixa de texto diferente (minúsculas).
    $this->assertEquals(10.0, $this->service->applyFixedCoupon('promo10'), "Falhou ao tratar letras minúsculas");

    // Erro Adivinhado 3: Cupom inexistente.
    $this->assertEquals(0.0, $this->service->applyFixedCoupon('CUPOM_INVALIDO'), "Falhou ao tratar cupom inválido");

    // Erro Adivinhado 4: Entrada nula.
    $this->assertEquals(0.0, $this->service->applyFixedCoupon(null), "Falhou ao tratar entrada nula");
}
```

CASE 4: CUPOM DE DESCONTO FIXO

```php
/**
 * METODO 4: Alvo da técnica de Error-Guessing (Adivinhação de Erro).
 * Aplica um desconto fixo com base num código de cupom.
 * A implementação é propositadamente sensível a erros comuns.
 *
 * @param string|null $couponCode O código inserido pelo utilizador.
 * @return float O valor do desconto fixo.
 */
public function applyFixedCoupon(?string $couponCode): float
{
    // A implementação espera um formato exato, tornando-a frágil.
    if ($couponCode === 'PROMO10') {
        return 10.0;
    }

    if ($couponCode === 'PROMO50') {
        return 50.0;
    }
```

CASE 5: CÁLCULO DO PREÇO FINAL

- Técnica: Teste Funcional Sistêmico
- Método Alvo: getFinalPrice()
- Objetivo: Validar a funcionalidade de ponta a ponta, garantindo que a interação entre todos os métodos funciona como esperado.

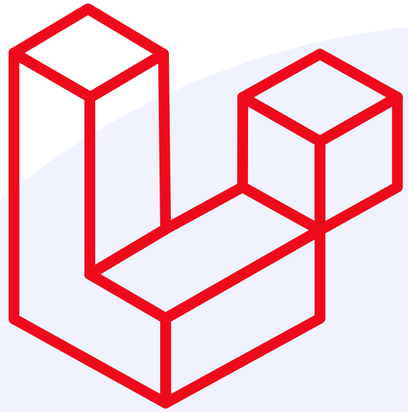
```php
public function it_calculates_the_final_price_for_multiple_complex_scenarios(): void
{
    // --- Cenário 1: Desconto de volume é o maior ---
    $finalPrice1 = $this->service->getFinalPrice(
        baseAmount: 800.0,
        itemCount: 12, // Ativa 15% de desconto por volume
        isMember: true,
        isFirstPurchase: false, // Ativa 10% de desconto de lealdade
        couponCode: 'PROMO50' // Ativa R$50 de desconto fixo
    );

    // Desconto por volume (15% de 800 = 120) é o maior.
    // Preço final: 800 - 120 = 680
    $this->assertEquals(680.0, $finalPrice1, "Falhou no cenário 1 onde o desconto por volume é maior");

    // --- Cenário 2: Desconto de lealdade é o maior ---
    $finalPrice2 = $this->service->getFinalPrice(
        baseAmount: 200.0,
        itemCount: 4, // Não ativa desconto por volume
        isMember: true,
        isFirstPurchase: true, // Ativa 15% de desconto de lealdade
        couponCode: 'PROMO10' // Ativa R$10 de desconto fixo
    );

    // Desconto de lealdade (15% de 200 = 30) é o maior.
    // Preço final: 200 - 30 = 170
    $this->assertEquals(170.0, $finalPrice2, "Falhou no cenário 2 onde o desconto de lealdade é maior");
}
```

CASE 5: CÁLCULO DO PREÇO FINAL

```php
public function getFinalPrice(float $baseAmount, int $itemCount, bool $isMember, bool $isFirstPurchase, ?string $couponCode): float
{
    // 1. Acumula descontos percentuais
    $loyaltyRate = $this->calculateLoyaltyDiscountRate($isMember, $isFirstPurchase);
    $bulkRate = $this->calculateBulkDiscountRate($itemCount);
    $totalPercentageDiscount = $loyaltyRate + $bulkRate;

    $amountAfterPercentageDiscounts = $baseAmount * (1 - $totalPercentageDiscount);

    // 2. Calcula e subtrai o desconto por faixa sobre o novo valor
    $tieredDiscount = $this->calculateTieredDiscount($amountAfterPercentageDiscounts);
    $amountAfterTiered = $amountAfterPercentageDiscounts - $tieredDiscount;

    // 3. Subtrai o cupom fixo
    $couponDiscount = $this->applyFixedCoupon($couponCode);
    $finalPrice = $amountAfterTiered - $couponDiscount;

    // Regra de negócio: o preço nunca pode ser negativo.
    return max(0, $finalPrice);
}
```

CONCLUSÃO

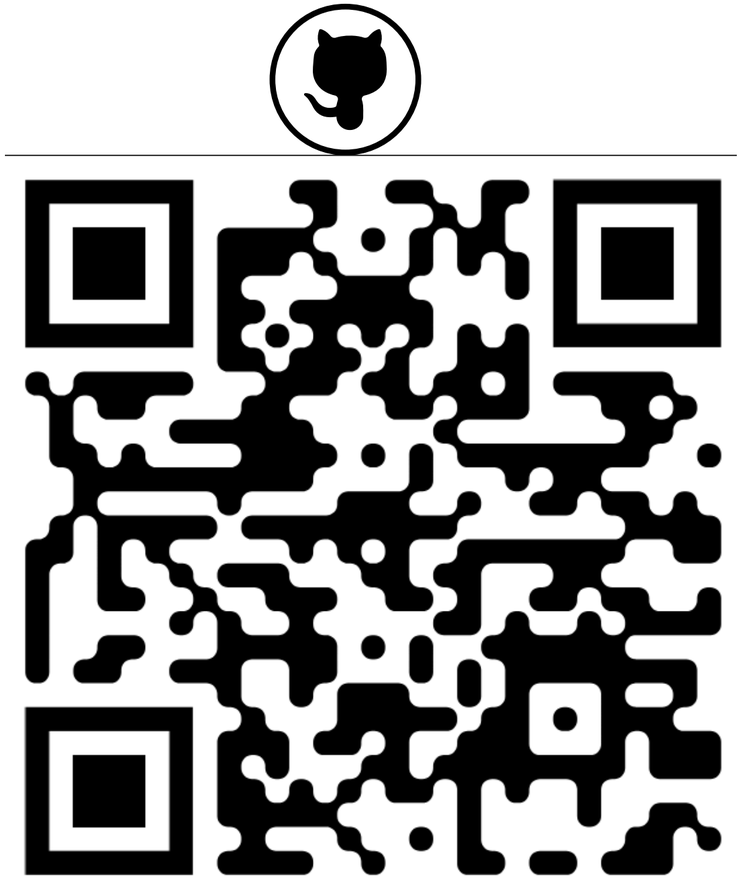
- estes automatizados são fundamentais para garantir a qualidade, estabilidade e manutenibilidade do software.
- O uso de técnicas de teste estruturadas nos permite criar casos de teste mais inteligentes e eficazes.
- Ferramentas como Laravel e PHPUnit tornam o processo de teste muito mais simples e produtivo.



TESTES DE SOFTWARE

OBRIGADO!

DÚVIDAS?



Participantes:

Arthur Freitas Jardim, Carlos Henrique Neimar, Luiz Filipe Nery , João Victor Temponi, Wilken Moreira