02

Vendendo produtos e formatação de datas no mysql

Para permitir a venda precisamos fazer agora com que ao mostrar um produto (no `mostra.php`) seja visualizado um formulário através do qual o usuário possa efetuar sua compra.

Dentro deste arquivo colocamos um formulário como já vimos antes:

```
<h2>Compre agora mesmo!</h2>
<?php
echo form_open("vendas/nova");


echo form_close();

?>
```

Precisamos de um campo para escolher a data de entrega:

```
echo form_label("Data de entrega" , "data_de_entrega");
echo form_input(array(
    "name" => "data_de_entrega",
    "class" => "form-control",
    "id" => "data_de_entrega",
    "maxlength" => "255",
    "value" => ""
));
```

Adicionamos também um botão para envio:

```
echo form_button(array(
    "class" => "btn btn-primary",
    "content" => "Comprar",
    "type" => "submit"
));
```

Vamos agora criar o controller `Vendas` e a função `nova`:

```
<?php if ( ! defined('BASEPATH')) exit('No direct script access allowed');

class Vendas extends CI_Controller {
    public function nova() {
    }
}
```

Primeiro carregamos um modelo relativo a vendas, então `vendas_model`:

```
        $this->load->model(array("vendas_model"));
```

Criamos agora nossa venda, com os campos que serão necessários:

```
$venda = array(
    "produto_id" => qual é ele?
    "comprador_id" => quem é ele?
    "data_de_entrega" => $this->input->post("data_de_entrega")
);
```

Agora qual o produto que estamos comprando mesmo? Temos que passar para a venda o produto sendo comprado. No formulário adicionamos este campo hidden, escondido:

```
echo form_hidden("produto_id", $produto["id"]);
```

E agora podemos ler este campo:

```
$venda = array(
    "produto_id" => $this->input->post("produto_id"),
    "comprador_id" => quem é ele?
    "data_de_entrega" => $this->input->post("data_de_entrega")
);
```

Falta o usuário comprador, que é quem está logado. Sabemos acessar este id do usuário através da seção:

```
$usuario = $this->session->userdata("usuario_logado");

$venda = array(
    "produto_id" => $this->input->post("produto_id"),
    "comprador_id" => $usuario["id"],
    "data_de_entrega" => $this->input->post("data_de_entrega")
);
```

E pedimos para salvar:

```
$this->vendas_model->salva($venda);
```

Agora posso redirecionar para a página de sucesso:

```
$this->session->set_flashdata("success", "Pedido de compra efetuado com sucesso");
redirect("/");
```

Mas faltou ainda criarmos nosso modelo de acesso ao banco, que recebe e salva a venda:

```
<?php
class Vendas_model extends CI_Model {
    public function salva($venda) {
        $this->db->insert("vendas", $venda);
```

```
    }
}
```

Testamos agora o processo da compra com data `08/09/2020` e sucesso. Conferimos no phpmyadmin e ela foi inserida no banco, mas repara que o campo data está estranho, ele ficou vazio, zerado. Acontece que o formato brasileiro de data que colocamos para inserir não é o mesmo formato que o mysql usa. Criamos então uma função de ajuda, um helper, que transforma uma data na outra: um arquivo chamado `date_helper.php` dentro do diretório `helpers` :

```
<?php
function dataPtBrParaMysql($dataPtBr) {
}
```

Queremos agora quebrar essa nossa data nas duas barras:

```
<?php
function dataPtBrParaMysql($dataPtBr) {
    $partes = explode("/", $dataPtBr);
}
```

E desejamos conectá-las usando hífens, como `ano-mes-dia` , no formato do mysql:

```
<?php
function dataPtBrParaMysql($dataPtBr) {
    $partes = explode("/", $dataPtBr);
    return "{$partes[2]}-{$partes[1]}-{$partes[0]}";
}
```

Agora usamos nosso helper dentro do controller que prepara a venda:

```
        $this->load->helper(array("date"));

        $venda = array(
            "produto_id" => $this->input->post("produto_id"),
            "comprador_id" => $usuario["id"],
            "data_de_entrega" =>
                dataPtBrParaMysql($this->input->post("data_de_entrega"))
        );
```

Testamos agora e a aplicação salva a data no formato adequado, suportando o mysql.