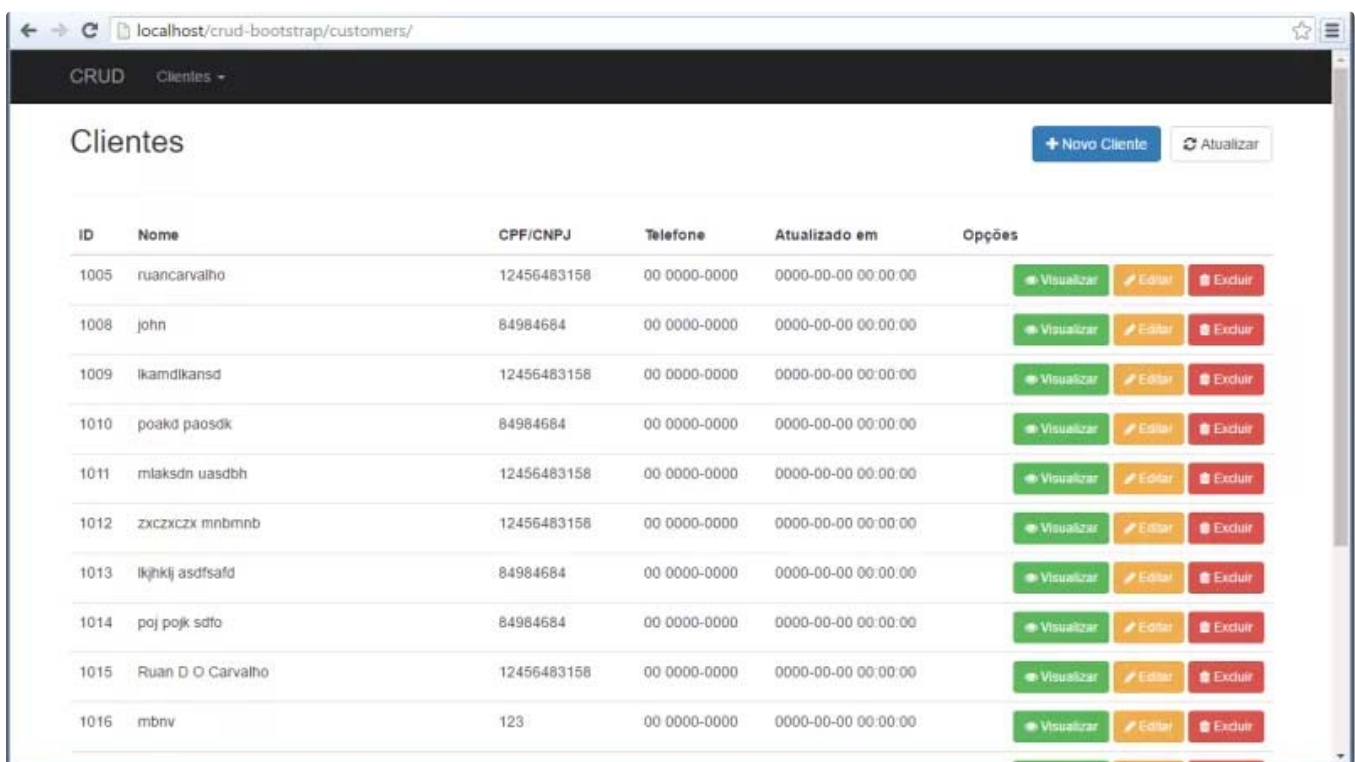


CRUD com Bootstrap, PHP & MySQL – Parte III

Esta é uma série de tutoriais, na qual você vai aprender uma forma eficaz de implementar um sistema de CRUD completo, usando o Bootstrap no front-end e PHP com MySQL no back end.

Agora, neste tutorial, você vai ver como criar os arquivos funções para usar em um módulo de cadastro de clientes. Depois disso, vamos criar uma listagem inicial com os botões de acesso para as funcionalidades do cadastro, como inserção, edição e exclusão de cada registro.



ID	Nome	CPF/CNPJ	Telefone	Atualizado em	Opções
1005	ruancarvalho	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1008	john	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1009	lkamdikansd	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1010	poakd paosdk	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1011	mlaksdn uasdbh	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1012	zxczxczx mnbmnb	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1013	lkjhklj asdfsafo	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1014	poj pojk sdfg	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1015	Ruan D O Carvalho	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1016	mbrv	123	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir

Ao final deste tutorial, você vai ter uma estrutura básica de listagem de clientes para reaproveitar no projeto, na qual o usuário vai poder acessar as funcionalidades de CRUD, do cadastro de clientes.

