CEFET SHOP

GSI9603NF - PROJETO INTEGRADOR DE SISTEMAS

PROFESSOR: THIAGO DELGADO

INTEGRANTES

Marcelli Marinho Assumpção

05578656110@cefet-rj.br

Micaela Diniz da Silva Paes

05578656110@cefet-rj.br

SUMÁRIO

1. Modelo de dados;
2. Diagrama de classes;
3. Práticas adotadas;
4. Códigos da API;
5. Códigos do Frontend;
6. Demonstração da aplicação;
7. Dúvidas;

MODELO DE DADOS

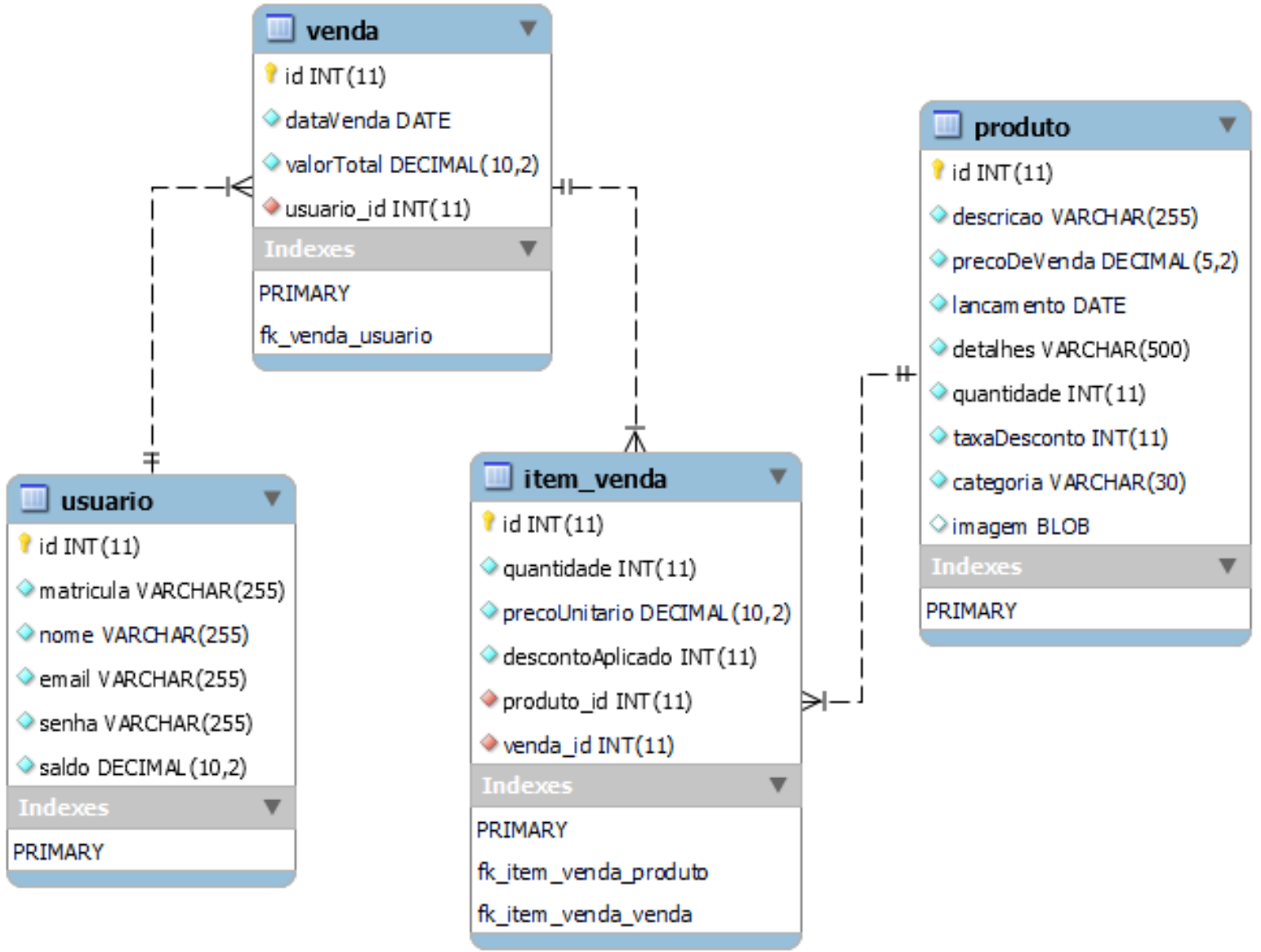

Figura 01 – Modelo de dados da aplicação.

