CAPÍTULO 10

PHP: parâmetros e bancos de dados

"Antes do software ser reusável, ele precisa ser usável."
— Ralph Johnson

10.1 - SUBMISSÃO DO FORMULÁRIO

Todo formulário criado no HTML tem seus dados enviados para o servidor quando submetido. Cada campo do formulário é enviado como **parâmetro na requisição** feita ao servidor.

No formulário, podemos indicar que página (URL) vai receber os dados preenchidos. É só especificar o atributo `action`. No nosso exemplo, temos um formulário na página **produto.php** e vamos criar uma próxima página, **checkout.php**, que vai receber o produto escolhido e deixar o usuário proceder com a compra.

No formulário de produto então fazemos:

```
<form action="checkout.php">
```

Há ainda um outro atributo do form que indica a *maneira como os dados são enviados*. É o atributo `method` que pode receber dois valores: GET ou POST. Ambos os métodos enviam os dados do formulário ao servidor, mas o GET faz isso via parâmetros na URL enquanto o POST envia os dados no corpo da requisição (e, portanto, não é visível na barra de endereços).

```
<form action="checkout.php" method="POST">
```

10.2 - PARÂMETROS COM PHP

Os dados enviados no formulário são recebidos no PHP e podemos acessá-los

através de variáveis do próprio PHP: `$_GET` e `_POST`, dependendo de qual foi o método do formulário.

Para acessar o valor de um certo campo preenchido, precisamos saber o **nome** dele. No HTML do formulário, sempre que criamos um componente, damos um `name` a ele:

```
<input type="text" name="mensagem">
```

No PHP, podemos acessar cada parâmetro individualmente usando uma sintaxe de arrays:

```
<?= $_GET["mensagem"] ?>
```

ou

```
<?= $_POST["mensagem"] ?>
```

Imprimindo variável com PHP

Até o capítulo anterior, quando queríamos imprimir algo na tela usamos a construção:

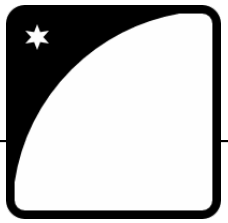
```
<?php print $dado; ?>
```

Mas o PHP permite uma sintaxe mais curta para declarar um bloco de código quando a única ação é imprimir algo e não há várias linhas de código.

```
<?= $dado ?>
```

É uma sintaxe mais curta que a primeira e tem exatamente o mesmo efeito.

No próximo exercício, vamos criar uma página simples de checkout que, por enquanto, apenas mostra uma mensagem de confirmação para o usuário seguida dos parâmetros que foram submetidos no formulário.



Consulte as vantagens do curso *Desenvolvimento Web com HTML, CSS e JavaScript*.

10.3 - Listas de definição no HTML

Quando falamos de listas em HTML, sempre lembramos da `<ul>` e da `<ol>`. Essas são listas mais clássicas, mudando apenas que uma não tem ordem e a outra tem.

Mas existe uma terceira lista, que semanticamente serve para definir itens. É uma lista de termos e suas respectivas definições. Se quiséssemos criar uma lista das siglas de cursos da Caelum e seu respectivo nome, podemos fazer assim:

```
<dl>
  <dt>WD-43</dt>
  <dd>Desenvolvimento Web com HTML, CSS e JavaScript</dd>

  <dt>WD-47</dt>
  <dd>Programação front end avançada com JavaScript e jQuery</dd>
</dl>
```

A lista é a DL e cada termos é representado por um DT seguindo por sua definição em um DD.

10.4 - Exercícios: checkout da compra

1. Crie o arquivo **checkout.php** com uma estrutura básica.

```
<!doctype html>
<html>
<head>
  <meta charset="UTF-8">
  <title>Checkout Mirror Fashion</title>
  <meta name="viewport" content="width=device-width">
</head>
<body>

  <h1>Ótima escolha!</h1>
  <p>Obrigado por comprar na Mirror Fashion!
  Preencha seus dados para efetivar a compra.</p>

</body>
</html>
```

Não vamos usar o menu da Mirror Fashion nessa página, para criar uma experiência de checkout mais imersiva.

