

## 12.2. PDO (PHP Data Object)

Neste tópico vamos falar a respeito do PDO, essa biblioteca (DRIVER) tão poderosa do PHP.

Antes de começar a trabalhar com o PDO, é necessário habilitar o driver do PDO e o driver referente ao banco que será utilizado. Para habilitar o PDO basta remover o comentário (;) da linha do quadro 64, no arquivo php.ini que encontra-se dentro do diretório onde foi instalado o PHP.

Quadro 64 - Habilitando PDO no Windows

```
extension=pdo_mysql
```

PDO\_MYSQL é um driver que implementa a interface PHP Data Objects (PDO) - [http://www.php.net/manual/pt\\_BR/intro.pdo.php](http://www.php.net/manual/pt_BR/intro.pdo.php) para acesso do PHP ao MySQL.

A utilização do PDO fornece uma camada de abstração em relação a conexão com o **banco de dados** visto que o PDO efetua a conexão com diversos bancos de dados da mesma maneira, modificando apenas a sua string de conexão. O quadro 65 mostra um exemplo de conexão utilizando o PDO.

Quadro 65

```
<?php

    $host = 'localhost';
    $usuario = 'root';
    $senha = '';
    $banco = 'trabalho';
    $dsn = "mysql:host={$host};port=3306;dbname={$banco}";
    try {
        $pdo = new PDO($dsn, $usuario, $senha);
    }
    catch (PDOException $e) {
        die($e->getMessage());
    }

?>
```

A **classe PDO em** sua instancia pede como parâmetro primeiro o banco que será utilizado, O caminho do banco de dados e o nome da base de dados. Após devemos inserir o login e a senha do banco de dados.

Para conectarmos com qualquer outro banco de dados, basta modificarmos a string contida na variável \$dsn.

Para trabalhar com operações no banco, o PDO possui um método conhecido como prepare. Como o próprio nome diz, este método apenas prepara uma operação no banco de dados, logo se faz necessário a utilização de outros métodos como execute por exemplo, para realmente executar a operação.

No quadro 66 é mostrado um exemplo de código para inserção de dados no banco, imagine a existência de uma tabela chamada de usuário com os campos login, senha e administrador e que será utilizada a biblioteca PDO para inserir nesta tabela.

Quadro 66

```
$login= $_POST['login'];  
  
$senha = $_POST['senha'];  
  
$administrador=0;  
  
// preparando o comando que insere nessa tabela os dados  
  
$sql = "INSERT INTO usuario (login, senha, administrador) VALUES  
(:login, :senha, :administrador)";  
  
$stmt = $pdo->prepare($sql);  
  
$stmt->bindParam(':login', $login, PDO::PARAM_STR);  
  
$stmt->bindParam(':senha', $senha, PDO::PARAM_STR);  
  
$stmt->bindParam(':administrador',$administrador, PDO::PARAM_INT);  
  
$stmt->execute();
```

?>

Lembrando, os métodos utilizados neste exemplo são apenas alguns dos métodos existentes na biblioteca PDO, para o conhecimento de todos os métodos sugiro que acessem ao **Manual do PHP** ([http://www.php.net/manual/pt\\_BR/book.pdo.php](http://www.php.net/manual/pt_BR/book.pdo.php))

Conforme podemos observar no código acima, primeiramente houve uma preparação no SQL que será enviado ao banco de dados. O método *prepare* ele apenas inicia uma query. Repare os `:login`, `:senha` e `:administrador`, que devem ser substituídos pelos valores reais adicionado.

Os valores deverão ser adicionados com o método *bindParam*, este método utiliza 2 ou 3 passagens de parâmetros. No exemplo foram feitas as três passagens de parâmetros, sendo a primeira referente ao literal (por exemplo, `:login`) no qual será substituído, o segundo parâmetro referente ao valor que será inserido no banco e o terceiro parâmetro que é referente ao tipo de valor que será enviado.

Com isso já foi possível inserir no banco de dados com o auxílio da biblioteca, com os dados já inseridos, o código do quadro 67 mostra como buscar todos os valores da tabela usuário.