Antes de Começar



pré-requisitos, e criar o BD.

Passo 1: Crie o Módulo e as Funções

Para começar, crie uma pasta chamada “*customers*”.

Essa pasta será o nosso módulo de clientes, e ela terá todas as funcionalidades relacionadas a este *model*, ou entidade.

Dentro da pasta, crie um arquivo chamado “*functions.php*”. Esse arquivo terá todas as funções das telas de cadastro de clientes. O código desse arquivo fica assim, por enquanto:

```
1  <?php
2
3  require_once('../config.php');
4  require_once(DBAPI);
5
6  $customers = null;
7  $customer = null;
8
9  /**
10   * Listagem de Clientes
11   */
12  function index() {
13      global $customers;
14      $customers = find_all('customers');
15  }
16
```

functions.php hosted with ❤ by GitHub

[view raw](#)

As primeiras linhas fazem a importação do arquivo de configurações e da camada de acesso a dados (DBAPI).

Depois, eu criei duas variáveis globais, para serem usadas entre as funções, e que vão guardar os registros que estiverem sendo usados:

- A variável *\$customers*, irá guardar um conjunto de registros de clientes.
- E a variável *\$customer* guardará um único cliente, para os casos de inserção e atualização (CREATE e UPDATE).

Observe a diferença entre plural e singular nos nomes de variáveis. Em web, você vai usar muito isso: plural para vários, e singular para um único item.

A função *index()* é a função que será chamada na tela principal de clientes, e ela fará a consulta



não existe ainda.

Precisamos implementar essa função no arquivo *database.php*.

Passo 2: Implemente a Consulta no Banco de Dados

Agora, vamos implementar as funções de consulta ao banco de dados. Vamos tentar deixar o mais genérico possível.

Abra o arquivo *database.php*, que está na pasta */inc* do seu projeto.

Crie a função a seguir:

```
<?php

/**
 * Pesquisa um Registro pelo ID em uma Tabela
 */
function find( $table = null, $id = null ) {

    $database = open_database();
    $found = null;

    try {
        if ($id) {
            $sql = "SELECT * FROM " . $table . " WHERE id = " . $id;
            $result = $database->query($sql);

            if ($result->num_rows > 0) {
                $found = $result->fetch_assoc();
            }

        } else {

            $sql = "SELECT * FROM " . $table;
            $result = $database->query($sql);

            if ($result->num_rows > 0) {
                $found = $result->fetch_all(MYSQLI_ASSOC);
            }

        }

        /* Metodo alternativo
        $found = array();

        while ($row = $result->fetch_assoc()) {
            array_push($found, $row);
        }
    }
}
```



```

37         $_SESSION[ 'message' ] = $e->GetMessage();
38         $_SESSION[ 'type' ] = 'danger';
39     }
40
41     close_database($database);
42     return $found;
43 }

```

database.php hosted with ❤ by GitHub

[view raw](#)

Essa função *find* faz uma busca em uma determinada tabela.

Se for passado algum *id*, nos parâmetros, a pesquisa será feita por esse *id*, que é a chave primária da tabela (como definimos no começo).

Se não for passado o *id*, a consulta retornará todos os registros da tabela.

Nos dois casos, a consulta retornará dados associativos (usando o *fetch_assoc* e *MYSQLI_ASSOC*), ou seja, são *arrays* com o nome da coluna e o valor dela. Assim, fica mais fácil na hora de implementar a tela.

Caso aconteça algum problema na consulta e for disparada uma Exceção, nós devemos exibir o que aconteceu em forma de mensagem. Para isso, eu criei duas variáveis de sessão, do PHP, que vão guardar a mensagem da exception, e assim poderemos exibir na tela.

Agora, crie também a seguinte função:

```

1  <?php
2
3  /**
4   * Pesquisa Todos os Registros de uma Tabela
5   */
6  function find_all( $table ) {
7      return find($table);
8  }
9

```

database.php hosted with ❤ by GitHub

[view raw](#)

Essa função é só um *alias* (leia-se “aláias”) para a função *find*, ou seja, uma outra forma mais prática de chamar a função sem precisar do parâmetro.

A função *find_all* retorna todos os registros de uma tabela.

Passo 3: Crie a Listagem dos Registros



clientes.

