**Entendendo a diferença entre GET e POST**

Vamos agora adicionar um campo de descrição no nosso cadastro de produtos. Um textarea é representado pela tag textarea. Vamos acrescentar esse campo então no formulário:

<form action="adiciona-produto.php">

<table>

<tr>

<td>Nome</td>

<td><input type="text" name="nome" class="form-control" /></td>

</tr>

<tr>

<td>Preço</td>

<td><input type="number" name="preco" class="form-control" /></td>

</tr>

<tr>

<td>Descrição</td>

<td><textarea name="descricao" class="form-control"></textarea>

</tr>

<tr>

<td><input type="submit" value="Cadastrar" /></td>

</tr>

</table>

</form>

Lá na página de adição, precisamos passar esse parâmetro a mais:

$descricao = $\_GET['descricao'];

if(insereProduto($conexao, $nome, $preco, $descricao)) {

// continua...

}

E completar a linha de INSERT lá no nosso banco-produto.php:

function insereProduto($nome, $preco, $descricao) {

$query = "insert into produtos (nome, preco, descricao) values ('{$nome}', {$preco}, '{$descricao}')";

$resultadoDaInsercao = mysqli\_query($conexao, $query);

}

Agora, precisamos criar essa coluna no banco de dados. Vamos lá no PHPMyAdmin, e vamos criar o campo:

alter table produtos add column descricao text;

Se fizermos um SELECT na tabela agora, vemos que a coluna está com valores NULL. Vamos fazer um simples UPDATE para remover eles da nossa tabela:

update produtos set descricao = 'Descricao deste produto';

Falta agora a listagem. Vamos exibir o atributo descrição lá no produto-lista.php:

<tr>

<td><?= $produto['nome'] ?></td>

<td><?= $produto['preco'] ?></td>

<td><?= $produto['descricao'] ?></td>

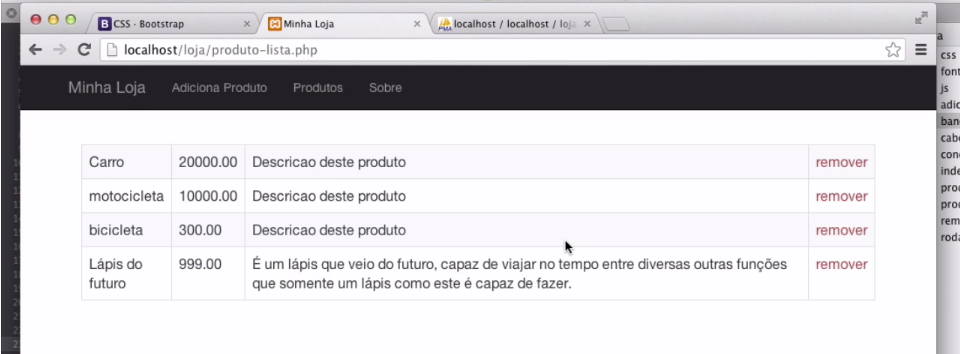
<td>

<a href="remove-produto.php?id=<?= $produto['id']?>" class="text-danger">remover</a>

</td>

</tr>

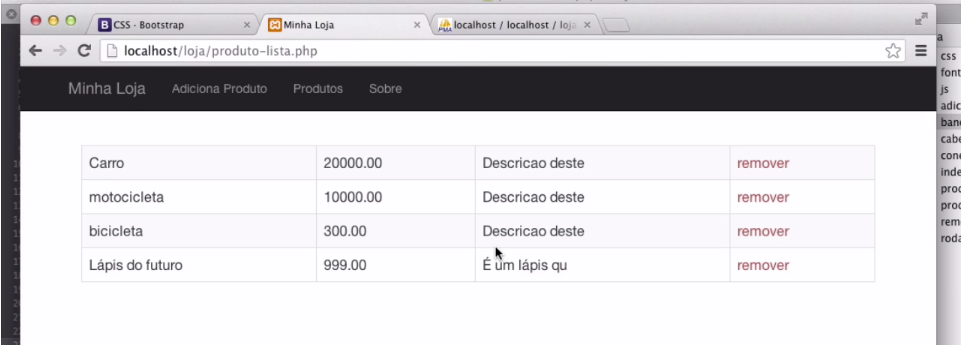
Excelente. O novo campo agora existe, e podemos cadastrar produtos com essa descrição. Agora, um problema de interface: veja só o que acontece se cadastramos um produto com uma descrição muito grande. A tabela fica feia, pois a parte da descrição fica muito grande.