Quadro 67

```
<?php

    // estabelecendo a conexão com o banco de dados
    require_once('dbConn.php');

    $result = $con->query("SELEC login, senha, administrador FROM usuario");
    while($row = $result->fetch(PDO::FETCH_OBJ)){
        echo $row->login . "<br />";
        echo $row->senha . "<br />";
        echo $row->administrador . "<br />";
    }

?>
```

Claro que isso é só um exemplo, pois não é recomendado listar senhas.

No código acima temos uma consulta sem passagem de parâmetro de todos os dados armazenados na tabela usuário. Utilizamos o método *query* para isso. O método *query* irá armazenar na variável `$result` todos os dados referentes a consulta do banco.

Após a variável `$result` ser populada, devemos utilizar um `while` para percorrer esses dados e trabalharmos com os valores. Dentro do `while` inserimos o método `fetch` do PDO que tem como função resgatar linha a linha do resultado da consulta. No método `fetch` utilizamos um parâmetro `PDO::FETCH_OBJ`, este parâmetro define a forma na qual os dados serão retornados e armazenados na variável `$row`, criada dentro do `while`.

O `PDO::FETCH_OBJ` trata cada linha da consulta como um objeto, transformando os campos que foram retornados em atributos do objeto `$row`. Para acessar estes atributos, utiliza-se os caracteres `->` (traço maior) e o nome do campo buscado.

Para criar uma consulta com passagem de parâmetros, será necessário a utilização do método *prepare*, pois o mesmo suporta inserção de elementos após a criação da query. O código do quadro 68 mostra a consulta dos dados da tabela usuário realizada através de uma igualdade do nome.

Quadro 68 – Consulta por nome

```
// estabelecendo a conexão com o banco de dados
require_once('dbConn.php');
$login=$_POST['login'];

$result = $con->prepare("SELECT login, senha, administrador FROM usuario WHERE
nome LIKE ?");
$result->bindParam(1, $login . "%");
if($result->execute())
{
    if($result->rowCount() > 0)
    {
        while($row = $result->fetch(PDO::FETCH_OBJ))
        {
            echo $row->login . "<br />";
            echo $row->senha . "<br />";
            echo $row->administrador . "<br />";
        }
    }
}
```

O método prepare foi utilizado no exemplo acima com a ideia apenas de iniciar a query e aguardar pela inclusão de valores posteriormente. Repare que nesse exemplo, foi utilizado a ?, essa interrogação (representada pelo 1 no bindParam) será substituída pelo valor real do nome através da variável \$nome. O nome foi inserido em um segundo momento com o % (per cento), pois em SQL ele representa um coringa. Logo estaremos buscando por todas as pessoas armazenadas no banco de dados cujo seu nome inicie com a(s) letra(s) informada na variável \$nome por exemplo.

Uma outra forma de codificar esse exemplo é recomendada pelo autor do material é a utilização de literal no lugar da ?, o quadro 69 mostra como fazer.

Quadro 69

```
// estabelecendo a conexão com o banco de dados
require_once('dbConn.php');
$login=$_POST['login'];

$result = $con->prepare("SELECT login, senha, administrador FROM usuario WHERE
nome LIKE :login");
$result->bindParam(':login', $nome . "%");
if($result->execute())
{
    if($result->rowCount() > 0)
    {
        while($row = $result->fetch(PDO::FETCH_OBJ))
        {
            echo $row->login . "<br />";
            echo $row->senha . "<br />";
            echo $row->administrador . "<br />";
        }
    }
}
```

Os quadros 70 e 71 mostram as operações de remoção e atualização de dados no banco teremos um processo parecido com a inserção.