Implemente a marcação abaixo, nesse arquivo:

```
<?php
    require_once('functions.php');
    index();
?>

<?php include(HEADER_TEMPLATE); ?>

<header>
    <div class="row">
        <div class="col-sm-6">
            <h2>Clientes</h2>
        </div>
        <div class="col-sm-6 text-right h2">
            <a class="btn btn-primary" href="add.php"><i class="fa fa-plus"></i> Novo Cliente</a>
            <a class="btn btn-default" href="index.php"><i class="fa fa-refresh"></i> Atualizar</a>
        </div>
    </div>
</header>

<?php if (!empty($_SESSION['message'])) : ?>
    <div class="alert alert-<?php echo $_SESSION['type']; ?> alert-dismissible" role="alert">
        <button type="button" class="close" data-dismiss="alert" aria-label="Close"><span ari
        <?php echo $_SESSION['message']; ?>
    </div>
    <?php clear_messages(); ?>
<?php endif; ?>

<hr>

<table class="table table-hover">
<thead>
    <tr>
        <th>ID</th>
        <th width="30%">Nome</th>
        <th>CPF/CNPJ</th>
        <th>Telefone</th>
        <th>Atualizado em</th>
        <th>Opções</th>
    </tr>
</thead>
<tbody>
<?php if ($customers) : ?>
```



```
<td><?php echo $customer['cpf_cnpj']; ?></td>
<td>00 0000-0000</td>
<td><?php echo $customer['modified']; ?></td>
<td class="actions text-right">
    <a href="view.php?id=<?php echo $customer['id']; ?>" class="btn btn-sm btn-su
    <a href="edit.php?id=<?php echo $customer['id']; ?>" class="btn btn-sm btn-wa
    <a href="#" class="btn btn-sm btn-danger" data-toggle="modal" data-target="#d
        <i class="fa fa-trash"></i> Excluir
    </a>
</td>
</tr>
<?php endforeach; ?>
<?php else : ?>
    <tr>
        <td colspan="6">Nenhum registro encontrado.</td>
    </tr>
<?php endif; ?>
</tbody>
</table>

<?php include(FOOTER_TEMPLATE); ?>
```

index.php hosted with ❤ by GitHub [view raw](#)

Vamos por partes...

As primeiras linhas fazem a ligação dessa página com o módulo de clientes (pelo arquivo *functions.php*) e chama função *index*, que é o backend desta página.

Depois, coloquei um *include* para trazer o template de header da página, com todos os CSS's e Metatags necessários. Assim, não precisamos reescrever essa parte.

A partir da linha oito (8) começa a marcação da listagem, apenas com um topo simples e algumas opções.

Na linha 20, você pode ver a verificação de mensagens de seção. Isso serve para exibir alguma notificação que tenha sido definida no backend, como mensagens de erro ou de sucesso, por exemplo.

Depois, você pode ver a tabela de registros. Dentro dela tem um loop (usando o *foreach*) que vai pegar cada registro da variável *\$customers* e criar uma linha nessa tabela e exibir os dados. Como usamos os dados de forma associativa (lá na camada de acesso a dados), é possível obter esses dados pelo nome das colunas.

Isso é um padrão em vários frameworks. Também é possível criar objetos dessa forma.



E, no final, coloquei o template de footer, para fechar a página.

Passo 4: Adicione Registros via SQL

Para ver essa tela funcionando, enquanto não temos as telas de inserção, você pode criar registros na tabela “customers” via SQL, usando o PHPMysqlAdmin, por exemplo:

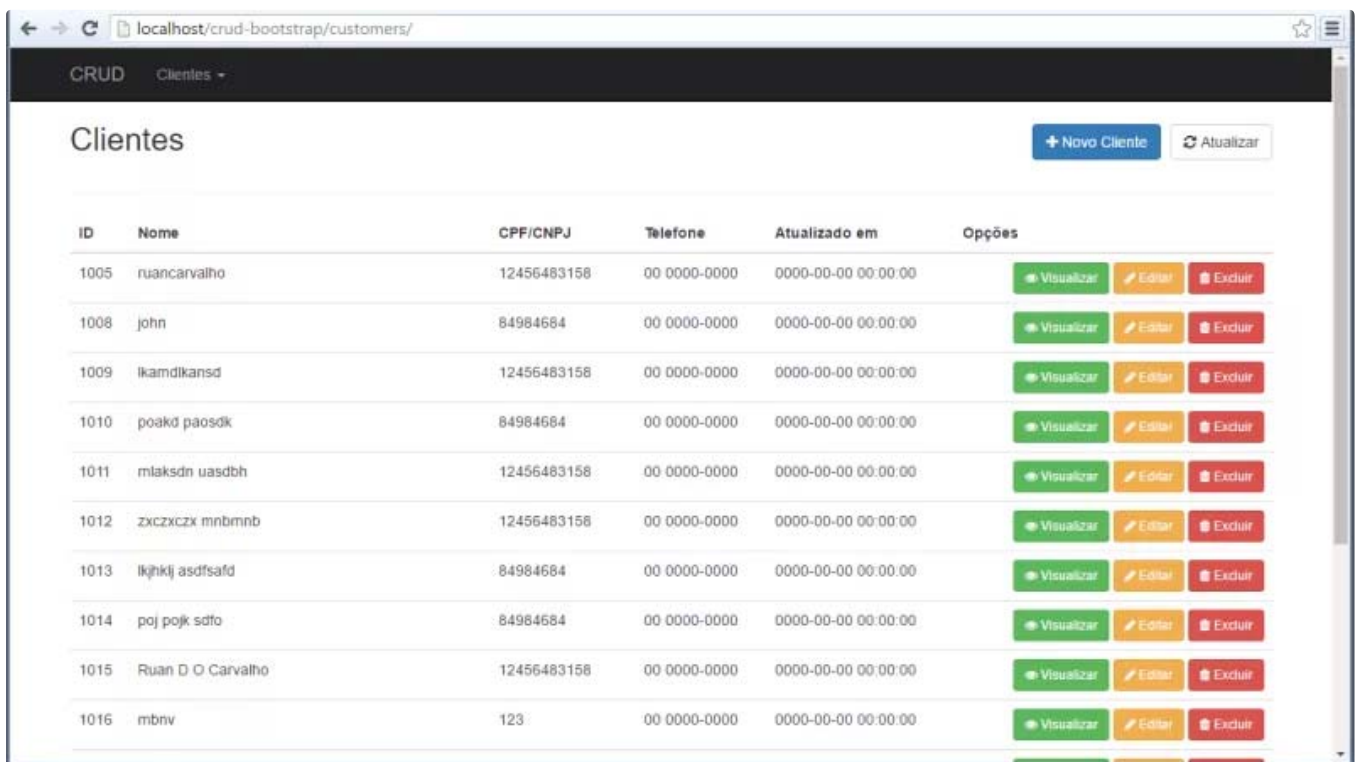
```
1 INSERT INTO `customers` (`id`, `name`, `cpf_cnpj`, `birthdate`, `address`,  
2 `hood`, `zip_code`, `city`, `state`, `phone`, `mobile`, `ie`, `created`, `modified`)  
3 VALUES ('0', 'Fulano de Tal', '123.456.789-00', '1989-01-01', 'Rua da Web, 123',  
4 'Internet', '1234568', 'Teste', 'Teste', '5555555', '55555555', '123456',  
5 '2016-05-24 00:00:00', '2016-05-24 00:00:00');
```

customer.sql hosted with ❤ by GitHub

[view raw](#)

Você pode alterar esse SQL para criar mais registros.

Até aqui, você deve ter uma tela, mais ou menos, como essa:



ID	Nome	CPF/CNPJ	Telefone	Atualizado em	Opções
1005	ruancarvalho	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1008	john	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1009	lkamdikansd	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1010	poakd paosdk	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1011	mlaksdn uasdbh	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1012	zxczxczx mnbmnb	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1013	lkjhklj asdfsafd	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1014	poj pojk sdf	84984684	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1015	Ruan D O Carvalho	12456483158	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir
1016	mbnv	123	00 0000-0000	0000-00-00 00:00:00	Visualizar Editar Excluir

E a estrutura do seu projeto deve estar assim:

- crud-bootstrap-php
 - css
 - customers
 - functions.php



- footer.php
- header.php
- js
- config.php
- index.php

Próximos Passos...

...

Nos próximos tutoriais, você vai precisar adicionar as outras funções no arquivo *functions.php*.

Compartilhe:



Aprenda a Fazer um CRUD com Bootstrap, PHP & MySQL



CRUD com Bootstrap, PHP & MySQL – Parte V



CRUD com Bootstrap, PHP & MySQL – Parte IV

← CRUD com Bootstrap, PHP & MySQL – Parte II

JavaScript: Trabalhando com Funções →

Cadastre seu e-mail para receber as dicas de Desenvolvimento Web da Academy...

Digite seu E-mail

Quero Receber!

