02

Validação

Até agora já aprendemos um pouco de rotas e proteger o nosso código de algum código malicioso, temos que garantir que alguns dados não sejam salvos de maneira incorreta, vamos *validar* as informações antes de salvar no banco de dados.

Vamos garantir que o preenchimento do campo nome seja obrigatório e o campo descrição tenha no minimo 10 caracteres na tela de cadastro. Então temos que alterar a nossa lógica que faz o cadastro, o produtos.php da pasta controller tem uma função que se chama `novo` , que pega o usuário, cria um array que representa um produto, carrega o produto_model e depois salva no banco, se der tudo certo mostra a mensagem de sucesso e depois é redirecionado para o index.php.

Primeiramente vamos carregar a biblioteca de validação:

```
$this->load->library("form_validation");
```

Depois temos que rodar essa validação, ele nos retornará se a submissão do formulário foi um sucesso ou não, se baseando no tipo validação que queremos então vamos gravar esse retorno numa variável chamada `$sucesso` :

```
$sucesso = $this->form_validation->run();
```

Agora temos que aplicar a lógica, Se caso a submissão for um sucesso será executado todo o conteúdo que já existe na function. Se na submissão do formulário houver algum problema ele será redirecionado para a página do formulário de cadastro. Veja como ficou:

```
public function novo() {
    $this->load->library("form_validation");
    $sucesso = $this->form_validation->run();
    if ($sucesso) {
        $usuarioLogado = $this->session->userdata("usuario_logado");
        $produto = array(
            "nome" => $this->input->post("nome"),
            "descricao" => $this->input->post("descricao"),
            "preco" => $this->input->post("preco"),
            "usuario_id" => $usuarioLogado["id"]S
        );
        $this->load->model("produtos_model");
        $this->produtos_model->salva($produto);
        $this->session->set_flashdata("success", "Produto salvo com sucesso");
        redirect("/");
    }else {
        $this->load->view("produtos/formulario");
        }
    }
```

O que falta é mostrar para `form_validation` quais são as regras necessárias para a submissão ser um sucesso. temos que adicionar a regra dentro da function.

```
$this->form_validation->set_rules("nome", "nome", "required");
```

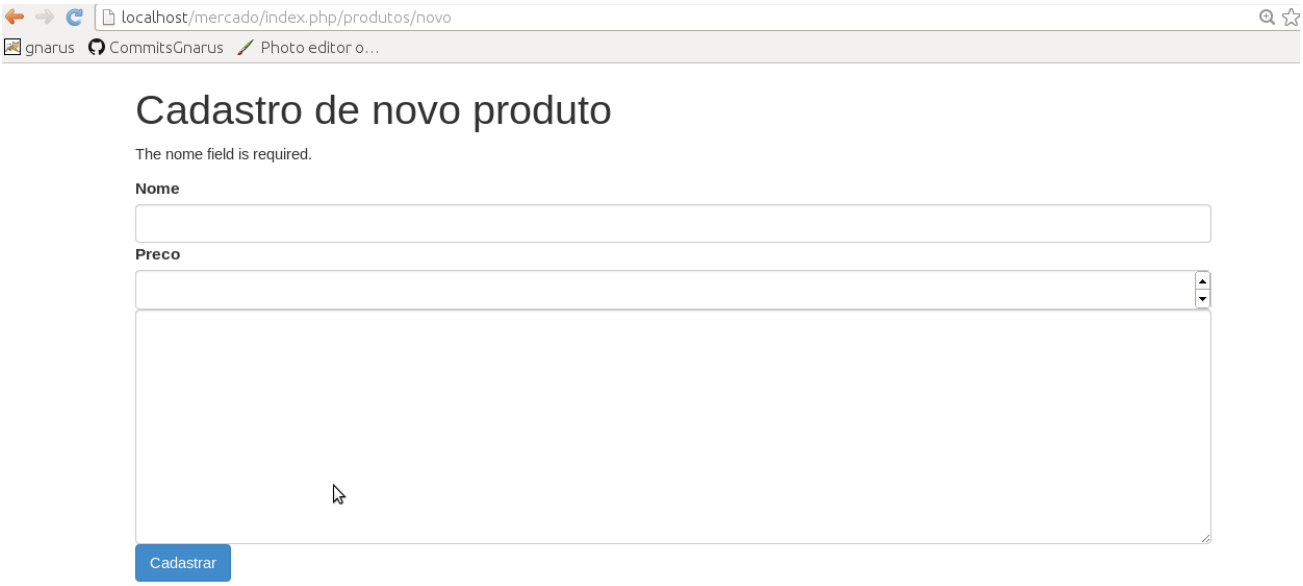
Onde o primeiro nome se trata do campo nome, o segundo é o label, e o terceiro é a regra que no nosso cado é `required` pois é obrigatório.