2. O formulário na página de produto precisa enviar os dados escolhidos para a página de checkout. Para isso, nosso formulário deve indicar para onde ser submetido.

Altere a tag `<form>` no arquivo **produto.php** para apontar para nossa nova página adicionando o atributo `action`:

```
<form action="checkout.php">
```

Abra a página de produto no navegador e teste o submit. Repare como a página de checkout é chamada e os parâmetros escolhidos na página anterior vão junto na URL.

3. Os parâmetros enviados no capítulo anterior aparecem na URL. Isso porque nosso formulário, por padrão, é do tipo GET. Podemos trocar para o tipo POST e, assim, os parâmetros serão enviados mas não estarão visíveis na URL.

Altere a tag `<form>` no arquivo **produto.php** para enviar via POST adicionando o atributo `method`:

```
<form action="checkout.php" method="POST">
```

Na página **checkout.php** podemos pegar os valores submetidos através de variáveis do PHP e exibir esses valores na tela. **Adicione** na página de checkout as informações de *cor* e *tamanho* escolhidos. Use uma lista de definições DL com DT/DD pra isso:

```
<h2>Sua compra</h2>
<dl>
  <dt>Cor</dt>
  <dd><?= $_POST["cor"] ?></dd>

  <dt>Tamanho</dt>
  <dd><?= $_POST["tamanho"] ?></dd>
</dl>
```

Teste acessando um produto e clicando em comprar. Os valores escolhidos devem aparecer na página de checkout, através dos parâmetros escolhidos.

Teste várias vezes, com escolhas diferentes, pra ver como nossa página de checkout é *dinamicamente* construída com os parâmetros enviados.

4. Imagine que vamos ter vários produtos diferentes na loja, todas enviando as compras pra nossa página de checkout. E lá queremos saber qual o nome do produto está sendo comprado (além da cor e do tamanho).

Podemos passar mais um parâmetro para a página de checkout com o **nome** do produto. Use um `input hidden` pra passar essa informação. Na página produto.php, **adicione dentro do form** um input hidden:

```
<input type="hidden" name="nome" value="Fuzzy Cardigan">
```

Por fim, na página checkout.php, **adicione** a impressão do parâmetro `nome` dentro da lista de definições que fizemos antes:

```
<dt>Produto</dt>
<dd><?= $_POST["nome"] ?></dd>
```

Teste o funcionamento do hidden fazendo uma nova compra de produto.

10.5 - Exercícios opcionais

1. Além do nome do produto, passe também o preço do produto escolhido através de um input hidden. Imprima essa informação na DL da página de checkout.

Crie ainda um outro input hidden pra passar o ID.

2. Na página **produto.php**, mude o `action` do formulário para **GET** ao invés do POST que temos hoje. Teste tudo de novo. O que muda? Qual a diferença? O que precisa ser alterado no checkout?

10.6 - Banco de dados e SQL

Na maioria dos sistemas Web, os dados do negócio ficam separados em um

banco de dados ao invés de ficar escritos diretamente no HTML. Em uma loja virtual como a nossa, os produtos seriam cadastrados nesse banco de dados externo e depois seus dados são exibidos na página produto.php.

