Disciplina de Desenvolvimento Web

PROFESSOR JEFFERSON CHAVES

jefferson.chaves@ifpr.edu.br

ATIVIDADE AVALIATIVA

A seguir, é apresentado um passo a passo detalhado para criar um CRUD de livros e autores em PHP e MySQL, com o modelo arquitetural MVC, usando uma camada DAO e uma conexão direta ao banco de dados

Passo 0: Preparando o projeto

Crie uma pasta chamada projeto-crud. Organize o projeto da seguinte forma:

```
projeto-crud

├── app
│   ├── controllers
│   ├── models
│   ├── views
│   ├── dao
│   └── config
├── public
│   └── index.php
└── assets
    └── css
```

Descrição dos diretórios:

- controllers: armazena os controladores que mediam a comunicação entre modelos e views.
- models: contém as classes das entidades, como Livro e Autor.
- views: contém os arquivos de visualização (HTML).
- DAO: contém as classes de acesso ao banco de dados (DAOs).
- config: contém a configuração de conexão com o banco de dados.
- public: contém o arquivo index.php para inicialização do projeto.

Passo 1: Configuração do Banco de Dados

Crie um banco de dados no MySQL chamado biblioteca, com duas tabelas: autores e livros.

```
CREATE DATABASE biblioteca;

USE biblioteca;

CREATE TABLE autores (

    id INT AUTO_INCREMENT PRIMARY KEY,

    nome VARCHAR(100) NOT NULL

);

CREATE TABLE livros (

    id INT AUTO_INCREMENT PRIMARY KEY,

    titulo VARCHAR(100) NOT NULL,

    autor_id INT,

    FOREIGN KEY (autor_id) REFERENCES autores(id)

);
```

Passo 2: Configuração da Conexão ao Banco de Dados

No diretório config, crie um arquivo Database.php para gerenciar a conexão ao banco de dados.

```
<?php

class Database {
    private $host = 'localhost';
    private $dbname = 'biblioteca';
    private $user = 'root';
    private $pass = '';

    public function getConnection() {
         try {
              $conn = new PDO("mysql:host=".$host."; dbname=".$dbname,
          $user, $pass);


              $conn→setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
           } catch (\PDOException $e) {
                echo "Erro de conexão: " . $e→getMessage();
           }

        return $conn;
    }
}
?>
```

Passo 3: Criação das Entidades

No diretório `models`, crie as classes `Autor.php` e `Livro.php` que representarão as entidades do sistema.

Autor.php

```
<?php

class Autor {
    private $id;
    private $nome;

    public function getId() {
        return $this→id;
    }

    public function getNome() {
        return $this→nome;
    }

    public function setId($id) {
        $this→id = $id;
    }

    public function setNome($nome) {
        $this→nome = $nome;
    }
}
?>
```

Livro.php

```php
<?php
class Livro {
    private $id;
    private $titulo;
    private $autorId;

    public function getId() {
        return $this->id;
    }

    public function getTitulo() {
        return $this->titulo;
    }

    public function getAutorId() {
        return $this->autorId;
    }

    public function setId($id) {
        $this->id = $id;
    }

    public function setTitulo($titulo) {
        $this->titulo = $titulo;
    }

    public function setAutorId($autorId) {
        $this->autorId = $autorId;
    }
}
?>
```

Passo 4: Criação das Classes DAO

No diretório dao, crie AutorDAO.php e LivroDAO.php para realizar operações de CRUD.

AutorDAO.php

```
<?php

class AutorDAO {
    private $conn;

    public function __construct() {
        $this->conn = Database::getConnection();
    }

    public function findAll() {
        $stmt = $this->conn->query("SELECT * FROM autores");
        return $stmt->fetchAll(\PDO::FETCH_CLASS, Autor::class);
    }

    public function findById($id) {
         $stmt = $this->conn->prepare("SELECT * FROM autores WHERE id =
?");
        $stmt->execute([$id]);
        return $stmt->fetchObject(Autor::class);
    }

    public function create(Autor $autor) {
         $stmt = $this->conn->prepare("INSERT INTO autores (nome) VALUES
(?)");
        $stmt->execute([$autor->getNome()]);
    }

    public function update(Autor $autor) {
         $stmt = $this->conn->prepare("UPDATE autores SET nome = ? WHERE
id = ?");
        $stmt->execute([$autor->getNome(), $autor->getId()]);
    }

    public function delete($id) {
        $stmt = $this->conn->prepare("DELETE FROM autores WHERE id = ?");
        $stmt->execute([$id]);
    }
}
?>
```

LivroDAO.php

```
<?php


class LivroDAO {
    private $conn;

    public function __construct() {
        $this->conn = Database::getConnection();
    }

    public function findAll() {
        $stmt = $this->conn->query("SELECT * FROM livros");
        return $stmt->fetchAll(\PDO::FETCH_CLASS, Livro::class);
    }

    public function findById($id) {
          $stmt = $this->conn->prepare("SELECT * FROM livros WHERE id =
?");
        $stmt->execute([$id]);
        return $stmt->fetchObject(Livro::class);
    }

    public function create(Livro $livro) {
            $stmt = $this->conn->prepare("INSERT INTO livros (titulo,
autor_id) VALUES (?, ?)");
        $stmt->execute([$livro->getTitulo(), $livro->getAutorId()]);
    }

    public function update(Livro $livro) {
           $stmt = $this->conn->prepare("UPDATE livros SET titulo = ?,
autor_id = ? WHERE id = ?");
             $stmt->execute([$livro->getTitulo(), $livro->getAutorId(),
$livro->getId()]);
    }

    public function delete($id) {
        $stmt = $this->conn->prepare("DELETE FROM livros WHERE id = ?");
        $stmt->execute([$id]);
    }
}
?>
```

Passo 5: Controladores

Os controladores vão conectar as views com as classes DAO para realizar operações de CRUD. Eles manipulam as **requisições** e organizam os dados para serem apresentados.