#### Quadro 70 – Remoção de dados

```
// estabelecendo a conexão com o banco de dados  
require_once('dbConn.php');  
$login=$_POST['login'];  
  
$stmt = $con->prepare("DELETE FROM usuario WHERE login = :login");  
$stmt->bindParam(':login', $login);  
$stmt->execute();
```

#### Quadro 71 – Alteração de dados

```
// estabelecendo a conexão com o banco de dados  
require_once('dbConn.php');  
$login=$_POST['login'];  
$senha=$_POST['senha'];  
$adm=$_POST['adm'];  
  
$stmt = $con->prepare("UPDATE usuario SET senha = :senha, administrador = :adm  
                        WHERE login = :login");  
$stmt->bindParam(':login', $login );  
$stmt->bindParam(':senha', $senha);  
$stmt->bindParam(':adm', $adm);  
$stmt->execute();
```

Repare que os literais foram nomeados com :login, :senha, :adm. Mas poderiam ser qualquer nomes, como por exemplo, user, psw e administrator por exemplo. A nomenclatura :nome é uma opção de uso do autor.

### 12.3. PHPMyAdmin

O phpMyAdmin é uma aplicação Web contida no painel de controle do Xampp que serve para controlar seu banco de dados MySQL. Você pode utilizá-lo para criar, copiar, deletar, renomear e alterar tabelas, fazer a manutenção de tabelas, deletar, editar e adicionar campos, exportar ou importar um banco de dados, e muito mais. A figura 8 mostra a tela do painel de controle que tem a chamada para o phpmyadmin realçada com um círculo em vermelho.

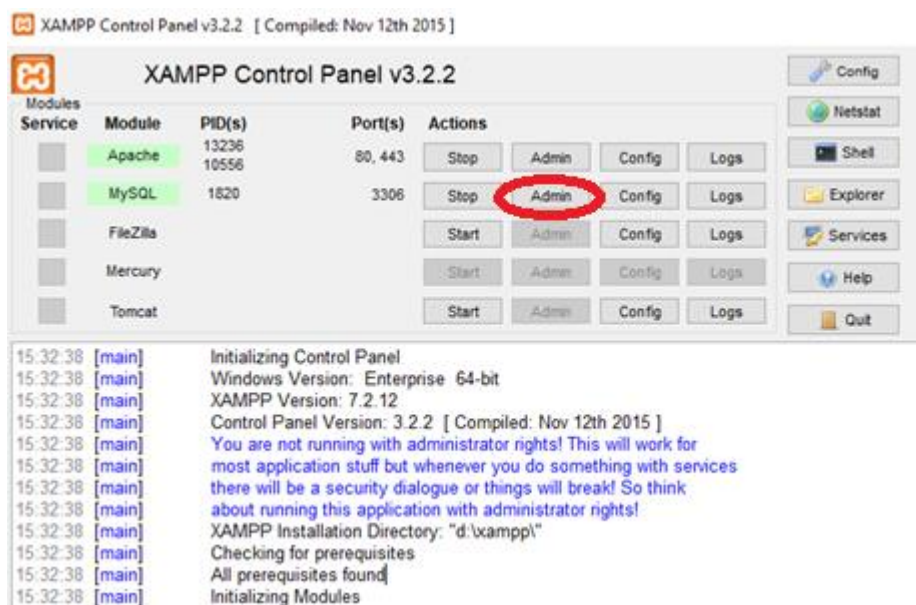


Figura 8 – Tela inicial do painel de controle do Xampp

Vale ressaltar, que o botão admin destacado com o círculo em vermelho, só fica disponível se o **MySQL for inicializado**. Mas somente será possível rodar o phpmyadmin se o **servidor apache também for inicializado**, pois lembre-se, o phpmyadmin disponível no Xampp é uma aplicação Web PHP, portanto precisa de um servidor. A figura 9 ilustra a tela inicial do phpmyadmin.