DIAGRAMA DE CLASSES

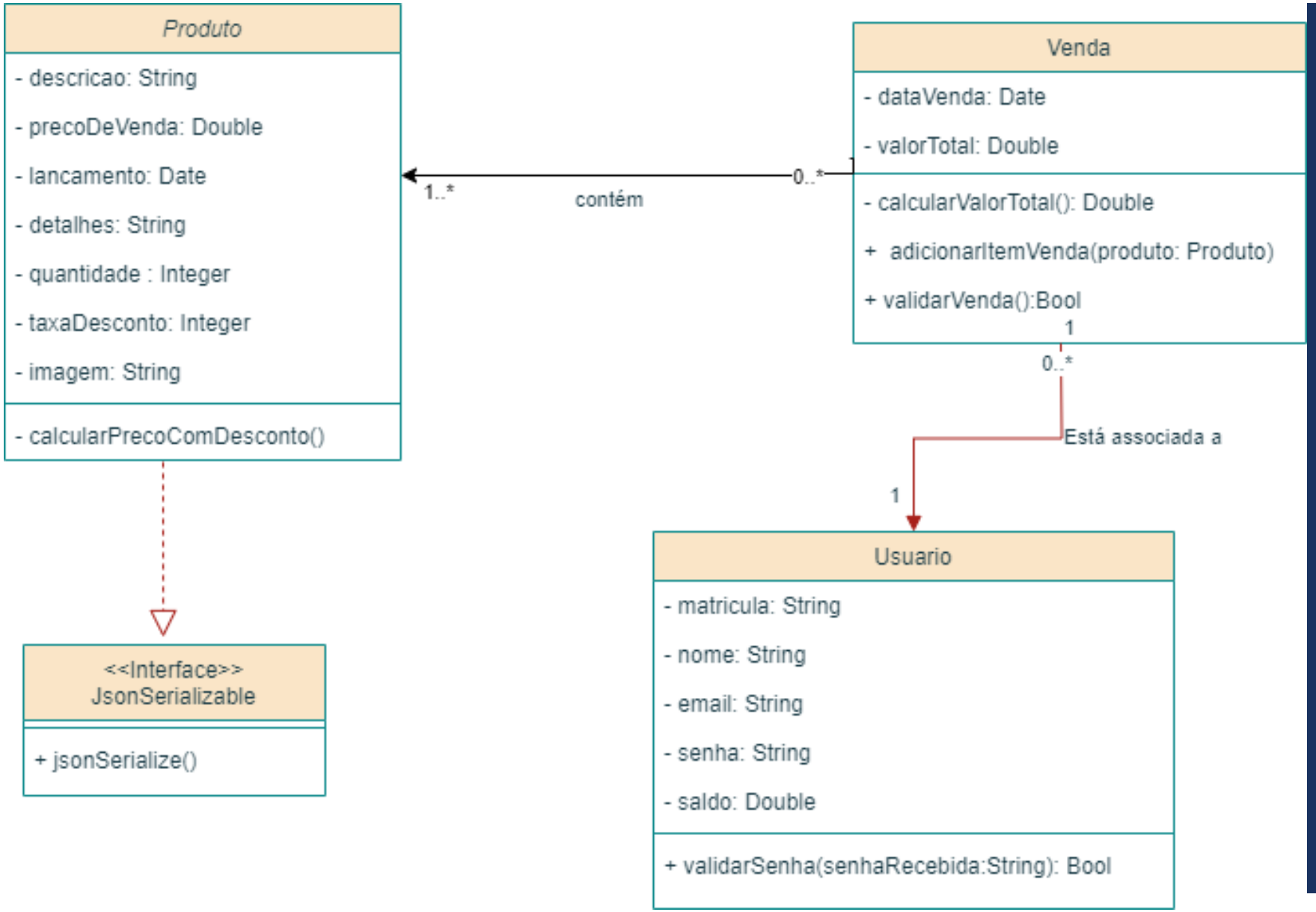

Figura 02 – Diagrama de classes da aplicação.

PRÁTICAS ADOTADAS

- TESTES UNITÁRIOS
- MODELO MVC NO BACKEND E FRONTEND
- CONVENÇÕES DE NOMENCLATURA
- PRETTIER

```
3 references
private function calcularPrecoComDesconto(){
    if( $this->taxaDesconto == null || $this->taxaDesconto <= 0 ){
            $this->precoDesconto =  $this->precoDeVenda;
            $this->taxaDesconto = 0;
            return;
    }
    $this->precoDesconto = $this->precoDeVenda * (1 - ($this->taxaDesconto / 100));
}
```

Figura 03 – Trecho de código da classe Produto.php.

CÓDIGOS DA API

API » SRC » PRODUTO.PHP

```php
public function jsonSerialize() {
    $json = [];
    if (isset($this->id) || $this->id !== 0 ) {
        $json['id'] = $this->id;
    }
    if (isset($this->descricao)) {
        $json['descricao'] = $this->descricao;
    }
    if (isset($this->precoDeVenda) ) {
        $json['precoDeVenda'] = $this->precoDeVenda;
    }
    if (isset($this->lancamento)) {
        $json['lancamento'] = $this->lancamento;
    }
    if (isset($this->detalhes)) {
        $json['detalhes'] = $this->detalhes;
    }
    if (isset($this->quantidade)) {
        $json['quantidade'] = $this->quantidade;
    }
    if (isset($this->taxaDesconto) ) {
        $json['taxaDesconto'] = $this->taxaDesconto;
    }
    if (isset($this->categoria) ) {
        $json['categoria'] = $this->categoria;
    }
```

Figura 04.1 – Trecho de código da classe Produto.php.

```php
    if (isset($this->lancamento)) {
