# Sistema CRUD - PHP PDO MariaDB

Sistema completo de cadastro de usuários desenvolvido em PHP utilizando PDO (PHP Data Objects) para conexão com banco de dados MariaDB/MySQL. O sistema implementa as quatro operações básicas de um CRUD: **Create** (Criar), **Read** (Ler), **Update** (Atualizar) e **Delete** (Excluir).

## 📋 Características

* **Interface moderna e responsiva** com design gradiente
* **Operações CRUD completas** para gerenciamento de usuários
* **Segurança robusta** com PDO e prepared statements
* **Validações** de dados no lado do servidor
* **Mensagens de feedback** para todas as operações
* **Código totalmente comentado** em português
* **Proteção contra SQL Injection** usando prepared statements
* **Proteção contra XSS** usando htmlspecialchars()

## 🛠️ Tecnologias Utilizadas

* **PHP 7.4+** - Linguagem de programação server-side
* **PDO** - PHP Data Objects para acesso ao banco de dados
* **MariaDB/MySQL** - Sistema de gerenciamento de banco de dados
* **HTML5** - Estrutura da página
* **CSS3** - Estilização e layout responsivo
* **JavaScript** - Interatividade e máscaras de entrada

## 📁 Estrutura de Arquivos

crud-php-pdo/

│

├── config.php # Configurações do banco de dados

├── conexao.php # Classe de conexão PDO

├── funcoes.php # Funções auxiliares e validações

├── index.php # Página principal com interface e operações CRUD

├── style.css # Estilos CSS

├── database.sql # Script SQL para criar banco e tabela

└── README.md # Este arquivo de documentação

## 🗄️ Estrutura do Banco de Dados

### Tabela: usuarios

| Campo | Tipo | Descrição |
| --- | --- | --- |
| id | INT | Chave primária auto-incrementável |
| nome | VARCHAR(100) | Nome completo do usuário (obrigatório) |
| email | VARCHAR(100) | E-mail único (obrigatório) |
| telefone | VARCHAR(20) | Número de telefone (opcional) |
| data\_nascimento | DATE | Data de nascimento (opcional) |
| criado\_em | TIMESTAMP | Data/hora de criação do registro |
| atualizado\_em | TIMESTAMP | Data/hora da última atualização |

## 🚀 Instalação e Configuração

### Pré-requisitos

* Servidor web (Apache, Nginx)
* PHP 7.4 ou superior
* MariaDB 10.3+ ou MySQL 5.7+
* Extensão PDO habilitada no PHP

### Passo 1: Criar o Banco de Dados

Execute o arquivo database.sql no seu servidor MariaDB/MySQL:

mysql -u root -p < database.sql

Ou acesse o phpMyAdmin e importe o arquivo database.sql.

### Passo 2: Configurar a Conexão

Edite o arquivo config.php e ajuste as credenciais do banco de dados:

define('DB\_HOST', 'localhost'); // Host do banco

define('DB\_NAME', 'crud\_pdo'); // Nome do banco

define('DB\_USER', 'root'); // Usuário do banco

define('DB\_PASS', ''); // Senha do banco

### Passo 3: Configurar o Servidor Web

**Para Apache:**

Certifique-se de que o módulo mod\_rewrite está habilitado e coloque os arquivos no diretório htdocs ou www.

**Para PHP Built-in Server (desenvolvimento):**

cd crud-php-pdo

php -S localhost:8000

Acesse: http://localhost:8000

### Passo 4: Testar o Sistema

Abra o navegador e acesse o endereço configurado. Você verá a interface do sistema com o formulário de cadastro e a tabela de usuários.

## 📖 Como Usar

### Cadastrar Novo Usuário

1. Preencha os campos do formulário (Nome e E-mail são obrigatórios)
2. Clique no botão **"Cadastrar Usuário"**
3. Uma mensagem de sucesso será exibida

### Editar Usuário

1. Na tabela de usuários, clique no botão **"Editar"** da linha desejada
2. Os dados serão carregados no formulário
3. Altere os campos necessários
4. Clique em **"Atualizar Usuário"**

### Excluir Usuário

1. Na tabela de usuários, clique no botão **"Excluir"** da linha desejada
2. Confirme a exclusão na janela de confirmação
3. O usuário será removido do banco de dados

### Consultar Usuários

Todos os usuários cadastrados são exibidos automaticamente na tabela abaixo do formulário, ordenados do mais recente para o mais antigo.

## 🔒 Segurança

O sistema implementa várias camadas de segurança:

### Prepared Statements

Todas as queries SQL utilizam prepared statements do PDO, protegendo contra **SQL Injection**:

$sql = "INSERT INTO usuarios (nome, email) VALUES (?, ?)";

$stmt = $pdo->prepare($sql);

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

### Proteção XSS

Todos os dados exibidos são filtrados com htmlspecialchars() para prevenir **Cross-Site Scripting**:

echo htmlspecialchars($usuario['nome']);

### Validações

* Validação de formato de e-mail
* Validação de telefone brasileiro
* Validação de data de nascimento
* Verificação de e-mail duplicado
* Sanitização de entradas

## 📝 Funcionalidades dos Arquivos

### config.php

Define as constantes de configuração do banco de dados e configurações gerais do sistema (fuso horário, exibição de erros).

