02

Join de tabelas e minhas vendas

Para não listar os produtos que já foram vendidos, no `produtos_model` adicionamos a cláusula de `where` vendido `false`:

```
        $this->db->where("vendido", false);
```

Agora como vendedor quero saber quais produtos **eu** vendi, assim posso agendar minhas entregas. Criamos então uma nova página de venda, portanto no `Vendas`, chamada `index`:

```
    public function index() {
    }
```

São as minhas vendas portanto preciso primeiro do meu usuário logado:

```
    public function index() {
        $usuario = $this->session->userdata("usuario_logado");
    }
```

Preciso agora buscar as vendas deste usuário, no produtos model:

```
        $usuario = $this->session->userdata("usuario_logado");
        $this->load->model("produtos_model");
        $produtosVendidos = $this->produtos_model->buscaVendidos($usuario);
```

Colocamos ele na lista de dados e redirecionamos para a página:

```
    public function index() {
        $usuario = $this->session->userdata("usuario_logado");
        $this->load->model("produtos_model");
        $produtosVendidos = $this->produtos_model->buscaVendidos($usuario);
        $dados = array("produtosVendidos" => $produtosVendidos);
        $this->load->view("vendas/index", $dados);
    }
```

Faltou criar a query no produtos model:

```
    public function buscaVendidos($usuario) {
    }
```

Quero o id do usuário:

```
    public function buscaVendidos($usuario) {
        $id = $usuario["id"];
```

```
    }
```

Sabemos buscar pelo id do usuário e se ele já está vendido, basta dizer que queremos todos os resultados, não só um:

```
    public function buscaVendidos($usuario) {
        $id = $usuario["id"];
        $this->db->where("vendido", true);
        $this->db->where("usuario_id", $id);
        return $this->db->get("produtos")->result_array();
    }
```

Criamos agora o arquivo `index.php` no diretório `vendas`, com o cabeçalho de bootstrap e uma tabela com todos os produtos, mostrando somente seu nome:

```
<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>Minhas vendas</h1>
        <table class="table">
        <?php foreach($produtosVendidos as $produto) : ?>
            <tr>
                <td><?= $produto["nome"]?></td>
            </tr>
        <?php endforeach ?>
        </table>
    </div>
</body>
</html>
```

Logado com um usuário que já vendeu um produto, consigo ver a lista acessando `/mercado/vendas/index` (ou `/mercado/vendas`). Os produtos estão lá, mas falta a data de entrega:

```
                <td><?= $produto["nome"]?></td>
                <td><?= $produto["data_de_entrega"] ?></td>
```

Quando rodamos ele indica que não tem o campo `data_de_entrega`! Na tabela produtos realmente não existe este campo! Quem tem `data_de_entrega` é a tabela `vendas` ... quando faço meu `select` tenho que selecionar as duas tabelas ao mesmo tempo, fazendo um `join`. Isto é, vou alterar meu método `buscaVendidos`:

```
    public function buscaVendidos($usuario) {
        $id = $usuario["id"];
        $this->db->where("vendido", true);
        $this->db->where("usuario_id", $id);
        return $this->db->get("produtos")->result_array();
    }
```

Primeiro, falarei de onde quero selecionar antes da última chamada:

```
public function buscaVendidos($usuario) {
    $id = $usuario["id"];
    $this->db->from("produtos");
    $this->db->where("vendido", true);
    $this->db->where("usuario_id", $id);
    return $this->db->get()->result_array();
}
```

Agora falo qual o join que quero fazer:

```
public function buscaVendidos($usuario) {
    $id = $usuario["id"];
    $this->db->from("produtos");
    $this->db->join("vendas", "vendas.produto_id = produtos.id");
    $this->db->where("vendido", true);
    $this->db->where("usuario_id", $id);
    return $this->db->get()->result_array();
}
```

Por fim, desejo os campos do produtos e um único campo da tabela vendas:

```
public function buscaVendidos($usuario) {
    $id = $usuario["id"];
    $this->db->select("produtos.*, vendas.data_de_entrega");
    $this->db->from("produtos");
    $this->db->join("vendas", "vendas.produto_id = produtos.id");
    $this->db->where("vendido", true);
    $this->db->where("usuario_id", $id);
    return $this->db->get()->result_array();
}
```

Testamos novamente a aplicação e agora sim o campo `data_de_entrega` vem corretamente da tabela `vendas` . Mas o formato dela ainda está estranho, no formato do mysql, precisamos transformar do formato mysql para o formato pt-br. Vamos no nosso helper e criamos a variação do que já existia:

```
function dataMysqlParaPtBr($dataMysql) {
}
```

Como já temos em um formato que o php e o CodeIgniter entendem, podemos instanciar um objeto que representa uma data:

```
function dataMysqlParaPtBr($dataMysql) {
    $data = new DateTime($dataMysql);
```

E agora mandamos a data ser formatada no padrão que estamos interessados:

```
function dataMysqlParaPtBr($dataMysql) {
    $data = new DateTime($dataMysql);
    return $data->format("d/m/Y");
}
```

Agora devemos chamar esta função antes de mostrar a data:

```
        <?php foreach($produtosVendidos as $produto) : ?>
            <tr>
                <td><?= $produto["nome"]?></td>
                <td><?= dataMysqlParaPtBr($produto["data_de_entrega"]) ?></td>
            </tr>
        <?php endforeach ?>
```

Precisamos adicionar o helper de `date`, para evitar colocar em toda página, colocaremos no autoload:

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

Vimos como formatar datas de para o padrão brasileiro e como trabalhar com queries mais complexas com a API do Code Igniter, usando joins e selects detalhados.