

Sistemas de e-commerce em PHP

**(Apostila utilizada nos cursos Alfamídia de PHP
em 2007 – disponibilizada agora gratuitamente)**

Todos os direitos reservados para Alfamídia Prow LTDA.

AVISO DE RESPONSABILIDADE

As informações contidas neste material de treinamento são distribuídas “NO ESTADO EM QUE SE ENCONTRAM”, sem qualquer garantia, expressa ou implícita. Embora todas as precauções tenham sido tomadas na preparação deste material, a Processor Alfamídia LTDA. não têm qualquer responsabilidade sobre qualquer pessoa ou entidade com respeito à responsabilidade, perda ou danos causados, ou alegadamente causados, direta ou indiretamente, pelas instruções contidas neste material ou pelo software de computador e produtos de hardware aqui descritos.

04/2007 – Versão 1.1

Alfamídia Prow LTDA
Porto Alegre, RS
(51) 3073-2100
<http://www.alfamidia.com.br>

Sistemas de e-commerce em PHP

UNIDADE 1	ORGANIZAÇÃO DO CURSO.....	4
UNIDADE 2	INTRODUÇÃO.....	5
UNIDADE 2	A CLASSE TEMPLATE POWER.....	7
UNIDADE 3	INTRODUÇÃO AO E-COMMERCE	13
UNIDADE 4	ESTRUTURA DA LOJA VIRTUAL	14
UNIDADE 5	A PÁGINA INICIAL.....	22
UNIDADE 6	ARQUIVOS, CLASSES E MÉTODOS PARA A LOJA.....	24
UNIDADE 7	O ARQUIVO PHP INICIAL	26
UNIDADE 8	ELABORAÇÃO DO ARQUIVO CSS:	29
UNIDADE 9	A LISTAGEM DE PRODUTOS	32
UNIDADE 10	O FORMULÁRIO DE CONTATO	36
UNIDADE 11	A IMPLEMENTAÇÃO DO CARRINHO DE COMPRAS	40
UNIDADE 12	O SCRIPT DO CARRINHO DE COMPRAS.....	44
UNIDADE 13	FINALIZAÇÃO DA COMPRA	50
UNIDADE 14	A ADMINISTRAÇÃO DA LOJA	57
UNIDADE 15	VALIDANDO A SESSÃO.....	62
UNIDADE 16	EFETUANDO LOGOUT, FINALIZANDO A SESSÃO.....	81

Unidade 1

Organização do curso

1.1 Sobre o Curso

O Curso Sistemas de E-commerce em PHP oferece aos alunos um apanhado geral sobre técnicas e conceitos envolvidos no desenvolvimento de um Sistema de E-commerce.

Ao final do curso os alunos serão capazes de desenvolver um Sistema completo de E-commerce utilizando a linguagem PHP.

Unidade 2

Introdução

A elaboração de sistemas para web sites pode ser totalmente independente do desenvolvimento do próprio web site. Se o seu cliente já possui um web site e deseja somente agregar novas ferramentas ou atratividades, esse é o caminho ideal para a obtenção de um ótimo resultado com rapidez e agilidade.

A implantação de uma loja virtual, por mais simples que seja, dá uma boa impressão aos visitantes do web site, visto que para ser implementada deve ter passado por um processo de elaboração bastante criterioso, a fim de não deixar margem a possíveis erros e reclamações dos usuários.

Dessa forma, um sistema como esse poderá, além de ampliar o alcance da loja e dos produtos da mesma no mercado, dar ao visitante e os clientes uma aparência de controle e boa estrutura física e administrativa da empresa.

O desenvolvimento desse um sistema desse porte deve ser muito bem elaborado desde o início e, por esse motivo, é indispensável a implementação de alguns passos. Esses passos podem ser:

- Análise do problema.
- Modelagem dos dados e verificação da integridade das tabelas.
- Diagrama de funcionamento.
- Desenvolvimento do "site map".
- Reutilização de arquivos (includes, funções e classes).
- Criação dos layouts (telas).
- Criação das páginas de processamento (PHP).
- Testes e ajustes.

