02

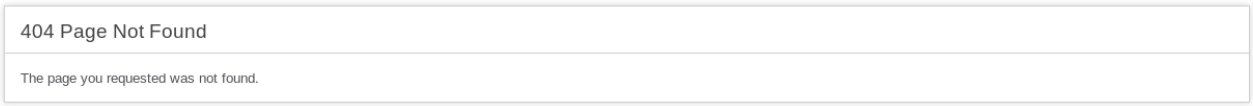
# Mostrando um produto e a tipografia

Continuando o nosso curso de Code Igniter, vamos fazer uma página especifica para cada produto, será acessivel através de um link com o nome do produto que está na lista.

No nosso `index.php` da pasta produtos que está na view temos um `for` com o nome do produto e seu valor, vamos remover o nome e no seu lugar vamos deixar o link para acessar a página. Já sabemos como usar o link usando help `anchor`, direcionando para o caminho `produtos/mostra` e o texto será o nome do produto, então:

```
<h1>Produtos</h1>
        <table class="table">
        <?php foreach($produtos as $produto) : ?>
            <tr>
                <td><?= anchor("produtos/mostra", $produto["nome"])?></td>
                <td><?= numeroEmReais($produto["preco"])?></td>
            </tr>
        <?php endforeach ?>
        </table>
```

Atualizando a nossa página vimos que já aparece o nome do produto em forma de link, mas ao clicarmos somos direcionados para uma página que não existe, pois ainda não criamos o arquivo `mostra.php` dentro da view de produtos.




Vamos criar a nossa função dentro do controller de produtos, ela se chama mostra, e essa função vai trazer do bancos as informações do produto. Para fazer isso temos que carregar o nosso `produtos_model` e fazer fazer ele buscar apenas 1 item, então podemos buscar o id do produto, mas como fazer essa página receber um id e ele fazer a buscar?

Podemos passar o id do produto quando é clicado na lista, só adicionar o id no anchor:

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

Fazendo um teste vimos que agora o id é enviado para página


Podemos fazer nossa função *mostra* buscar as informações do produto a partir do id, depois carregar o modelo *produto_model* e finalmente fazer a fazer a busca.

```
public function mostra(){
        $id = $this->input->get("id");
        $this->load->model("produtos_model");
        $produto = $this->produtos_model->busca($id);
    }
```

Agora vamos carregar todas as informações na view, essa função receberá o caminho da view e um array que deve ter as informações do produto:

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

Temos que fazer a função busca dentro de `produto_model` , ele recebe um id e fará a consulta, mas desta vez faremos uma variação diferente do get, que já tem o `where` da nossa cláusula. No parâmetro é passado o nome da tabela, que no caso é produtos. Passamos um conjunto que seria a condições que será validado, no nosso caso será o id, veja como ficou :

```
public function busca($id) {
    return $this->db
->get_where("produtos", array("id" => $id))
->row_array();
}
```