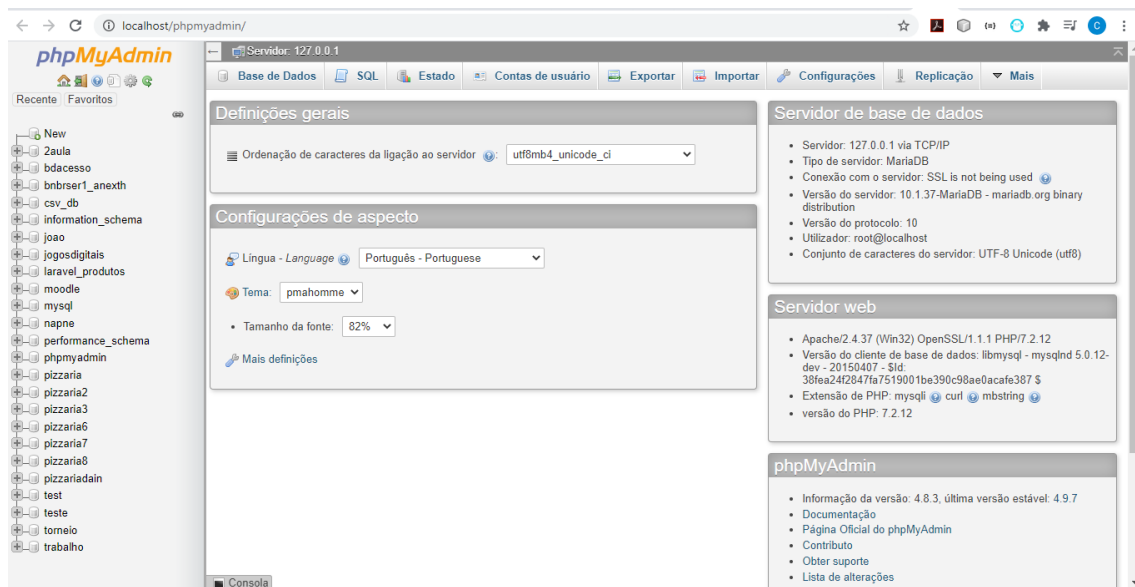


Figura 8 – Tela inicial do phpmyadmin

A tela inicial do PHPMyAdmin mostra várias áreas de configuração. Nela é possível, criar tabelas, inserir, alterar, remover e consultar dados. Além de criar usuários, dar e revogar permissões de acessos, fazer backup, exportar e importar dados entre outras funções.

**IMPORTANTE:** quem tem todas as permissões de acesso à base de dados é o usuário root, no servidor localhost, sem a necessidade de digitação de senha.

As figuras 9 e 10 mostram as telas responsáveis pela criação de um banco de dados e uma tabela respectivamente.

