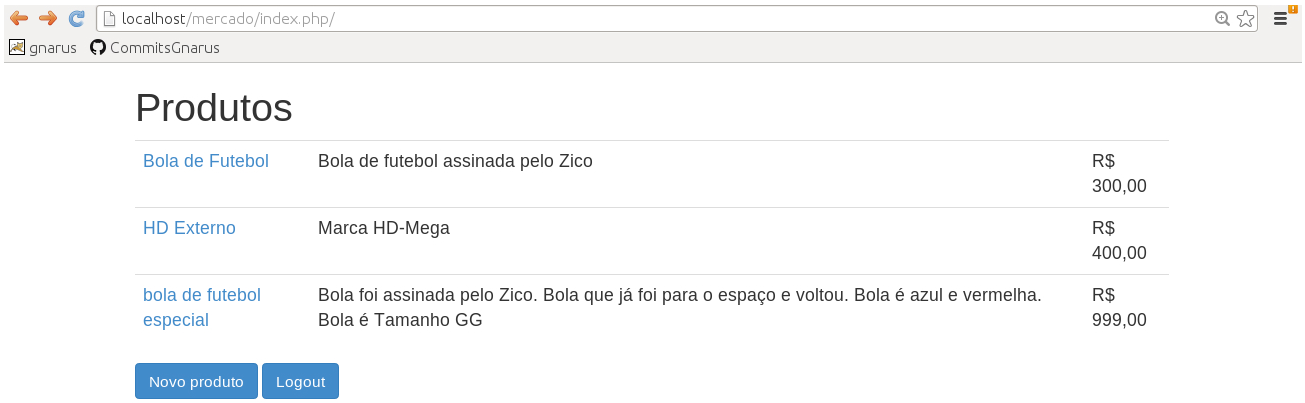
02

Limitando conteúdo html e evitando injection de script

No nosso index, vamos criar uma nova coluna que terá a descrição dos nossos produtos, incluido uma nova `td` na tabela ficará dessa forma:

```
<tr>
    <td><?= anchor("produtos/mostra?id={$produto['id']}", $produto["nome"])?></td>
    <td><?= $produto["descricao"]?> </td>
    <td><?= numeroEmReais($produto["preco"])?></td>
</tr>
```

Fazendo um teste veja como ficou a tela de produtos:



Note como ficou poluída nossa lista! Tem muita informação e por isso ela deixa de ser agrável, temos que deixar a descrição, mas o ideal é que só mostremos alguma parte dela, apenas o começo, os primeiros carateres, e o code igniter já tem um helper de text que faz isso. Para usá-lo basta fazer o load no controller, mas nesse caso é melhor fazer o carregamento em /configs/autoload.php que assim podemos usar em qualquer controller.

```
$autoload['helper'] = array('url', 'form','text');
```

Agora que está carregado podermos usar o `character_limiter()` , passando no parâmetro o conteúdo que queremos limitar, e depois o limite de caracteres.

```
    <tr>
        <td><?= anchor("produtos/mostra?id={$produto['id']}", $produto["nome"])?></td>
        <td><?= character_limiter($produto["descricao"],10)?> </td>
        <td><?= numeroEmReais($produto["preco"])?></td>
    </tr>
```

Com nossa lista mais interessante, podemos ver um detalhe que também não está agradável: quando acessamos os detalhes de algum produto temos o seguinte caminho na url :

`/mercado/index.php/produtos/mostra?id=1` Esse tipo de caminho também não é agradável, poderíamos simplificar isso usando `/produtos/mostra/{id}` . O id é uma *parâmetro* que queremos receber então podemos receber na função mostra, dessa forma podemos remover o get. Veja como ficou:

```
public function mostra($id){
        $this->load->model("produtos_model");
        $produto=$this->produtos_model->busca($id);
        $this->load->helper("typography");
        $dados = array("produto"=>$produto);
        $this->load->view("produtos/mostra", $dados);
    }
```

Agora que nossa url está mais clara, temos que realizar uma alteração no index.php, pois os links da descrição está apontando para a rota errada, temos que alterar para receber o id do produto:

```
<?php foreach($produtos as $produto) : ?>
    <tr>
        <td><?= anchor("produtos/mostra/{$produto['id']}", $produto["nome"])?></td>
        <td><?= character_limiter($produto["descricao"],10)?> </td>
        <td><?= numeroEmReais($produto["preco"])?></td>
    </tr>
<?php endforeach ?>
```

Fazendo o teste veja com os links estão ok agora, se olharmos para o link da descrição podemos ver que é possivel melhorar mais ainda, Se fosse possivel acessar pelo caminho `/mercado/produtos/{id}` ? ficaria mais claro ainda. Podemos customizar a nossa rota, deixando ela em um caminho específico, vamos configurar essa rota no code igniter, acessando o arquivo config/routes.php podemos definir novas rotas para as rotas padrão, no nosso exemplo ficaria dessa o nosso routes.php:

```
$route['produtos/1'] = 'produtos/mostra/1';
```

Acessando /mercado/produtos/1 vimos que funciona! Mas o que acontece se acessarmos algum produto de id diferente ? Temos que adicionar uma rota para cada produto? O ideal seria generalizar uma rota utilizando um parâmetro, e isso é possivel :

```
$route['produtos/(:num)'] = 'produtos/mostra/$1';
```