Um banco de dados é como uma grande planilha do Excel que possui várias tabelas dentro. Cada tabela tem colunas com campos específicos e muitas linhas com os dados cadastrados. Podem existir relações entre as tabelas.

id	nome	preco	descricao	data	vendas
1	Fuzzy Cardigan	R$ 129,00	Esse é o melhor casaco de Cardigã qu...	2013-09-01	5
2	Camiseta Ecko Caveira Bad to The Bone	R$ 48,95	Camiseta confortável ideal para eventos cas...	2013-10-01	90
3	Cardigan Thelure Basic	R$ 296,00	Cardigan em ótimo caimento. Ideal fazer sob...	2013-09-29	12
4	Casaco Winter	R$ 699,00	Jaqueta confeccionada em couro ecológico, e...	2013-09-20	3
5	Sport Top	R$ 89,00	Camisa em Algodão estampada ideal para uma ...	2013-09-27	45
6	Top Basic	R$ 45,00	Blusa multicolorida com decote em V, a manga longa...	2013-09-01	178
7	Camiseta Gwol	R$ 125,00	Camiseta em modelagem reta, mangas curtas e decote...	2013-09-01	84
8	Camiseta Tiup	R$ 129,00	Regata básica em seda. Combinaç&atil...	2013-09-03	123
9	Camisa Squares	R$ 199,00	Camisa em tecido mega confortável. Gola fec...	2013-09-18	78
10	Top in Slub	R$ 118,00	Esse é o melhor casaco de Cardigã qu...	2013-09-03	146
11	Shorts Lez a Lez Towel	R$ 77,00	Sport deluxe é a tendência quent&iacu...	2013-09-02	345
12	Camisa Richards Sam	R$ 420,00	Em 1974 a Richards lançou no Brasil um novo...	2013-09-07	256
13	Blusa Lez a Lez Feel Sft l	R$ 107,00	Blusa manga longa, decote redondo com fechamento e...	2013-09-03	42

Figura 10.1: Exemplo de uma tabela com produtos da loja virtual

Para manipular os dados dessa tabela, usamos uma linguagem separada conhecida como **SQL**. Seu papel é permitir que façamos buscas nas tabelas por certos tipos de dados e que possamos inserir, remover e atualizar dados.

É uma linguagem a mais pra aprender, mas bem diferente das que vimos até agora como PHP ou JavaScript. O SQL serve *apenas para acessar bancos de dados* e é bem mais simples.

Por exemplo, se queremos acessar todas as informações da tabela com nome `produtos`, fazemos:

```
SELECT * FROM produtos
```

O comando **SELECT** indica que estamos selecionamos dados. O asterisco indica que queremos todos os dados. E o **FROM produtos** aponta o nome da tabela.

Podemos selecionar apenas um dado específico. Por exemplo, apenas os nomes dos produtos:

```
SELECT nome FROM produtos
```

Vamos ver outros recursos do SQL mais pra frente. Mas não é foco do curso. Para se aprofundar no tema, recomendamos esse curso online do Alura:

http://www.alura.com.br/cursos-online-introducao/banco-de-dados-sql

10.7 - MySQL e phpMyAdmin

Existem muitos bancos de dados no mercado, como MySQL, Oracle, SQL Server, DB2, Postgre, SQLite. Todos eles aceitam comandos SQL que vimos antes, com pequenas variações apenas.

No curso, usaremos o **MySQL** que é um dos bancos mais usados no mundo e bastante usado por programadores PHP. É quase uma escolha natural.

http://www.mysql.com

Para ajudar a visualizar e administrar o MySQL podemos instalar alguma interface gráfica compatível. O próprio MySQL tem um produto chamado **Workbench** que é um programa Desktop pra isso. Mas uma opção muito comum de encontrar no mercado e em empresas de hosting é o **phpMyAdmin**.

Ele é um administrador de MySQL escrito em PHP para Web e que roda num servidor normal e pode ser acessado direto no navegador. Isso faz dele uma ferramenta versátil e útil para acessar bancos de dados remotos.

Vamos usar o phpMyAdmin no curso. Basta baixar o zip no site deles e subir um servidor php na pasta dele, como fizemos na pasta do nosso próprio projeto.

http://www.phpmyadmin.net/

10.8 - Para saber mais: instalação do MySQL em casa

O MySQL pode ser baixado em https://dev.mysql.com/downloads/mysql/

Lá, escolha seu sistema operacional (Windows, Mac, Linux) e baixe o pacote correto. Basta executá-lo que a instalação acontecerá.