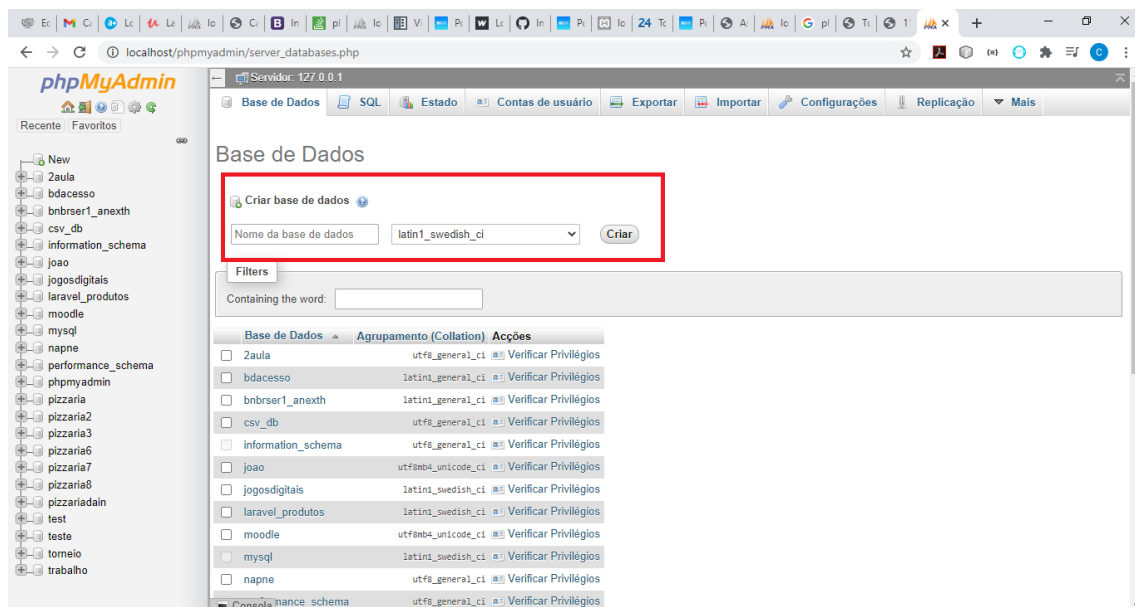


Figura 9 – phpmyadmin – Criação de banco de dados



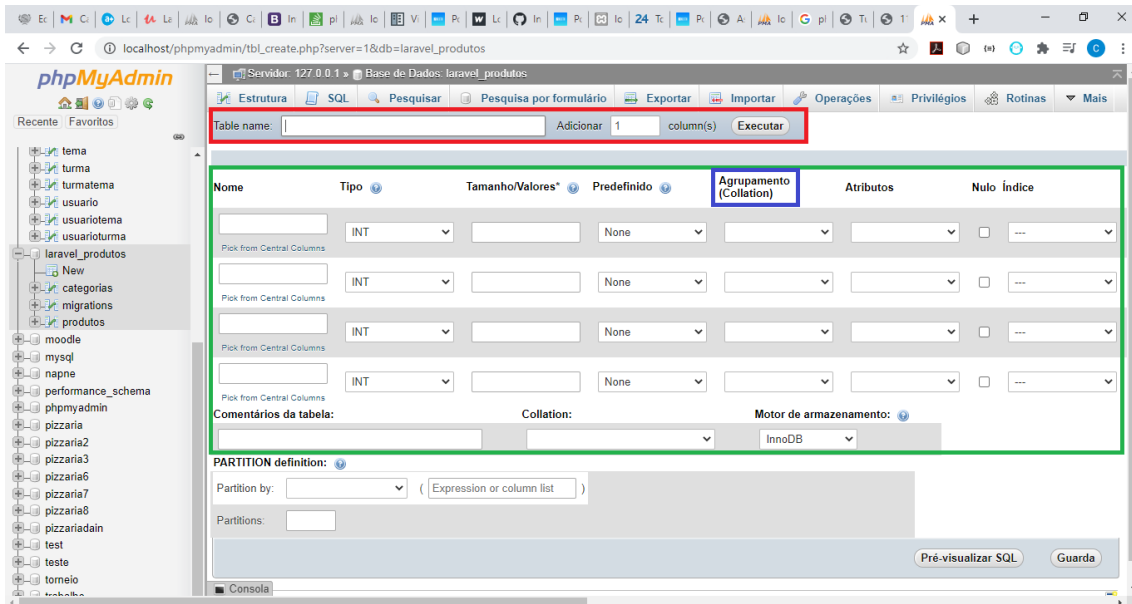


Figura 10 – phpmyadmin- Criação de tabelas

O retângulo com bordas vermelhas ilustra onde o usuário deve escrever o nome da tabela e a quantidade de colunas(campos) que a tabela vai ter. Caso precise de mais colunas, existe a possibilidade de adiciona-las posteriormente. O mesmo serve para o caso da necessidade da utilização de menos colunas. Já a área do retângulo verde é o local onde será definido o nome do campo, o seu tipo, tamanho, valor padrão, collation, atributos modificadores, se o campo pode ser nulo e se será chave.

Vale uma atenção especial, para a coluna agrupamento (collation) destacada em azul. É nessa coluna que será definida o tipo de caractere que será armazenado no banco. Para a língua portuguesa que tem acentuação, recomenda-se o uso do latin1\_general\_ci, mas não é o único tipo que trabalha com acentuação.

#### 12.4. Exercício

Aproveitando os exemplos de PDO, construa um site com as opções de inclusão, alteração, exclusão, consulta e listagem de usuários. Construa a tabela usuário com os seguintes campos:

- Login
- Senha
- Email
- Administrador (Sim ou não)

Utilize a criatividade para a construção da interface do site.