Crie dois controladores: `AutorController.php` e `LivroController.php` no diretório `app/controllers`.

AutorController.php

Este controlador vai gerenciar as ações CRUD para autores.

```
<?php


class AutorController {

    private $autorDAO;

    public function __construct() {
        $this->autorDAO = new AutorDAO();
    }

    public function index() {
        $autores = $this->autorDAO->findAll();
        include __DIR__ . '/../views/autor/index.php';
    }

    public function show($id) {
        $autor = $this->autorDAO->findById($id);
        include __DIR__ . '/../views/autor/show.php';
    }

    public function create() {
        include __DIR__ . '/../views/autor/create.php';
    }

    public function store($nome) {
        $autor = new Autor();
        $autor->setNome($nome);
        $this->autorDAO->create($autor);
        header("Location: /autores");
    }
```

```
    public function edit($id) {
        $autor = $this->autorDAO->findById($id);
        include __DIR__ . '/../views/autor/edit.php';
    }

    public function update($id, $nome) {
        $autor = new Autor();
        $autor->setId($id);
        $autor->setNome($nome);
        $this->autorDAO->update($autor);
        header("Location: /autores");
    }

    public function delete($id) {
        $this->autorDAO->delete($id);
        header("Location: /autores");
    }
}
?>
```

LivroController.php

Este controlador é semelhante ao AutorController, mas lida com a entidade Livro.

```
<?php


class LivroController {
    private $livroDAO;
    private $autorDAO;

    public function __construct() {
        $this->livroDAO = new LivroDAO();
        $this->autorDAO = new AutorDAO();
    }

    public function index() {
        $livros = $this->livroDAO->findAll();
        include __DIR__ . '/../views/livro/index.php';
    }

    public function show($id) {
        $livro = $this->livroDAO->findById($id);
        include __DIR__ . '/../views/livro/show.php';
```

```
    }

    public function create() {
        $autores = $this->autorDAO->findAll(); // Para selecionar o autor
do livro
        include __DIR__ . '/../views/livro/create.php';
    }

    public function store($titulo, $autorId) {
        $livro = new Livro();
        $livro->setTitulo($titulo);
        $livro->setAutorId($autorId);
        $this->livroDAO->create($livro);
        header("Location: /livros");
    }

    public function edit($id) {
        $livro = $this->livroDAO->findById($id);
        $autores = $this->autorDAO->findAll(); // Para selecionar o autor
do livro
        include __DIR__ . '/../views/livro/edit.php';
    }

    public function update($id, $titulo, $autorId) {
        $livro = new Livro();
        $livro->setId($id);
        $livro->setTitulo($titulo);
        $livro->setAutorId($autorId);
        $this->livroDAO->update($livro);
        header("Location: /livros");
    }

    public function delete($id) {
        $this->livroDAO->delete($id);
        header("Location: /livros");
    }
}
?>
```

Passo 6: Views

No diretório views, crie os arquivos para listar, criar, editar e mostrar os autores e livros.

Views para Autores (`views/autor/`):

- `index.php`: lista todos os autores.
- `create.php`: formulário para criar um novo autor.
- `edit.php`: formulário para editar um autor.

Views para Livros (`views/livro/`):

- `index.php`: lista todos os livros.
- `create.php`: formulário para criar um novo livro (inclui a seleção do autor).
- `edit.php`: formulário para editar um livro.

Exemplo de `index.php` para Autores:

```
<!-- app/views/autor/index.php -->
<h1>Lista de Autores</h1>
<a href="/autores/create">Novo Autor</a>
<table>
    <tr>
        <th>ID</th>
        <th>Nome</th>
        <th>Ações</th>
    </tr>
    <?php foreach ($autores as $autor): ?>
    <tr>
        <td><?= $autor->getId(); ?></td>
        <td><?= $autor->getNome(); ?></td>
        <td>
            <a href="/autores/edit?id=<?= $autor->getId(); ?>">Editar</a>
                    <a  href="/autores/delete?id=<?=  $autor->getId();
?>">Excluir</a>
        </td>
    </tr>
    <?php endforeach; ?>
</table>
```

Exemplo de create.php para Autores:

```
<!-- app/views/autor/create.php -->
<h1>Novo Autor</h1>
<form action="/autores/store" method="post">
    <label>Nome: </label>
    <input type="text" name="nome">
    <button type="submit">Salvar</button>
</form>
<a href="/autores">Voltar</a>
```

Repita o processo para as views de livros.