SW-I
SISTEMAS WEB I

Prof. Anderson Vanin

AULA 14 – CONEXÃO COM BANCO DE DADOS MYSQL

Objetivo

- Criar um banco de dados **MySQL**
- Conectar ao banco com **PDO**
- Executar comandos **CRUD**: Criar, Ler, Atualizar e Excluir dados de uma tabela de produtos.

CRUD

CRUD é um acrônimo das palavras em inglês:

- **C** → Create (Criar)
- **R** → Read (Ler)
- **U** → Update (Atualizar)
- **D** → Delete (Excluir)

O que o CRUD representa?

O CRUD representa as **quatro operações básicas que podemos fazer com dados** em um sistema, especialmente em aplicações web que usam banco de dados.

Explicação simples

Imagine um sistema de cadastro de produtos. Com base no CRUD, você poderá:

Letra	Operação	O que faz	Exemplo
C	Criar	Adiciona um novo registro no banco	Cadastrar um novo produto
R	Ler	Consulta e exibe dados	Listar todos os produtos
U	Atualizar	Altera informações existentes	Alterar o preço de um produto
D	Excluir	Remove dados do banco	Apagar um produto do sistema

CRUD com PHP e MySQL

No PHP com MySQL, o CRUD normalmente é feito assim:

Operação	SQL usado	Exemplo
Create	`INSERT`	`INSERT INTO produtos (...) VALUES (...)`
Read	`SELECT`	`SELECT * FROM produtos`
Update	`UPDATE`	`UPDATE produtos SET nome = ... WHERE id = ...`
Delete	`DELETE`	`DELETE FROM produtos WHERE id = ...`

1. Criando o Banco de Dados e a Tabela

Passo 1: Crie o banco de dados

No **phpMyAdmin** ou no ***cmd***, execute:

```
CREATE DATABASE loja;
```

Passo 2: Use o banco criado

```
USE loja;
```

Passo 3: Crie a tabela produtos

```
CREATE TABLE produtos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nome VARCHAR(100) NOT NULL,
    preco DECIMAL(10,2) NOT NULL,
    estoque INT NOT NULL
);
```

O que é o PDO (PHP Data Objects)?

O **PDO (PHP Data Objects)** é uma **interface de acesso a bancos de dados no PHP** que permite se conectar e interagir com diferentes tipos de bancos de forma **segura, orientada a objetos e moderna**.

Por que usar o PDO?

- Suporta múltiplos bancos de dados (MySQL, PostgreSQL, SQLite, Oracle, etc).
- Permite trocar de banco de dados sem alterar muito o código.
- Oferece métodos preparados (**prepared statements**) que evitam ataques de **SQL Injection**.
- É mais flexível e seguro do que funções antigas como ***mysql_**** e até ***mysqli_****.

Como **prepared statements** evitam SQL Injection?

O que é SQL Injection?

É um tipo de ataque onde o usuário mal-intencionado insere comandos SQL diretamente em campos de entrada para roubar, apagar ou modificar dados do banco.

Por exemplo:

```
$usuario = $_GET['usuario'];
$sql = "SELECT * FROM usuarios WHERE nome = '$usuario'";
```

Se alguém digitar na URL:

```
?usuario=' OR '1'='1
```

Como **prepared statements** evitam SQL Injection?

O SQL final será:

```
SELECT * FROM usuarios WHERE nome = '' OR '1'='1'
```

Ou seja, retorna todos os usuários!

Como o prepare() protege?

Com ***prepared statements***, o SQL é separado dos dados. O banco entende que o que você **envia são valores e não comandos SQL**. Isso impede que códigos maliciosos sejam executados.

Como funcionam: prepare(), bindParam() e execute()

Esses três métodos trabalham juntos para enviar dados de forma segura ao banco.

1. **prepare()**

Prepara o SQL, mas ainda não executa. Os valores são substituídos por **placeholders** (:nome, :preco, etc.).

```
$sql = "INSERT INTO produtos (nome, preco) VALUES (:nome, :preco)";
$stmt = $pdo->prepare($sql);
```

Como funcionam: prepare(), bindParam() e execute()

2. **bindParam()**

Liga (associa) cada placeholder a uma **variável PHP**.

```php
$nome = "Mouse";
$preco = 89.90;


$stmt->bindParam(':nome', $nome);
$stmt->bindParam(':preco', $preco);
```

Como funcionam: prepare(), bindParam() e execute()

3. **execute()**

Executa o comando SQL já com os valores corretos. É nesse momento que a consulta realmente acontece no banco.

```
$stmt->execute();
```

Resumo do fluxo

```php
// 1. Prepara
$stmt = $pdo->prepare("INSERT INTO produtos (nome, preco) VALUES (:nome, :preco)");

// 2. Liga os parâmetros
$stmt->bindParam(':nome', $nome);
$stmt->bindParam(':preco', $preco);

// 3. Executa
$stmt->execute();
```

2. Conectando com PDO (**conexao.php**)

```php
<?php
   $host = 'localhost';
   $dbname = 'loja';
   $user = 'root';
   $pass = '';

   try {
       $pdo = new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);
       // Habilita erros do PDO
       $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
       echo "Conexão bem-sucedida!";
   } catch (PDOException $e) {
       echo "Erro na conexão: " . $e->getMessage();
   }
?>
```

3. Criar (INSERT) (**inserir.php**)

```php
<?php
   require 'conexao.php';
   $nome = "Teclado Gamer";
   $preco = 199.99;
   $estoque = 10;
   $sql = "INSERT INTO produtos (nome, preco, estoque) VALUES (:nome, :preco, :estoque)";
   $stmt = $pdo->prepare($sql);

   $stmt->bindParam(':nome', $nome);
   $stmt->bindParam(':preco', $preco);
   $stmt->bindParam(':estoque', $estoque);

   if ($stmt->execute()) {
       echo "Produto inserido com sucesso!";
   } else {
       echo "Erro ao inserir produto.";
   }
?>
```

4. Ler (SELECT)(**listar.php**)

```php
<?php
  require 'conexao.php';

  $sql = "SELECT * FROM produtos";
  $stmt = $pdo->query($sql);

  while ($produto = $stmt->fetch(PDO::FETCH_ASSOC)) {
      echo "ID: " . $produto['id'] . "<br>";
      echo "Nome: " . $produto['nome'] . "<br>";
      echo "Preço: R$" . $produto['preco'] . "<br>";
      echo "Estoque: " . $produto['estoque'] . "<br><br>";
  }
?>
```

5. Atualizar (UPDATE)(**atualizar.php**)

```php
<?php
  require 'conexao.php';

  $id = 1;
  $novoPreco = 149.99;

  $sql = "UPDATE produtos SET preco = :preco WHERE id = :id";
  $stmt = $pdo->prepare($sql);

  $stmt->bindParam(':preco', $novoPreco);
  $stmt->bindParam(':id', $id);

  if ($stmt->execute()) {
      echo "Produto atualizado com sucesso!";
  } else {
      echo "Erro ao atualizar produto.";
  }
?>
```

6. Excluir (DELETE)(**excluir.php**)

```php
<?php
  require 'conexao.php';

  $id = 1;

  $sql = "DELETE FROM produtos WHERE id = :id";
  $stmt = $pdo->prepare($sql);

  $stmt->bindParam(':id', $id);

  if ($stmt->execute()) {
      echo "Produto excluído com sucesso!";
  } else {
      echo "Erro ao excluir produto.";
  }
?>
```

Exercício

- Criar um novo produto com nome e preço personalizados.
- Listar todos os produtos.
- Atualizar o nome ou estoque de um produto.
- Apagar um produto da lista.