Este sistema que estaremos desenvolvendo ainda contará com a utilização de uma programação bem organizada e, de uma certa forma, portátil, baseada em orientação a objetos, uma vez que o PHP nos possibilita esse uso. Utilizaremos a classe TemplatePower para tal fim.

Antes de começarmos a desenvolver as etapas para a criação de uma loja virtual, veremos como utilizar a classe TemplatePower.

Unidade 2

A classe Template Power

Com a melhor integração do PHP ao paradigma de orientação a objetos, diversas classes foram implementadas e distribuídas com o objetivo de facilitar a vida dos programadores. Dentre classes que manipulam imagens, arquivos em pdf, compactação de arquivos, etc. uma delas destaca-se por resolver um dos maiores problemas enfrentados no desenvolvimento de aplicações web em php a mistura do código HTML com a linguagem dinâmica.

A classe Template Power teve sua última versão implementada em 15 de março de 2003, e está disponível para download no site: <http://templatepower.codocad.com/download.php>, veja abaixo um exemplo comparativo de como funciona a classe Template Power:



Comportando-se como uma parede divisória entre as duas codificações, a sistemática de implementação é facilitada, pois o layout (HTML) não fica mais misturado com a aplicação (PHP). Problemas como alteração de layout do site não afetam mais a estrutura de programação, problemas que eram comuns principalmente quando utilizávamos aplicativos para a manipulação do layout os quais não interagiam com a linguagem, eliminando as vezes algumas linhas de programação essenciais na aplicação do site.

A forma de manipulação do layout segue-se que teremos um arquivo PHP para cada arquivo HTML que confeccionarmos, sendo que em casos mais específicos poderemos realizar a junção de dois, ou mais arquivos com o intuito de formamos uma pagina única.

Veja a seguir os métodos existentes na classe Template Power e como se segue o comportamento de cada um.

Método	Funcionalidade:
TemplatePower()	Método construtor
assignInclude	Junção de dois ou mais arquivos de layout.
prepare	Método que prepara o documento HTML para interpretação do PHP.
assign()	Responsável pela substituição de strings do HTML por conteúdo dinâmico do PHP.
newBlock()	Método responsável por direcionar o ponteiro de leitura da classe para determinada área do documento HTML.
gotoBlock()	Método responsável por enviar o ponteiro de leitura da classe para fora de determinado bloco.
showUnAssigned()	Opção que serve para debug de variáveis que não foram “assignadas”.
getOutPutContent()	Retorna todo conteúdo gerado pelo HTML para uma variável, é muito utilizado em scripts para envio de newsletters e mailing.
printToScreen()	Método final que mostra o resultado gerado pela classe na tela em formato HTML.

Os arquivos de layout podem possuir extensão HTML, ou TPL. Entretanto a segunda é menos utilizada devido a falta de capacidade de interpretação dessa extensão por aplicativos de desenvolvimento de layout (Macromedia Dreamweaver por exemplo).

Para utilizarmos a classe deveremos sempre dar um INCLUDE no arquivo class.TemplatePower.php. Esse include deverá estar presente em cada arquivo php que precise de um layout para ser apresentado.

2.1 Exemplo 1

Vamos criar o nosso primeiro documento utilizando a classe Template Power. Comece criando um documento com extensão “.HTM” e salvando o arquivo com o nome de exemplo1.htm.

```
<html>
<head>
```



```
<title>.: Exemplo 1 :.</title>
</head>
  <body>
    Bem Vindo, {fulano}
  </body>
</html>
```

Como você pode observar o código possui uma palavra entre chaves, “{fulano}”, a classe Template Power possui métodos que visam substituir o conteúdo de palavras entre chaves por conteúdo dinâmico gerado pelo script em php.

Na segunda parte do nosso exemplo iremos criar agora um arquivo com extensão “.PHP” que possuirá o mesmo nome que o criado anteriormente (exemplo1.php).