        $json['lancamento'] = $this->lancamento;
    }
    if (isset($this->detalhes)) {
        $json['detalhes'] = $this->detalhes;
    }
    if (isset($this->quantidade)) {
        $json['quantidade'] = $this->quantidade;
    }
    if (isset($this->taxaDesconto) ) {
        $json['taxaDesconto'] = $this->taxaDesconto;
    }
    if (isset($this->categoria) ) {
        $json['categoria'] = $this->categoria;
    }
    if (isset($this->imagem) ) {
        $json['imagem'] = $this->imagem;
    }
    if ( isset($this->precoDesconto)) {
        $json['precoDesconto'] = $this->precoDesconto;
    }
    return $json;
  }
}
```

Figura 04.2 – Trecho de código da classe Produto.php.

CÓDIGOS DA API

API » SRC » PRODUTO.PHP

```php
require_once ('produto.php');
1 reference | 1 implementation | You, 4 days ago | 1 author (You)
interface RepositorioProduto{
    0 references | 1 override
    public function obterPagina( $pagina, $qtdRegistros );
    1 reference | 1 override
    public function obterPorNomeOuId( Produto $produto );
    0 references | 1 override
    public function obterMaisVendidos();

}
```

Figura 05 – Trecho de código da classe repositório-produto.php.

CÓDIGOS DA API

API » SRC » REPOSITORIO-PRODUTO.PHP

```
2 references | 0 implementations | You, 5 seconds ago | 1 author (You)
class RepositorioProdutoEmPDO implements RepositorioProduto{
    4 references
    private $pdo;

    2 references | 0 overrides
    public function __construct( $pdo ){
        $this->pdo = $pdo;
    }
```

Figura 06 – Trecho de código da classe repositório-produto-em-pdo.php.

