1. O que é POO?

POO é um paradigma de programação que organiza o código em "objetos". Esses objetos são instâncias de **classes**, que são como moldes (ou templates) que definem as características e comportamentos desses objetos. Cada objeto tem propriedades (atributos) e métodos (funções).

Estrutura Geral do Projeto

Aqui está um resumo do que cada arquivo faz na sua aplicação:

- conexao.php: Este arquivo contém a classe responsável pela conexão com o banco de dados. Ele encapsula as informações de conexão e métodos para gerenciar a conexão.
- Produto.php: Aqui você tem a classe que representa um produto. Ela possui atributos como id, produto, valor, imagem, além de métodos para realizar operações de CRUD (Criar, Ler, Atualizar e Excluir) no banco de dados.
- index.php: Exibe a lista de produtos cadastrados.
- adicionar.php: Formulário para adicionar um novo produto.
- editar.php: Formulário para editar um produto existente.
- excluir.php: Responsável por excluir um produto.
- uploads/: Pasta onde as imagens dos produtos são armazenadas.

1. Conexão com o Banco de Dados - Conexao.php

A classe Conexao é responsável por criar e gerenciar a conexão com o banco de dados.

```php
class Conexao {
    private $host = "localhost";
    private $port = 7306;
    private $dbname = "poo";
    private $username = "root";
    private $password = "";
    private $conn;

    public function conectar() {
        try {
            $this->conn = new PDO("mysql:host=$this->host;port=$this->port;dbname=$this->dbname",
                                  $this->username, $this->password);
            $this->conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
            return $this->conn;
        } catch (PDOException $e) {
            echo "Erro de conexão: " . $e->getMessage();
            return null;
        }
    }
}
```

- **Atributos privados**: São as configurações de conexão com o banco de dados (host, porta, nome do banco, etc).
- **Método** conectar(): Esse método tenta estabelecer a conexão com o banco de dados usando a classe PDO. Se a conexão for bem-sucedida, retorna a instância da conexão ($this->conn), caso contrário, retorna null.

2. A Classe Produto - Produto.php

A classe Produto representa um produto no seu sistema e contém os atributos e métodos necessários para realizar o CRUD no banco de dados.

```php
class Produto {
    private $id;
    private $produto;
    private $valor;
    private $imagem;
    private $conn;

    public function __construct($id = null, $produto = null, $valor = null, $imagem = null) {
        $this->conn = (new Conexao())->conectar();
        $this->id = $id;
        $this->produto = $produto;
        $this->valor = $valor;
        $this->imagem = $imagem;
    }
```

- **Atributos privados**: Cada produto tem id, produto (nome), valor e imagem. O atributo $conn guarda a conexão com o banco de dados.
- **Construtor** __construct: Este método é chamado automaticamente quando um objeto é criado. Ele inicializa os atributos da classe (como o nome, valor, etc) e também cria a conexão com o banco chamando o método conectar() da classe Conexao.

Métodos de CRUD

- **Adicionar (**adicionar()**)**: Adiciona um novo produto no banco de dados.

```php
public function adicionar() {
    $sql = "INSERT INTO produtos (produto, valor, imagem) VALUES (:produto, :valor, :imagem)";
    $stmt = $this->conn->prepare($sql);
    $stmt->bindParam(':produto', $this->produto);
    $stmt->bindParam(':valor', $this->valor);
    $stmt->bindParam(':imagem', $this->imagem);
    return $stmt->execute();
}
```

- **Editar (**editar()**)**: Atualiza os dados de um produto existente no banco.

```php
public function editar() {
    $sql = "UPDATE produtos SET produto = :produto, valor = :valor, imagem = :imagem WHERE id = :id";
    $stmt = $this->conn->prepare($sql);
    $stmt->bindParam(':id', $this->id);
    $stmt->bindParam(':produto', $this->produto);
    $stmt->bindParam(':valor', $this->valor);
    $stmt->bindParam(':imagem', $this->imagem);
    return $stmt->execute();
}
```

- **Excluir (**excluir()**)**: Exclui um produto do banco de dados.

```php
public function excluir() {
    $sql = "DELETE FROM produtos WHERE id = :id";
    $stmt = $this->conn->prepare($sql);
    $stmt->bindParam(':id', $this->id);
    return $stmt->execute();
}
```

- **Listar (**listar()**)**: Recupera todos os produtos do banco de dados.

```php
public function listar() {
    $sql = "SELECT * FROM produtos";
    $stmt = $this->conn->query($sql);
    return $stmt->fetchAll(PDO::FETCH_ASSOC);
}
```

- **Buscar por ID (**buscarPorId()**)**: Recupera um único produto baseado no seu id.

```php
public function buscarPorId() {
    $sql = "SELECT * FROM produtos WHERE id = :id";
    $stmt = $this->conn->prepare($sql);
    $stmt->bindParam(':id', $this->id);
    $stmt->execute();
    return $stmt->fetch(PDO::FETCH_ASSOC);
}
```

3. Manipulando Produtos na Interface - index.php**,** adicionar.php**,** editar.php**,** excluir.php

- index.php: Exibe a lista de produtos no banco de dados.
- adicionar.php: Formulário para criar um novo produto e adicionar no banco.
- editar.php: Formulário para editar um produto existente.
- excluir.php: Exclui um produto específico.

Esses arquivos interagem com a classe Produto para criar, editar, excluir e listar produtos. Por exemplo:

- No index.php, a lista de produtos é recuperada com o método listar() e exibida em uma tabela HTML.
- No adicionar.php, ao enviar o formulário, o método adicionar() é chamado para inserir os dados no banco.
- No editar.php, o produto é recuperado com o buscarPorId() e atualizado com o editar().
- No excluir.php, o método excluir() é chamado para remover um produto do banco de dados.