O código deverá ficar como mostrado abaixo:

```
<?php
include("class.TemplatePower.php");
$tpl = new TemplatePower("ex_1.htm");
$tpl->prepare();
$tpl->assign("fulano","Cícero Feijo");
$tpl->printToScreen();
?>
```

Observe a ordem que os métodos estão sendo mencionados no código:

2.2 Estruturas de repetição

Estruturas de repetição são comumente usadas na exibição de registros que foram recuperados do banco de dados, visando mostrar de uma maneira mais agradável os registros na integração com o Template.

Estas estruturas devem ser “citadas” no arquivo de Template de uma forma bastante específica, neste caso, utilizamos comentários em HTML para simbolizar uma estrutura de repetição ou BLOCO.

Veja o exemplo abaixo:

```
<!-- START BLOCK : numeros -->
  <tr>
    <td>{numero}</td>
    <td>{numero_vezes_10}</td>
  </tr>
```

```
<!-- END BLOCK : numeros -->
```

No arquivo PHP que será responsável por processar este Template, iremos nos referir a este BLOCO da seguinte maneira:

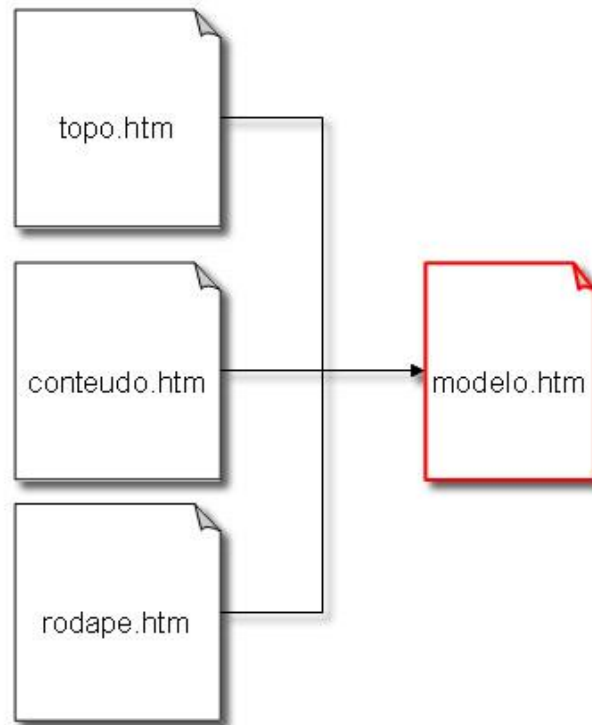
```
for($x = 0; $x < 10; $x++){  
    $tpl->newBlock('numeros');  
    $tpl->assign("numero", $x);  
    $tpl->assign("numero_vezes_10", $x*10);  
}  
$tpl->gotoBlock('_ROOT');
```

Não esquecendo do método “gotoBlock(‘_ROOT’)”, o qual é responsável por liberar o ponteiro de leitura do arquivo do Template, disponibilizando a leitura do restante do arquivo HTML.

2.3 Integração entre arquivos de Template diferentes

Dentre as formas de montagem de um site utilizando Template, uma das mais usadas é a mesclagem de diversos arquivos de Template de forma que estes sejam interpretados pelo script PHP de forma única. Chamamos esse processo de Inclusão de Bloco.

Um exemplo prático seria a inclusão de um MENU ou RODAPÉ no site, o qual tivessem a necessidade de interpretação de conteúdo dinâmico.



A maneira de integrar estes três arquivos no mesmo Template se dá da seguinte forma:

Arquivo HTML:

```
<body>
<!-- INCLUDE BLOCK : TOPO -->
<!-- INCLUDE BLOCK : CONTEUDO -->
<!-- INCLUDE BLOCK : RODAPE -->
</body>
```

Arquivo PHP:

```
$tpl->new TemplatePower('../htm/modelo.htm');
$tpl->assignInclude('topo','../htm/topo.htm');
$tpl->assignInclude('conteudo','../htm/conteudo.htm');
$tpl->assignInclude('rodape','../htm/rodape.htm');
$tpl->prepare();
.
.
.
$tpl->printToScreen();
```

O processamento dos arquivos será feito somente por um arquivo PHP, sendo que todas as TAGS de Template colocadas em ambos serão interpretadas de uma única vez.