### conexao.php

Implementa a classe Database que gerencia a conexão PDO usando o padrão Singleton, garantindo uma única instância de conexão durante a execução do script.

### funcoes.php

Contém funções auxiliares para:

* Validação de campos (e-mail, telefone, data)
* Sanitização de dados
* Formatação de dados para exibição
* Verificação de duplicidade
* Registro de logs (opcional)

### index.php

Arquivo principal que contém:

* Processamento das operações CRUD
* Interface HTML do formulário
* Tabela de listagem de usuários
* Lógica de edição e exclusão
* JavaScript para máscaras e interatividade

### style.css

Folha de estilos com:

* Design moderno com gradientes
* Layout responsivo
* Animações suaves
* Estilos para formulários e tabelas
* Mensagens de feedback estilizadas

### database.sql

Script SQL que:

* Cria o banco de dados crud\_pdo
* Cria a tabela usuarios com todos os campos
* Insere dados de exemplo para teste

## 🎨 Personalização

### Alterar Cores

Edite o arquivo style.css e modifique as cores do gradiente:

background: linear-gradient(135deg, **#667eea** 0%, **#764ba2** 100%);

### Adicionar Campos

1. Adicione o campo na tabela do banco de dados
2. Adicione o campo no formulário HTML (index.php)
3. Inclua o campo nas queries SQL de INSERT e UPDATE
4. Adicione validações em funcoes.php se necessário

### Alterar Validações

Edite as funções em funcoes.php para ajustar as regras de validação conforme suas necessidades.

## 🐛 Tratamento de Erros

O sistema utiliza blocos try-catch para capturar exceções do PDO:

try {

// Operação no banco de dados

} catch (PDOException $e) {

$mensagem = "Erro: " . $e->getMessage();

$tipo\_mensagem = "erro";

}

Erros comuns tratados:

* E-mail duplicado (UNIQUE constraint)
* Campos obrigatórios vazios
* Formato de dados inválido
* Falha na conexão com o banco

## 📱 Responsividade

O sistema é totalmente responsivo e se adapta a diferentes tamanhos de tela:

* **Desktop**: Layout completo com tabela expandida
* **Tablet**: Layout ajustado com fontes menores
* **Mobile**: Layout otimizado para telas pequenas

## 🔧 Requisitos do Sistema

### Servidor

* Apache 2.4+ ou Nginx 1.18+
* PHP 7.4 ou superior
* MariaDB 10.3+ ou MySQL 5.7+

### Extensões PHP Necessárias

* PDO
* pdo\_mysql
* mbstring
* json

Verifique se as extensões estão habilitadas no php.ini:

extension=pdo

extension=pdo\_mysql

extension=mbstring

## 📊 Fluxo de Operações

### Inserir (CREATE)

Usuário preenche formulário

↓

Dados são enviados via POST

↓

Sistema valida os dados

↓

PDO prepara o statement

↓

Dados são inseridos no banco

↓

Mensagem de sucesso é exibida

### Consultar (READ)

Página é carregada

↓

Sistema executa SELECT

↓

PDO busca todos os registros

↓

Dados são exibidos na tabela

### Atualizar (UPDATE)

Usuário clica em "Editar"

↓

Sistema busca dados do usuário

↓

Formulário é preenchido

↓

Usuário altera os dados

↓

Sistema atualiza o registro

↓

Mensagem de sucesso é exibida

### Excluir (DELETE)

Usuário clica em "Excluir"

↓

JavaScript exibe confirmação

↓

Sistema recebe ID via GET

↓

PDO executa DELETE

↓

Registro é removido

↓

Mensagem de sucesso é exibida

## 🎓 Conceitos Aprendidos

Este projeto demonstra:

* **PDO (PHP Data Objects)**: Interface moderna para acesso a bancos de dados
* **Prepared Statements**: Prevenção de SQL Injection
* **CRUD Operations**: Operações básicas de banco de dados
* **MVC Pattern** (simplificado): Separação de lógica e apresentação
* **Validação de Dados**: Validação server-side
* **Tratamento de Exceções**: Try-catch para erros
* **HTML Semântico**: Estrutura correta de formulários
* **CSS Responsivo**: Design adaptável
* **JavaScript**: Interatividade e validação client-side

## 🤝 Contribuições

Este é um projeto educacional. Sinta-se livre para:

* Adicionar novas funcionalidades
* Melhorar a segurança
* Otimizar o código
* Corrigir bugs
* Melhorar a documentação

## 📄 Licença

Este projeto é de código aberto e está disponível para uso educacional e comercial.

## 📞 Suporte

Para dúvidas ou problemas:

1. Verifique se todas as configurações estão corretas
2. Confira os logs de erro do PHP
3. Teste a conexão com o banco de dados
4. Verifique as permissões dos arquivos

## 🎉 Conclusão

Este sistema CRUD demonstra as melhores práticas de desenvolvimento PHP moderno, utilizando PDO para acesso seguro ao banco de dados, implementando validações robustas e oferecendo uma interface amigável e responsiva.

**Desenvolvido com ❤️ para fins educacionais**

**Versão:** 1.0.0  
**Data:** 2025  
**Linguagem:** PHP 7.4+  
**Banco de Dados:** MariaDB/MySQL