4. Vantagens da POO nesta Aplicação

- **Reutilização de Código**: Ao encapsular a lógica de produtos dentro da classe Produto, você pode reutilizar esse código sempre que precisar manipular produtos, sem duplicar lógica.
- **Organização**: A estrutura de classes e métodos torna o código mais organizado e fácil de entender.
- **Facilidade de Manutenção**: Alterações no comportamento de produtos (como validar dados ou alterar a estrutura de banco) podem ser feitas diretamente na classe Produto, sem precisar alterar a lógica do front-end.

Vamos entender o funcionamento das **funções** get **e** set de forma simples e didática, já que elas são muito importantes na Programação Orientada a Objetos (POO). Elas são comumente chamadas de **métodos de acesso** e servem para acessar e modificar os atributos (propriedades) de uma classe. Esses métodos ajudam a manter o princípio da **encapsulação**, que é um dos pilares da POO.

O que são os Métodos get **e** set**?**

- get: O método get é usado para **recuperar** o valor de um atributo da classe. Ele "pega" o valor da propriedade e retorna esse valor.
- set: O método set é usado para **modificar** o valor de um atributo da classe. Ele "define" um novo valor para a propriedade.

Na aplicação de CRUD de produtos que estamos analisando, as funções get e set são usadas para acessar e modificar os valores dos atributos de cada produto, como o nome, o valor e a imagem.

Exemplo Prático no Código

Na classe Produto, temos os seguintes atributos privados:

```
private $id;
private $produto;
private $valor;
private $imagem;
```

Esses atributos são **privados**, o que significa que não podemos acessá-los diretamente de fora da classe. Para acessar ou modificar esses valores, usamos os métodos get e set.

Métodos get **(Obter valor)**

Esses métodos retornam o valor de um atributo da classe. Vamos analisar os métodos get da classe Produto:

```php
public function getProduto() {
    return $this->produto;
}

public function getValor() {
    return $this->valor;
}

public function getImagem() {
    return $this->imagem;
}
```

- getProduto(): Retorna o nome do produto.
- getValor(): Retorna o valor do produto.
- getImagem(): Retorna o nome da imagem do produto.

Métodos set **(Definir valor)**

Esses métodos são responsáveis por atribuir um novo valor a um atributo da classe. Vamos ver os métodos set da classe Produto:

```php
public function setProduto($produto) {
    $this->produto = $produto;
}

public function setValor($valor) {
    $this->valor = $valor;
}

public function setImagem($imagem) {
    $this->imagem = $imagem;
}
```

- setProduto($produto): Atribui um novo valor para o atributo produto.
- setValor($valor): Atribui um novo valor para o atributo valor.
- setImagem($imagem): Atribui um novo valor para o atributo imagem.

Por que usar get **e** set**?**

Agora que entendemos o que essas funções fazem, vamos falar um pouco sobre a **razão** de usá-las na prática. Há algumas boas razões para utilizá-las em vez de acessar ou modificar diretamente os atributos:

1. **Encapsulamento**:
    - O principal motivo para usar get e set é o **encapsulamento**, que é um dos pilares da POO. Encapsular significa proteger o estado interno de um objeto, garantindo que ele só seja modificado de formas controladas.

- Se os atributos fossem públicos, qualquer parte do código poderia alterar os valores diretamente, o que poderia levar a erros ou a mudanças indesejadas.

2. **Validação**:
    - Usando set, você pode adicionar **validações** antes de atribuir valores aos atributos. Por exemplo, se você quiser garantir que o valor de um produto não seja negativo, pode fazer essa verificação dentro do método setValor().
    - Exemplo:

```php
public function setValor($valor) {
    if ($valor < 0) {
        echo "O valor não pode ser negativo!";
        return;
    }
    $this->valor = $valor;
}
```

    - Isso ajuda a evitar dados inválidos ou inconsistentes.
3. **Facilidade de Manutenção**:
    - Se em algum momento você precisar alterar a forma como os dados são armazenados ou calculados, pode fazer isso diretamente nos métodos get e set, sem precisar modificar o código que utiliza esses métodos.
    - Por exemplo, se você decidir que os valores dos produtos sempre precisam ser armazenados em centavos (em vez de reais), pode fazer essa conversão dentro do método setValor().
4. **Segurança**:
    - Os métodos get e set também ajudam a manter a segurança do código. Por exemplo, ao **obter** um valor com get, você tem controle sobre o que é retornado. Se for necessário, pode até modificar os dados antes de retorná-los.
    - Isso permite esconder detalhes de implementação, mantendo a interface do seu código mais limpa e segura.

Exemplo de Uso Prático

Suponha que você quer criar um produto e definir seu valor usando os métodos set e get:

```php
$produto = new Produto();

// Usando set para atribuir valores
$produto->setProduto("Produto A");
$produto->setValor(99.99);
$produto->setImagem("produto_a.jpg");

// Usando get para acessar os valores
echo "Nome do produto: " . $produto->getProduto(); // Produto A
echo "Valor do produto: R$ " . $produto->getValor(); // 99.99
echo "Imagem do produto: " . $produto->getImagem(); // produto_a.jpg
```

Neste exemplo, você usa o set para atribuir valores ao objeto e o get para recuperar esses valores mais tarde. Isso mantém o controle do que está acontecendo internamente na classe, e o código fora da classe não acessa diretamente os atributos.

Conclusão

- **Métodos** get **e** set são uma forma poderosa de controlar como os dados de um objeto são acessados e modificados.
- Eles são essenciais para **encapsulamento**, permitindo que você tenha controle sobre o estado interno dos objetos.
- Eles também são úteis para adicionar **validações** e garantem que os dados de sua aplicação sejam consistentes e seguros.