CÓDIGOS DA API

API » SRC » REPOSITORIO-PRODUTO-EM-PDO.PHP

```php
public function obterPagina($limite, $deslocamento){
    $sql = "SELECT id, descricao, precoDeVenda, imagem, taxaDesconto FROM produto
        LIMIT :limite  OFFSET :deslocamento";
    $produtos = null;
    try{
        $ps = $this->pdo->prepare($sql);
        $ps->setFetchMode(PDO::FETCH_ASSOC);
        $ps->bindParam(':limite', $limite, PDO::PARAM_INT);
        $ps->bindParam(':deslocamento', $deslocamento, PDO::PARAM_INT);
        $ps->execute();
        if( $ps->rowCount()<0 ){
            return null;
        }
        $dados = $ps->fetchAll();
        foreach( $dados as $d ){
            $produtos[] = new Produto( utf8_encode($d['descricao']), doubleval($d['precoDeVenda']),
                null ,null, null, intval($d['taxaDesconto']), null, base64_encode($d['imagem']), null,
                intval($d['id']) );
        }
        return $produtos;

    }catch( PDOException $e ){
        throw new RepositorioProdutoException("Não foi possível obter os produtos");
    }
```

Figura 07 – Trecho de código da classe repositório-produto-em-pdo.php.

CÓDIGOS DA API

API » SRC » REPOSITORIO-PRODUTO-EM-PDO.PHP

```php
public function obterMaisVendidos() {
    $sql = "SELECT itemv.id, p.descricao as descricao,p.id as produto_id, p.precoDeVenda as precoDeVenda,
            p.imagem as imagem,p.taxaDesconto as taxaDesconto,SUM(itemv.quantidade) AS total_vendido
            FROM item_venda itemv
            JOIN produto p ON itemv.produto_id = p.id
            GROUP BY p.id
            ORDER BY total_vendido DESC LIMIT 6";
    $produtos = null;
    try{
        $ps = $this->pdo->prepare($sql);
        $ps->setFetchMode(PDO::FETCH_ASSOC);
        $ps->execute();
        if( $ps->rowCount()<0 ){
            return null;
        }
        $dados = $ps->fetchAll();
        foreach( $dados as $d ){
            $produtos[] = new Produto( utf8_encode($d['descricao']), doubleval($d['precoDeVenda']),
                null ,null, null, intval($d['taxaDesconto']), null, base64_encode($d['imagem']),
                intval($d['total_vendido']), intval($d['produto_id']) );
        }
        return $produtos;

    }catch( PDOException $e ) {
        throw new RepositorioProdutoException( "Não foi possível obter os produtos mais vendidos." );
    }
```

Figura 08 – Trecho de código da classe repositório-produto-em-pdo.php.

CÓDIGOS DA API

API » SRC » REPOSITORIO-PRODUTO-EM-PDO.PHP

```php
public function produtosEmJson( $produtos ) {
    return json_encode($produtos);
}

1 reference | 0 overrides
public function exibirSucesso( $codigo ){
    http_response_code( $codigo );
    header('Content-Type:application/json;charset=utf-8');
}
```

Figura 09 – Trecho de código da classe visao-produto.php.

CÓDIGOS DA API

API » SRC » VISAO-PRODUTO.PHP

```php
public function exibirErro( $mensagem, $codigo ) {
    http_response_code( $codigo );
    header('Content-Type:application/json;charset=utf-8');
    return json_encode( $mensagem );
}

0 references | 0 overrides
public function obterPaginaDesejada(){
    if( isset( $_GET['pag'] ) ) {
        return intval( htmlspecialchars($_GET['pag'] ) );
    }
    return 1;
}
```

Figura 10 – Trecho de código da classe visao-produto.php.