Fazendo o teste cadastrando um produto em branco veja que está validando, pois está voltando para a página de formulário só que está faltando algo não? A mensagem de erro! Temos que avisar o usuário que houve um falha, deixando a mensagem na tela de cadastro de produtos, ou seja no formulario.php da view de produtos e ela ficará abaixo do título do formulário.

```
<h1>Cadastro de novo produto</h1>

<?= validation_errors() ?>
//resto do código
```

Agora a mensagem :



Veja que agora está funcionando, para deixar a mensagem mais clara , vamos mudar a fonte dele e usar a tag `<p>` , podemo usar o próprio validation_errors para isso :

```
<?= validation_errors("<p class='alert alert-danger'>","</p>") ?>
```

O que foi incluído antes da virgula, vai ficar antes da mensagem de alerta e o que ficará depois da virgula ficará depois da mensagem. Assim temos uma mensagem mais elegante.

Para terminar nossas validações da tela de cadastro, precisamos incluir uma regra na descrição que valide a quantidade de caracteres com o minimo de 10. Para isso basta adicionar a regra no controller, informando a regra que precisamos:

```
$this->form_validation->set_rules("descricao", "descricao", "required|min_length[10]");
```

Veja que foi inserido o `required | min_length[10]` pois temos que informar que é obrigação a quantidade mínima de caracteres, podemos ainda fazer da seguinte forma : `trim | required | min_length[10]` para não contar os espaços. Vamos incluir isso para a descrição e delimitar o mínimo de caracteres de 5 para o nome do produto.

```
$this->form_validation->set_rules("nome", "nome", "required|min_length[5]");
$this->form_validation->set_rules("descricao", "descricao", "required|min_length[10]");
```

Agora que validação está ok, podemos melhorar a usabilidade fazendo que sempre ocorra a falha na validação na tela as informações que já preenchemos não se perca. Colocando um valor que veio do formulário nos três campos do cadastro, usando `"value" => set_value("nome","")` para incluir o valor, e no parâmetro é colocado o nome e seu valor default, veja como ficou os campos:

```
<?php
echo form_open("produtos/novo");

echo form_label("Nome", "nome");
echo form_input(array(
    "name" => "nome",
    "class" => "form-control",
    "id" => "nome",
    "maxlength" => "255",
    "value" => set_value("nome","")
));

echo form_label("Preco", "preco");
echo form_input(array(
    "name" => "preco",
    "class" => "form-control",
    "id" => "preco",
    "maxlength" => "255",
    "type" => "number",
    "value" => set_value("preco","")
));

echo form_textarea(array(
    "name" => "descricao",
    "class" => "form-control",
    "id" => "descricao",
    "value" => set_value("descricao","")
));
```

Após alteração podemos fazer o teste, veja que apenas os campos e nome e descrição está mantendo as informações, isto ocorre pois não temos regra de validação para o campo preço, então temos que aplicar a regra no nosso controller produtos dizendo que o preço é obrigatório.

```
$this->form_validation->set_rules("preco","preco", "required");
```

Completando, vamos mostrar as mensagens de validação bem próximo ao campo informando o tipo de validação:

```
echo form_label("Nome", "nome");
echo form_input(array(
```

```
    "name" => "nome",
    "class" => "form-control",
    "id" => "nome",
    "maxlength" => "255",
    "value" => set_value("nome","")
));

echo form_error("nome");



echo form_label("Preco", "preco");
echo form_input(array(
    "name" => "preco",
    "class" => "form-control",
    "id" => "preco",
    "maxlength" => "255",
    "type" => "number",
    "value" => set_value("preco","")
));

echo form_error("preco");



echo form_textarea(array(
    "name" => "descricao",
    "class" => "form-control",
    "id" => "descricao",
    "value" => set_value("descricao","")
));

echo form_error("descricao");
```

Assim podemos remover a linha que já mostrava as mensagens antes:

```
<?= validation_errors("<p class='alert alert-danger'>","</p>") ?>
```

Fazendo teste veja que as mensagens estão próximas aos seus respectivos campos. Só falta incluir o `<p class='alert alert-danger'> ,</p>` temos que ficar escrevendo em toda mensagem de validação? Não! podemos mandar a mensagem de validação no controller, assim não precisamos ficar repetindo código.

```
$this->form_validation->set_error_delimiters("<p class='alert alert-danger', </p>");
```

Podemos também chama-lo em um helper, para não ficar repetindo novamente o código novamente.