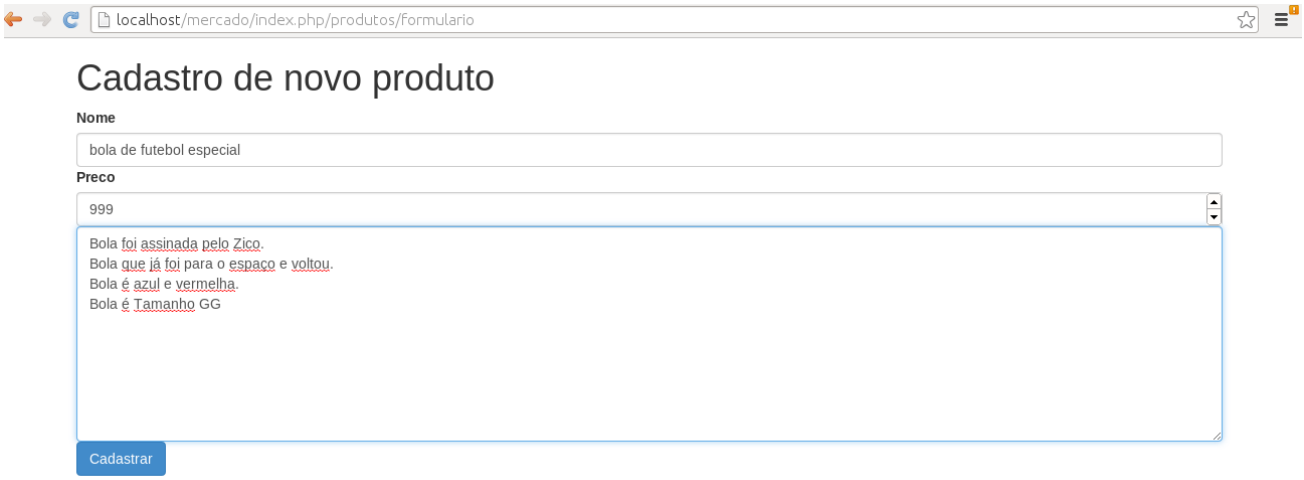
Lembrando que a linha `row_array()` foi usado no lugar do `result_array()` pois queremos apenas um único produto.

Revisando o que fizemos: Temos o controller, que pega o id e chama o model que faz a query para depois enviar para a view, para terminar temos que a view mostra.php em produtos mostra as informações do nosso produto:

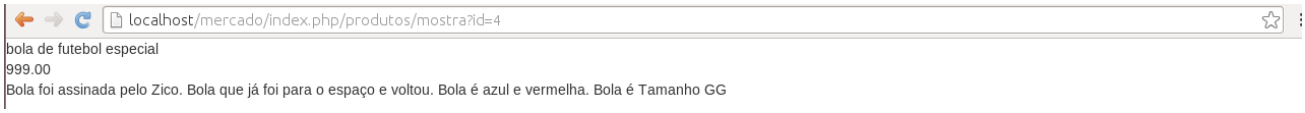
```
<html>
    <head>
            <link rel="stylesheet" href="<?= base_url("css/bootstrap.css") ?>">
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>

    <body>
        <?= $produto["nome"] ?><br>
        <?= $produto["preco"] ?><br>
        <?= $produto["descricao"] ?><br>
    </body>
</html>
```


Agora com nossa nova tela funcionando, podemos fazer um teste, logando no sistema e adicionando um novo produto. Nesse produto colocamos um nome e preço, mas nos detalhes, vamos fazer várias linhas com as informações:




Fazendo o cadastro consultamos esse produto:




Veja que temos um problema: todas as linhas que nós quebramos com a tecla *Enter* que é o caractere \n (que representa a quebra de linha) ficaram estranhas. O html não reconhece a quebra de linha através desse caractere pois ele já tem a tag que tem essa função.

Temos que fazer uma substituição do caractere pela tag, assim o html irá reconhecer quando ele deve realizar a quebra. O code igniter já tem um helper que faz isso, ele se chama `typography` , basta carregá-lo no controller de produtos:

```
public function mostra(){
    $id=$this->input->get("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);
}
```

E depois vamos chamá-lo na view mostra.php:

```
<html>
    <head>
            <link rel="stylesheet" href="<?= base_url("css/bootstrap.css") ?>">
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>

    <body>
```

```
        <?= $produto["nome"] ?><br>
        <?= $produto["preco"] ?><br>
        <?= auto_typography($produto["descricao"])?><br>
    </body>
</html>
```

Tudo certo! Agora só temos que deixar o mostra.php mais bonito, vamos adicionar uma div com a class container e o nome do produto em h1:

```
<html>
    <head>
            <link rel="stylesheet" href="<?= base_url("css/bootstrap.css") ?>">
            <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    </head>

    <body>
        <div class="container">
        <h1><?= $produto["nome"] ?><br></h1>
        <?= $produto["preco"] ?><br>
        <?= auto_typography($produto["descricao"]) ?><br>
        </div>
    </body>
</html>
```