Importante: Se você instalou o WAMP no primeiro capítulo de PHP da apostila, não precisa instalar o MySQL agora.

10.9 - Buscas no MySQL com PHP

Sabendo usar um banco de dados e a escrever SELECTs para extrair dados dele, o próximo passo é aprender a fazer isso de dentro da nossa página PHP. Isto é, queremos pegar dados no banco de dados e imprimir na tela usando PHP.

Precisamos ver como conectar no MySQL e disparar comando SQL usando PHP.

Conexão

A conexão com o banco de dados pode ser feita com a função `mysqli_connect`:

```
$conexao = mysqli_connect("localhost", "root", "", "wd43");
```

Essa função recebe onde de conectar (localhost), o usuário (root), a senha (em branco) e o nome do banco de dados disponível (wd43). Ela abre a conexão se tudo der certo e devolve uma variável `$conexao` que representa a conexão aberta.

A busca

Próximo passo é mandar o MySQL processar um certo comando SQL, como

nossa busca SELECT de antes. Pra isso, usamos a função `mysqli_query` que recebe a conexão (que abrimos antes) e o SQL da busca:

```
$dados = mysqli_query($conexao, "SELECT * FROM produtos");
```

Essa função devolve uma variável `$dados` com o retorno que a busca der.

Dados no PHP

Último passo é transformar os dados da tabela em algo usável no PHP. Uma maneira comum é transformar os dados num **array** com `mysqli_fetch_array`:

```
$produto = mysqli_fetch_array($dados);
```

A variável `$produto` é um array PHP com os dados do primeiro produto da busca indexados pelo nome da coluna no banco de dados. Isso quer dizer que podemos acessar, por exemplo, o preço do produto fazendo `$produto["preco"]` e assim por diante para cada coluna.

10.10 - Refinando as buscas com WHERE

Há muitas opções possíveis no SQL para refinarmos a busca. O `SELECT *` que fizemos antes retorna todos os dados da tabela inteira, o que pode ser muita coisa. Imagine que estamos interessados nos dados apenas de um produto específico, de uma certa linha.

Podemos indicar ao SELECT que queremos os dados do produto de um certo ID, que é uma coluna numérica que temos no banco de dados para identificar o código individual de cada produto. Fazemos isso no SQL com a cláusula **WHERE**.

```
SELECT * FROM produtos WHERE id = 4
```

Esse código devolve todas as colunas do produto cujo id for 4, e apenas ele.

10.11 - Exercícios: phpMyAdmin

1. Vá no terminal e encontre a pasta do **phpmyadmin**:

```
$ cd /caelum/cursos/43/phpmyadmin
```

Dentro dela, rode um servidor PHP em uma porta diferente, como 8000:

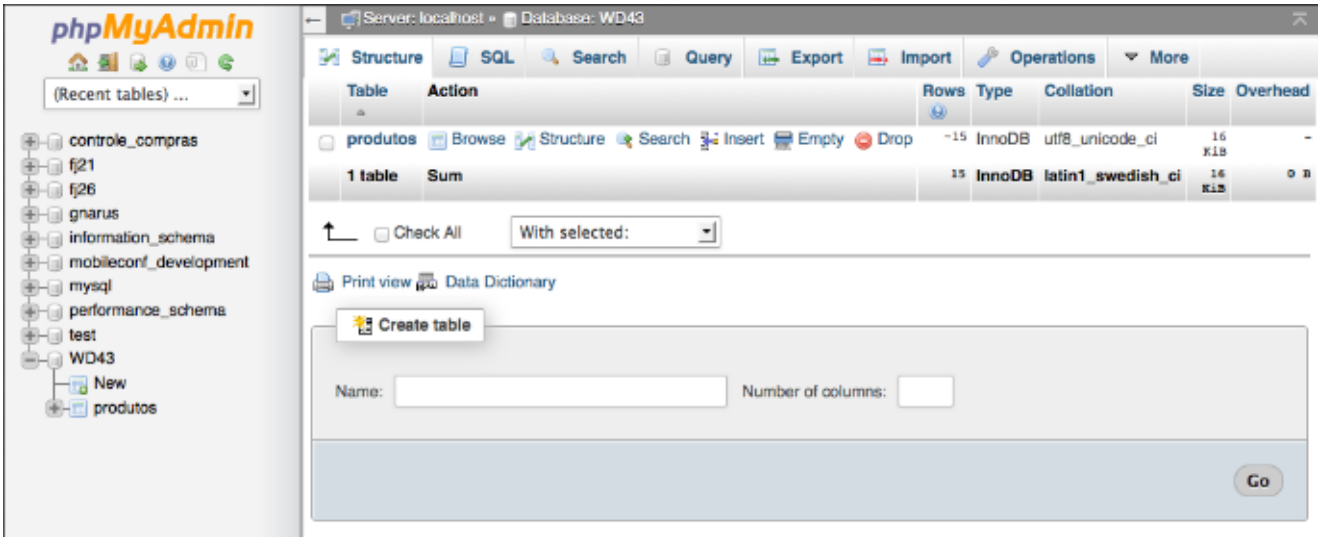
```
$ php -S 0.0.0.0:8000
```

Vá no navegador e acesse: http://localhost:8000

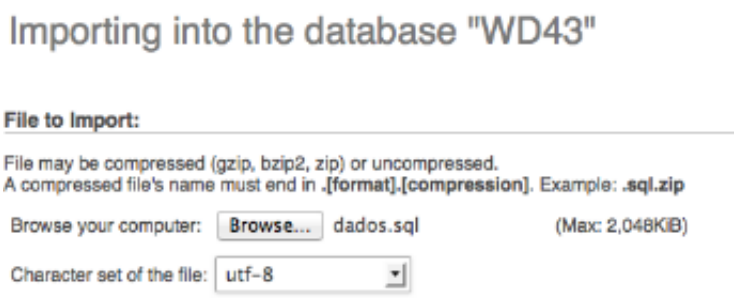
Você deverá ver o phpMyAdmin. Coloque usuário **root** e deixe a senha em branco para entrar.

2. A primeira tela do phpMyAdmin pode ser assustadora de tantas opções. Ele tem muitos recursos.

Na parte esquerda ficam os **bancos de dados** disponíveis na máquina. Localize e clique no banco **WD43** que corresponde aos dados do nosso curso.



Podemos importar os dados do curso pra esse banco. Eles estão no arquivo **dados.sql** na pasta do curso. Para importar, vá no menu **Import** no topo do phpMyAdmin. Clique em *Browse* para selecionar o arquivo no seu computador e depois clique em **Go**.



Após a importação, selecione a **tabela produtos** dentro de WD43 no menu da esquerda. Ele deve mostrar os dados que estão dentro da tabela, suas colunas e linhas.

Observe o comando SELECT que o phpMyAdmin gerou para obter os dados:

```
SELECT *
FROM `produtos`
LIMIT 0 , 30
```


10.12 - Exercícios: PHP com MySQL

1. No topo do arquivo **produto.php** abra a conexão com o banco de dados e selecione os dados do produto:

```
<?php
  $conexao = mysqli_connect("127.0.0.1", "root", "", "WD43");
  $dados = mysqli_query($conexao, "SELECT * FROM produtos");
  $produto = mysqli_fetch_array($dados);
?>
```

Repare que criamos uma variável `$produto` no PHP que contém os dados do

produto. Ela é um array e podemos acessar as diferentes colunas através do nome.