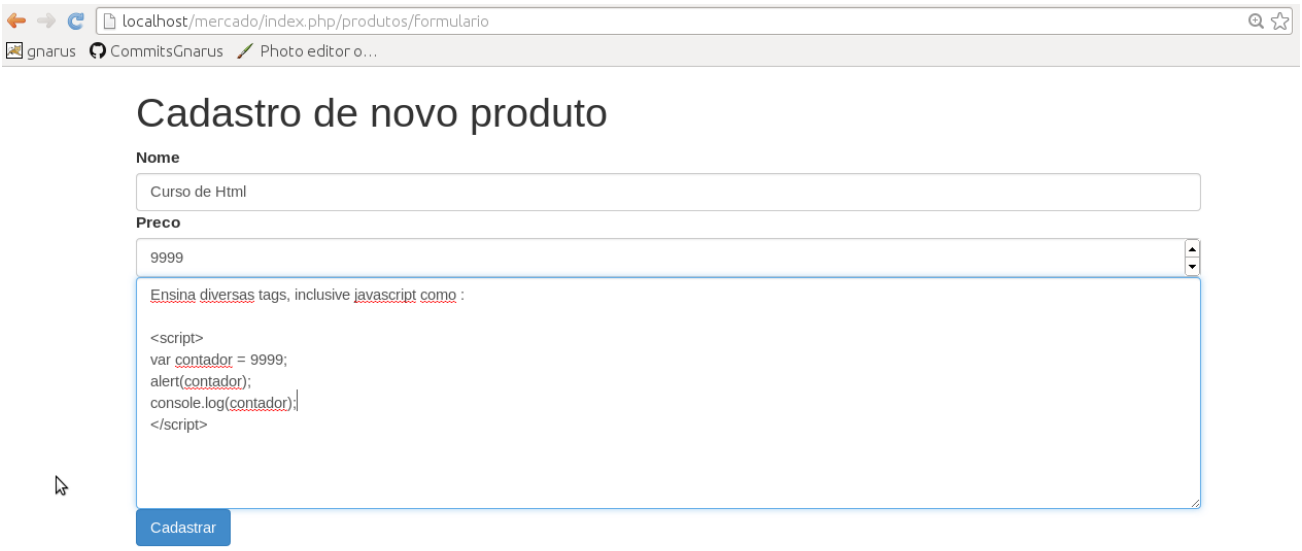
Onde o `(:num )` representa um valor e o `$1` o parâmetro que será recebido, é possivel receber mais números dependendo da situação : `$route['produtos/(:num)/(:num)'] = 'produtos/mostra/$1/$2';` respeitando o total de valores que deve ser igual a quantidade de parâmetros.

Agora temos que alterar o link que a acessa descrição dos produtos no index.php novamente, pois nele ainda chama o caminho que contém o *mostra*, basta retira-lo:

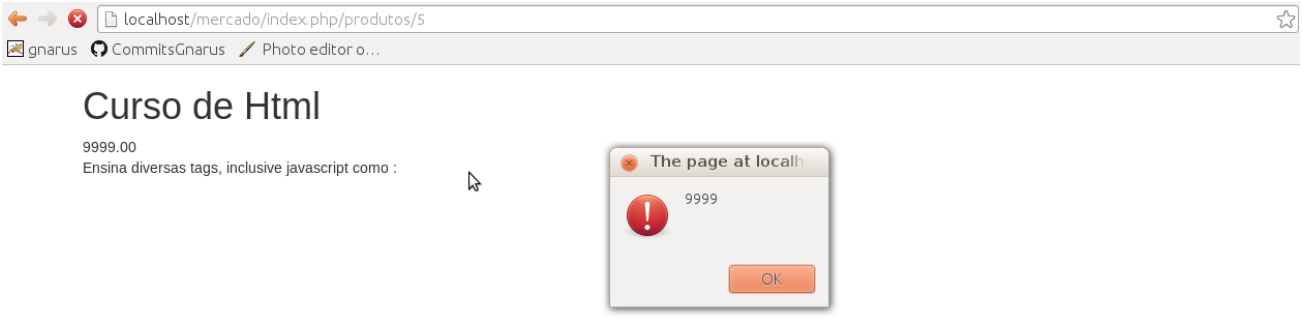
```
<tr>
<td><?= anchor("produtos/{$produto['id']}", $produto["nome"])?></td>
</tr>
```

Podemos usar o routes.php para passar qualquer valor na rota, seja números usando o `(:num)` ou o qualquer caractere usando o `(:any)` , entre outros que pode ser visto na documentação do code igniter.

Navegando pelo nosso sistema, vamos criar um novo curso, será um de html, terá um valor qualquer e na descrição, vamos dizer que será vistos as tags html e também será visto javascript junto com um exemplo, deixando mais ou menos dessa forma :



Após salvar o nosso curso vamos acessá-lo e veja que o nosso sistema executou o código exemplo da descrição!



E se fosse um código malicioso na descrição para coletar alguma informação de outros usuários?

Veja que pela opção de exibir código fonte do navegador é possível ver exatamente o mesmo código que cadastramos. Além disso o sistema também está interpretando *tags html*. O code iginiter tem um helper que faz escapar esses tipos de tags e ele se chama `html_escape` , basta adiciona-lo no nosso mostra.php:

```
<?= auto_typography(html_escape($produto["descricao"])) ?><br>
```

Lembrando que em todo lugar que mostra alguma informação que foi salva pelo usuário é necessário usar o `html_escape` para garantir a nossa segurança do nossos sistema.