Vamos fazer então com que, na listagem, apareça apenas os 15 primeiros caracteres. Para isso, usaremos a funçãosubstr, que recebe dois parâmetros: a posição do caractere inicial, e a posição do caractere final:

<td><?= substr($produto['descricao'], 0, 15) ?></td>

Veja, agora ficou melhor:



Podemos mudar agora a quantidade de caracteres, pro tamanho que quisermos. Poderíamos colocar 40, por exemplo. Repare que a função é bastante inteligente: se o texto for menor que 40, a função simplesmente vai até o último caractere existente.

O PHP tem muitas outras funções, que nos ajudam em diversas tarefas, como manipulação de strings, e etc. Podemos vê-las no manual do php em [http://www.php.net](http://www.php.net/). O site é bem legal, e tem tanto em português quanto em inglês. Visite o site constantemente, afinal é impossível decorar todas as funções da linguagem.

Temos mais uma mudança importante a fazer. Veja que, sempre que adicionamos um produto, os parâmetros são enviados pela URL: adiciona-produto.php?nome=blabla&preco=blabla&descricao=blabla. O problema disso é que a URL tem um tamanho máximo de dados que podem ser passados por ela. E veja que isso pode ser um problema, já que nosso campo de descrição pode ter tamanho ilimitado!

Podemos resolver isso fácil. Existem duas maneiras de fazer uma requisição. A que estamos fazendo até agora, chama-se GET. Sempre que você digita uma URL no browser, por exemplo, fazemos um GET. O nome desse tipo de requisição já diz o que ela faz: ela "pega" conteúdo. Existe uma outra maneira, mais adequada para quando temos dados a serem enviados, que chamamos de POST. O POST, cujo nome já deixa claro a ideia de "postar" dados, permite o envio de grandes quantidades de dados, sem preocupação.

E para mudar isso é fácil. Basta irmos na declaração do form, e acrescentar o "método":

<form action="adiciona-produto.php" method="post">

Mas agora que não estamos enviando por GET, mas sim por POST, precisamos mudar a maneira de receber esses dados no PHP. Basta mudar o array $\_GET para $\_POST:

$nome = $\_GET['nome'];

$preco = $\_GET['preco'];

$descricao = $\_POST['descricao'];

Se você testar agora, você perceberá que os dados não são mais passados pela URL; eles vão por debaixo dos panos. É assim que o POST funciona. Se você ver a aba Network do Developer Tools, você verá que os dados são enviados por ali; repare nos "headers". É através do POST que enviamos arquivos binários, por exemplo, quando fazemos upload de arquivos.

Usar GET para adicionar ou remover coisas na nossa aplicação não é uma boa ideia. Plugins que aceleram a navegação, ou mesmo hardwares intermediários que rodam pela internet, tem o costume de cachear resultados de links; afinal, a ideia do GET é "pegar dados", um GET não deveria "mudar dados". Ou seja, sempre que você for alterar/adicionar algo, o ideal é que você use POST, e não GET.

Outro exemplo é o próprio Google. É assim que ele alimenta a sua própria base de dados: ele segue os links que acha na sua página. Ou seja, seu GET não pode ter efeitos colaterais, ou seja, não pode mexer no banco de dados.

Já corrigimos o formulário, falta agora corrigir o link de Remove, afinal é um link, que faz um GET, e deleta um produto. Vamos transformar isso em um POST:

<tr>

<td><?= $produto['nome'] ?></td>

<td><?= $produto['preco'] ?></td>

<td><?= $produto['descricao'] ?></td>

<td>

<form action="remove-produto.php?id=<?=$produto['id']?>" method="post">

<button class="btn btn-danger">remover</button>

</form>

</td>

</tr>

Agora lá no remove-produto.php, precisamos trocar a linha que pega o id do produto, e passar a pegar do POST:

$id = $\_POST['id'];

Ainda não funcionou, pois ainda estamos passando o "id" pela URL. Vamos mudar isso, e passar por debaixo dos panos, usando o campo "hidden". Esse campo é um campo escondido, ou seja, não aparece para o usuário, mas é postado para o usuário final:

<tr>

<td><?= $produto['nome'] ?></td>

<td><?= $produto['preco'] ?></td>

<td><?= $produto['descricao'] ?></td>

<td>

<form action="remove-produto.php" method="post">

<input type="hidden" name="id" value="<?=$produto['id']?>" />

<button class="btn btn-danger">remover</button>

</form>

</td>

</tr>

Veja que agora cada botão de remover tem o seu próprio formulário, com um campo "hidden" com o ID.

O POST é a alternativa certa para você enviar/adicionar/remover dados para o servidor. Lembre-se que o GET deve ser apenas para exibir dados. Aqui ainda não discutimos nada de segurança. O POST, apesar dos dados serem enviados por debaixo dos panos, não é "seguro"; ou seja, alguém consegue falsificar um post. Se precisar de segurança, você precisa usar HTTPS. Mas isso, discutiremos mais pra frente.