Aqui está um exemplo de conexão PHP com MySQL usando PDO (PHP Data Objects), seguindo boas práticas:

<?php

// Configurações do banco de dados

$host = 'localhost';

$dbname = 'nome\_do\_banco';

$username = 'usuario';

$password = 'senha';

// Opções de configuração do PDO

$options = [

PDO::ATTR\_ERRMODE => PDO::ERRMODE\_EXCEPTION, // Ativa relatórios de erro

PDO::ATTR\_DEFAULT\_FETCH\_MODE => PDO::FETCH\_ASSOC, // Define o fetch mode padrão

PDO::ATTR\_EMULATE\_PREPARES => false, // Desativa prepares emulados para segurança

PDO::MYSQL\_ATTR\_INIT\_COMMAND => 'SET NAMES utf8mb4' // Define charset

];

try {

// Cria a conexão PDO

$pdo = new PDO(

"mysql:host=$host;dbname=$dbname;charset=utf8mb4",

$username,

$password,

$options

);

// Exemplo de query

$stmt = $pdo->query("SELECT \* FROM usuarios");

$resultados = $stmt->fetchAll();

foreach ($resultados as $linha) {

echo $linha['nome'] . "<br>";

}

} catch (PDOException $e) {

// Tratamento de erros

echo "Erro na conexão: " . $e->getMessage();

exit;

}

// Fecha a conexão (opcional, pois o PHP fecha automaticamente)

$pdo = null;

?>

**Principais Características:**

1. **Segurança:**
   * Usa **ERRMODE\_EXCEPTION** para capturar erros
   * Desativa **ATTR\_EMULATE\_PREPARES** para prepared statements reais
   * Define charset explicitamente
2. **Boas Práticas:**
   * Separa configurações de conexão
   * Usa **utf8mb4** para suporte completo a Unicode
   * Fecha a conexão explicitamente
3. **Funcionalidades:**
   * Suporte a prepared statements
   * Fetch de resultados associativos
   * Tratamento de erros robusto

**Exemplo com Prepared Statements:**

// Inserção segura com parâmetros

$nome = "João";

$email = "joao@example.com";

$stmt = $pdo->prepare("INSERT INTO usuarios (nome, email) VALUES (?, ?)");

$stmt->execute([$nome, $email]);

// Com named placeholders

$stmt = $pdo->prepare("INSERT INTO usuarios (nome, email) VALUES (:nome, :email)");

$stmt->execute(['nome' => $nome, 'email' => $email]);

**Recomendações Adicionais:**

1. Armazene credenciais em arquivos de configuração separados (não comitados no versionamento)
2. Use variáveis de ambiente para senhas (ex: usando **getenv()**)
3. Implemente camada de abstração para consultas repetitivas
4. Atualize para PHP 8.1+ para melhor suporte e segurança

Para ambientes de produção, adicione:

// Oculta erros detalhados

ini\_set('display\_errors', 0);

error\_reporting(0);

Estrutura recomendada de diretórios:

projeto/

├── config/

│ └── database.php

├── src/

│ └── models/

│ └── UsuarioModel.php

└── public/

└── index.php