CÓDIGOS DA API

API » SRC » VISAO-PRODUTO.PHP

```php
public function obterNomeOuId() {
    $idRequisicao = 0;
    $nomeRequisicao = null;
    if( isset( $_GET['id'] ) ) {
        $idRequisicao = intval(htmlspecialchars($_GET['id']));
    }else if( isset( $_GET['descricao'] ) ){
        $nomeRequisicao = htmlspecialchars($_GET['descricao']);
    }else{
        return false;
    }
    return new Produto( $nomeRequisicao, null, null, null, null, null,
        null, null, null, $idRequisicao );
}
```

Figura 11 – Trecho de código da classe visao-produto.php.

CÓDIGOS DA API

API » SRC » VISAO-PRODUTO.PHP

```php
1 reference | 0 overrides
public function obterProdutos(){
    try{
        $pagina = $this->visaoProduto->obterPaginaDesejada();
        $deslocamento = $this->calcularDeslocamento( $pagina );
        $produtos = $this->repProdutos->obterPagina(REGISTROS_POR_PAGINA, $deslocamento);
        if( $produtos == null ){
            return $this->visaoProduto->exibirErro( 'Não há produtos para retornar.', 404 );
         }
        $this->visaoProduto->exibirSucesso( 200 );
        return $this->visaoProduto->produtosEmJson( $produtos );
    }catch( RepositorioProdutoException $e ){
        return $this->visaoProduto->exibirErro( "Não foi possível obter os produtos.", 500 );
    }
}
```

Figura 12 – Trecho de código da classe controladora-produto.php.

CÓDIGOS DA API

API » SRC » CONTROLADORA-PRODUTO.PHP

```php
1 reference | 0 overrides
public function obterMaisVendidos(){
    try{
        $produtos = $this->repProdutos->obterMaisVendidos();
        if( $produtos == null ){
            return $this->visaoProduto->exibirErro( 'Não há produtos para retornar.', 404 );
         }
        $this->visaoProduto->exibirSucesso( 200 );
        return $this->visaoProduto->produtosEmJson( $produtos );
    }catch( RepositorioProdutoException $e ){
        return $this->visaoProduto->exibirErro( "Não foi possível obter os produtos mais vendidos.", 500 );
    }
}
1 reference | 0 overrides
public function calcularDeslocamento( $pagina ){
    $deslocamento = REGISTROS_POR_PAGINA * ( $pagina - 1 );
    return $deslocamento;
}
```

Figura 13 – Trecho de código da classe controladora-produto.php.

CÓDIGOS DA API

API » SRC » CONTROLADORA-PRODUTO.PHP

```php
$caminho = $_SERVER['REQUEST_URI'];
$rota = basename($caminho);
$metodo = $_SERVER['REQUEST_METHOD'];

$controladora = new ControladoraProduto();

if ($metodo === 'GET' && preg_match('/^produtos(\?pag=\d{1,5})?$/', $rota)) {
  $produtos = $controladora->obterProdutos();
  echo $produtos;
}else if( $metodo === 'GET' && preg_match('/^produto(?:\?id=\d+)?(?:[&\?]descricao=[^&]{1,255})?$/', $rota ) ) {
    $produto = $controladora->obterPorNomeOuId();
    echo $produto;
}else if( $metodo === 'GET' && preg_match('/^mais-vendidos$/i', $rota ) ){
    $produtos = $controladora->obterMaisVendidos();
    echo $produtos;
}else{
  http_response_code(404);
  header('Content-Type: application/json;charset=utf-8');
  echo json_encode("Desculpe, o recurso que você está procurando não foi encontrado ou a requisição está inválida.");
}
```

Figura 14 – Trecho de código do script index.php.