Altere os títulos na página para usar os dados dinâmicos do banco:

```
<h1><?= $produto["nome"] ?></h1>
<p>por apenas <?= $produto["preco"] ?></p>
```

Altere também, no final da página, o local onde mostramos o texto da descrição do produto:

```
<p><?= $produto["descricao"] ?></p>
```

Teste nossa página de produto no navegador e repare que os dados vêm dinamicamente do banco. Observe o código fonte HTML final gerado, como é idêntico ao que tínhamos antes.

2. Queremos que nossa página seja capaz de exibir os dados de qualquer produto do banco. Para escolher qual produto mostrar, vamos usar um *parâmetro na URL* com o código do produto, o ID.

Altere o código da busca que fizemos antes para incluir a cláusula **WHERE** no final baseada no id do produto passado como parâmetro:

```
"SELECT * FROM produtos WHERE id = $_GET[id]"
```

Teste a página no navegador passando ids diferentes com parâmetro na URL: **produto.php?id=2**.

3. A imagem do produto também é diferente para cada produto. Abra a pasta **img/produtos/** e repare nas várias imagens que estão lá. Elas seguem um padrão. O nome contém o código do produto e as três variações de cor. Podemos gerar o endereço das imagens no HTML usando o ID que vem do PHP.

Altere os caminhos das imagens na página de produto pra passar o ID dinamicamente no endereço da foto:

```
<img src="img/produtos/foto<?= $produto["id"] ?>-verde.png">
```

Teste a página no navegador passando ids diferentes com parâmetro na URL: **produto.php?id=4**.

4. Adicione mais um input hidden dentro do formulário passando o id. Precisaremos dele mais a frente.

```
<input type="hidden" name="id" value="<?= produto["id"] ?>">
```

5. (opcional) O `<title>` da página é um fator muito importante para motores de busca (SEO). O ideal é ter títulos descritivos e únicos em cada página. Numa loja virtual como a nossa, o nome do produto deve fazer parte do título da página.

Altere o título da página pra puxar o nome do produto do banco de dados dinamicamente.

6. (opcional) Mude os input hidden que fizemos no capítulo anterior com nome/preço pra pegar valores dinâmicos do banco de dados usando PHP.

10.13 - Busca de muitos resultados

Podemos fazer uma busca de muitos resultados removendo o WHERE e fazendo o SELECT simples de antes:

```
SELECT * FROM produtos
```

Mas também podemos restringir o número de resultados. Imagine que a tabela é imensa, de milhares de registros, mas queremos apenas os primeiros 10 agora. No MySQL, isso pode ser feito com o comando **LIMIT** na busca:

```
SELECT * FROM produtos LIMIT 0, 10
```

Isso significa que queremos 10 resultados contando a partir do primeiro (0). Podemos trocar o primeiro número pra acessar informações em outras partes do banco.

10.14 - Ordenação dos resultados

Outro recurso interessante do banco de dados é devolver as informações ordenadas de acordo com certo critério. Se quisermos os produtos em ordem alfabética pelo nome dele:

```
SELECT * FROM produtos ORDER BY nome
```

Podemos ainda escolher o sentido da ordenação com **ASC** (ascendente) e **DESC** (decrescente). Por exemplo, para ordenar os produtos no sentido do mais recente para o mais antigo:

```
SELECT * FROM produtos ORDER BY data DESC
```

E, por fim, misturar tudo isso num SQL complexo:

```
SELECT * FROM produtos ORDER BY data DESC LIMIT 0, 10
```

Tire suas dúvidas no novo GUJ Respostas

O GUJ é um dos principais fóruns brasileiros de computação e o maior em português sobre Java. A nova versão do GUJ é baseada em uma ferramenta de *perguntas e respostas* (QA) e tem uma comunidade muito forte. São mais de 150 mil usuários pra ajudar você a esclarecer suas dúvidas.

Faça sua pergunta.

