Tema 9 – Conexão com Banco de Dados

Conexão do PHP com MySQL

No PHP, você pode conectar ao MySQL usando **MySQLi** ou **PDO**. No seu projeto, você usa **PDO**, que é mais moderno, seguro e flexível.

1️⃣ Configuração da conexão (tasks.php)

```
$servername = "localhost";
$port = 7306; // porta do MySQL no XAMPP, geralmente 3306 se padrão
$username = "root";
$password = "";
$dbname = "todo_app";

try {
  $conn = new PDO("mysql:host=$servername;port=$port;dbname=$dbname",
$username, $password);
  $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
} catch(PDOException $e) {
  echo "Erro na conexão com o banco de dados: " . $e->getMessage();
}
```

✅ Explicação:

- new PDO(...) cria a conexão com o banco.
- "mysql:host=$servername;port=$port;dbname=$dbname" indica: tipo do banco (MySQL), servidor, porta e banco de dados.
- $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION) faz com que erros gerem exceções, facilitando debug.
- try/catch captura erros caso a conexão falhe.

2️⃣ Operações no banco (CRUD)

Você definiu funções para **Criar, Ler, Atualizar e Excluir** tarefas (tasks.php):

a) Ler tarefas

```
function displayTasks($conn) {
  $sql = "SELECT * FROM tasks";
  $stmt = $conn->prepare($sql);
  $stmt->execute();

  while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $row['description'];
  }
}
```

- prepare() → prepara a query (evita SQL injection).
- execute() → executa a query.
- fetch(PDO::FETCH_ASSOC) → retorna os resultados como array associativo.

b) Criar tarefa

```
function addTask($conn, $description) {
   $sql = "INSERT INTO tasks (description) VALUES (:description)";
   $stmt = $conn->prepare($sql);
   $stmt->bindParam(':description', $description);
   $stmt->execute();
}
```

- :description → parâmetro seguro.
- bindParam() → substitui o parâmetro pela variável.

c) Atualizar tarefa

```
function updateTask($conn, $id, $description) {
   $sql = "UPDATE tasks SET description = :description WHERE id = :id";
   $stmt = $conn->prepare($sql);
   $stmt->bindParam(':id', $id);
   $stmt->bindParam(':description', $description);
   $stmt->execute();
}
```

d) Excluir tarefa

```
function deleteTask($conn, $id) {
   $sql = "DELETE FROM tasks WHERE id = :id";
   $stmt = $conn->prepare($sql);
   $stmt->bindParam(':id', $id);
   $stmt->execute();
}
```

3️⃣ Encerrando a conexão

```
$conn = null;
```

- Libera recursos do servidor. Com PDO, basta atribuir null.

4️⃣ Integração com HTML (index.php)

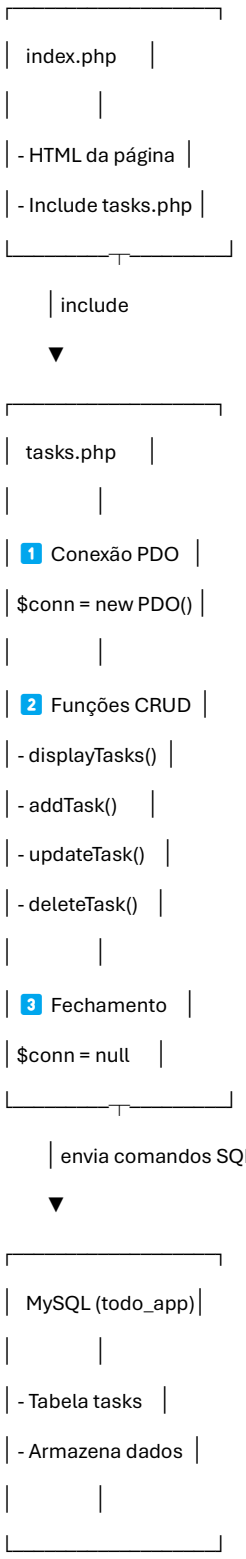
```
<?php include 'tasks.php'; ?>
```

- Inclui o arquivo PHP que conecta ao banco e exibe as tarefas.
- O HTML mantém a estrutura da página, enquanto o PHP faz a lógica do CRUD.

5️⃣ Boas práticas mostradas

- Uso de **PDO** com prepare() → evita SQL injection.
- Separação de lógica (tasks.php) e visual (index.php).
- Tratamento de erros com try/catch.
- Fechamento de conexão ao final.

Diagrama textual simples, mostrando o fluxo de dados entre index.php, tasks.php, PDO e MySQL.



Explicação do fluxo:

1. **index.php** carrega o HTML da página e inclui o arquivo tasks.php.
2. **tasks.php**:

- Cria a **conexão PDO** com MySQL (todo_app).
- Executa as funções **CRUD** conforme ações do usuário (adicionar, atualizar, excluir, listar).
- Fecha a conexão ao final.

3. **MySQL** recebe os comandos SQL (SELECT, INSERT, UPDATE, DELETE) e retorna os dados para o PHP, que os exibe na página.

💡 **Resumo didático visual**:

- **index.php** → interface do usuário
- **tasks.php** → lógica e manipulação do banco
- **PDO** → canal seguro entre PHP e MySQL
- **MySQL** → armazenamento e gerenciamento dos dados