CÓDIGOS DA API

API » SRC » INDEX.PHP

```php
describe("conexaoPDO", function(){
    it("deve criar a conexão corretamente ao enviar o nome cefet-shop.", function() {
        $pdo = conexaoPDO('cefet-shop');
        expect($pdo)->toBeAnInstanceOf('PDO');
    });
    it("deve lançar PDOException ao enviar o nome de banco de dados inválido.", function() {
        $lancaPdoException = function(){
            conexaoPDO('lazaro');
        };
        expect($lancaPdoException)->toThrow(new PDOException("SQLSTATE[HY000] [1049] Unknown database 'lazaro'", 1049)
    });

});
```

Figura 15 – Trecho de código do script de teste conexao.spec.php

CÓDIGOS DA API

API » SPEC » CONEXAO-PDO.SPEC.PHP

```php
describe("Produto", function(){
    beforeAll(function(){
        $this->produto = new Produto('descricao',100.00,'22-02-2023','detalhes',1,0,'categoria','imagem',1);
    });

    it("deve settar o precoDesconto igual ao precoVenda", function() {
        expect( $this->produto->getPrecoDesconto() )->toBe($this->produto->getPrecoDeVenda() );
    });

    it("deve calcular novamente o precoDesconto ao alterar a taxaDesconto", function() {
        $this->produto->setTaxaDesconto(10);
        expect( $this->produto->getPrecoDesconto() )->toBe( 90.0 );
        $this->produto->setTaxaDesconto(0);
    });

    it("deve calcular novamente o precoDesconto ao alterar o precoVenda", function() {
        $this->produto->setPrecoDeVenda(300.0);
        expect( $this->produto->getPrecoDesconto() )->toBe( 300.0 );
        $this->produto->setPrecoDeVenda(100.0);
    });
```

Figura 16 – Trecho de código de teste conexao-pdo.spec.php

```php
it("deve retornar registros 11 ao 20 quando o limite e deslocamento for igual a 10", function() {
    $produtos = $this->repProdPDO->obterPagina(10,10);
    $primeiroProduto = reset($produtos);
    $ultimoProduto = end($produtos);
     expect( count( $produtos ) )->toBe(10);
     expect($primeiroProduto->getId())->toBe(11);
     expect($ultimoProduto->getId())->toBe(20);
});
it("deve checar se todos os atributos necessários estão armazenados nos objetos de Produto paginados", function() {
  //campos: descricao, id, precoDeVenda, imagem, taxaDesconto, total_vendido, precoDesconto
  $produtosPaginados = $this->repProdPDO->obterMaisVendidos();
  foreach($produtosPaginados as $produto){
      $camposNecessariosPresentes = $produto->getDescricao() != null && $produto->getId() > 0 &&
      $produto->getId() != null && $produto->getPrecoDeVenda() != null && $produto->getImagem() != null &&
      $produto->getTotalVendidos() != null && $produto->getTotalVendidos() > 0 && $produto->getPrecoDesconto() >= 0;
      expect($camposNecessariosPresentes)->toBeTruthy();
  }
});
```

Figura 17 – Trecho de código do teste conexao-pdo.spec.php.

CÓDIGOS DA API

API » SPEC » CONEXAO-PDO.SPEC.PHP

```php
it("deve retornar um objeto produto com todos os atributos do banco de dados", function() {
    $produtoObtido = $this->repProdPDO->obterPorNomeOuId( $this->produto );
    $produtoObtido = end($produtoObtido);
    expect( $produtoObtido )->toBeAnInstanceOf( "Produto" );
    expect( $produtoObtido->getPrecoDeVenda() )->not->toBe(null);
    expect( $produtoObtido->getLancamento() )->not->toBe(null);
    expect( $produtoObtido->getDetalhes() )->not->toBe(null);
    expect( $produtoObtido->getQuantidade() )->not->toBe(null);
    expect( $produtoObtido->getTaxaDesconto() )->not->toBe(null);
    expect( $produtoObtido->getCategoria() )->not->toBe(null);
    expect( $produtoObtido->getImagem() )->not->toBe(null);
    expect( $produtoObtido->getPrecoDesconto() )->not->toBe(null);
});
```

Figura 18 – Trecho de código de teste conexao-pdo.spec.php

CÓDIGOS DA API

API » SPEC » CONEXAO-PDO.SPEC.PHP

DEMONSTRAÇÃO DA APLICAÇÃO

OBRIGADA PELA ATENÇÃO !