10.15 – Exercícios: mais buscas com PHP

1. A nossa home page lista os produtos mas, do jeito que fizemos, está tudo estático com dados de mentira. Vamos alterar para fazer uma busca no banco de dados e retornar os produtos a serem exibidos. É muito parecido com o que fizemos na página de produto; a diferença é que vamos listar vários produtos de uma vez ao invés de um só.

Em primeiro lugar, precisamos transformar nossa página num arquivo PHP, para poder usar o banco de dados nela. **Renomeie** o arquivo `index.html` para `index.php` se ainda não estiver.

No **index.php**, localize o **painel novidades** e **apague** todos os `<li>` que listam produtos com dados estáticos.

No lugar, escreva um código PHP que faz a busca dos produtos no banco de dados e percorra essa lista com um laço **while**:

```
<ol>
  <?php
    $conexao = mysqli_connect("127.0.0.1", "root", "", "WD43");
    $dados = mysqli_query($conexao, "SELECT * FROM produtos");

    while ($produto = mysqli_fetch_array($dados)):
  ?>

  <li>
    <a href="produto.php?id=<?= $produto["id"] ?>">
      <figure>
        <img src="img/produtos/miniatura<?= $produto["id"] ?>.png"
             alt="<?= $produto["nome"] ?>">
        <figcaption><?= $produto["nome"] ?> por <?= $produto["preco"] ?>
```

```
</figcaption>
      </figure>
    </a>
  </li>

  <?php endwhile; ?>
</ol>
```

Repare como refizemos o `<li>` de antes mas usando todos os dados dinâmicos do banco de dados.

Teste a home no navegador e veja a busca dinâmica acontecendo. Confira o código fonte HTML gerado, igual ao que tínhamos antes.

2. A busca que fizemos antes com o `SELECT` traz todos os dados da tabela. Isso é potencialmente bem grande numa loja de verdade. O ideal é restringir a busca apenas pelos dados necessários. Pra isso, no MySQL, podemos usar o comando `LIMIT` passando o máximo de resultados que estamos interessados.

Altere o código anterior para incluir o **LIMIT** no SQL da busca:

```
SELECT * FROM produtos LIMIT 0, 6
```

Teste novamente a home.

3. Repare que a ordem que os produtos vêm não é a ordem que gostaríamos. No painel novidades, queríamos que viessem ordenados pelo produto mais recente ao mais antigo.

Podemos fazer isso adicionando uma cláusula de ordem no SQL com **ORDER BY**.

Altere a busca anterior para incluir uma ordenação com base no campo `data` de maneira decrescente:

```
SELECT * FROM produtos ORDER BY data DESC LIMIT 0, 6
```

Teste novamente a home e veja o resultado.

4. (opcional) Implemente o mesmo recurso de busca dinâmica no outro painel, o de produtos mais vendidos. A única diferença é que queremos ordenar os elementos a partir da quantidade de vendas. Use o campo **vendas** no ORDER BY pra isso.

5. (opcional) Teste outros valores para o LIMIT nas buscas. O primeiro número indica *offset*, ou seja a partir de qual item estamos interessados. Usar **3,6** indica que queremos os itens do terceiro ao nono.

Teste também outras ordenações. Além do DESC, temos o ASC.

10.16 - Exercícios opcionais

1. Em vez de passar os dados do produto para o checkout via input hidden, a solução mais comum na prática é só passar o ID do produto sendo comprado. E a página de checkout, para obter os dados (nome, preço, etc), faz novamente uma busca no banco de dados.

Essa solução é mais segura pois impede que o usuário altere os dados no HTML (como o preço). Só o ID é passado como parâmetro e os dados sempre vêm do banco de dados.

Implemente essa solução no seu projeto.

CAPÍTULO ANTERIOR:

Progressive enhancement e mobile-first

PRÓXIMO CAPÍTULO:

Bootstrap e formulários HTML5

Facebook

Newsletter

Casa do Código

Twitter