2.4 Exercício:

Confeccione uma seção inicial de um site, que possua três arquivos, um cabeçalho, um rodapé, um menu lateral. No topo do cabeçalho deverá ser apresentada a data atual, e no rodapé do mesmo os créditos de implementação do projeto.

Você deverá possuir apenas um script PHP gerenciando estas três partes do site.

Unidade 3

Introdução ao E-Commerce

Atualmente com o vínculo cada vez maior da internet ao nosso dia a dia, não basta apenas para um empresa possuir uma propaganda da sua corporação publicada na internet de forma estática, esse conteúdo deve ser apresentado como uma espécie de serviço, onde o cliente possa interagir com o site, e obter informações diretas e precisas sobre o que procura. Além da divulgação de produtos e empresas a internet acabou tendenciando uma nova forma de comércio, o E-Commerce, ou comércio eletrônico, onde existem algumas modalidades que merecem ser citadas aqui.

- **B2B:** Business to Business: comércio praticado entre fornecedores e clientes empresariais, ou seja, de Empresa para Empresa.
- **B2C:** Business to Consumer: comércio efetuado diretamente entre a empresa produtora, vendedora ou prestadora de serviços com o consumidor final.
- **C2C:** Consumer to Consumer: comércio eletrônico entre usuários da internet, não envolvendo intermediários, sendo realizado direto pelo consumidor final.
- **G2C:** Government to Consumer: do governo para os consumidores, como pagamento de tarifas, declaração de impostos, multas e tarifas públicas.
- **G2B:** Government to Business: relação de negócios na internet entre governos e empresas.

Veremos a seguir a implementação parte a parte de cada módulo para o desenvolvimento de um Comércio Virtual na modalidade B2C, mas que pode ser aplicado a qualquer uma das outras mencionadas acima, sem necessidade de remodelações discrepantes.

Unidade 4

Estrutura da Loja Virtual

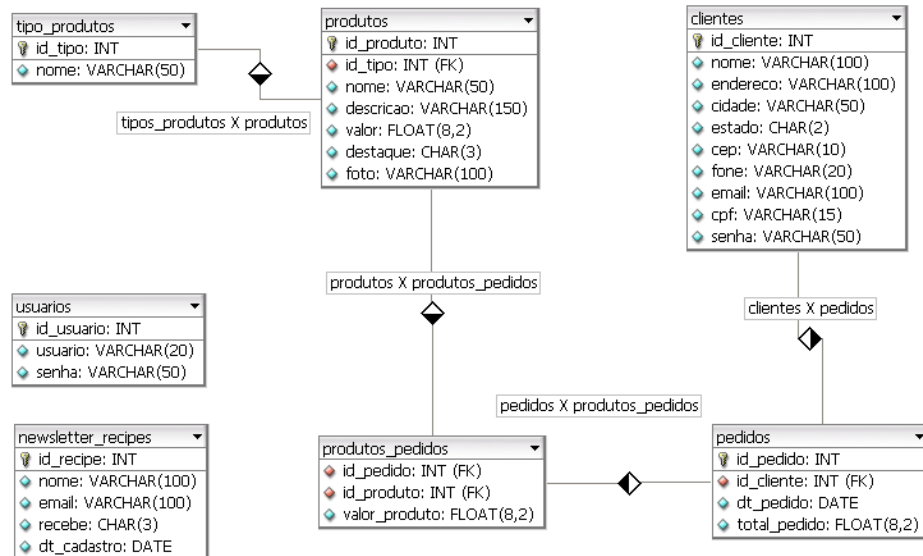
Para a elaboração de uma loja virtual que supra as necessidades do cliente deveremos executar cada passo visto na formação até o momento. Desde a diagramação Entidade Relacionamento do Banco de Dados, até a implementação dos scripts em PHP e arquivos de Template do Projeto.

Nosso projeto final será como mostrado na imagem abaixo, possuindo um ambiente administrativo responsável por alimentar os dados da loja (Cadastro de Produtos e Tipos de Produtos) e possibilitar o acompanhamento de Pedidos que foram efetuados na loja.



4.1 O Diagrama E.R. do Banco de Dados

Abaixo você pode acompanhar como será a estrutura do banco de dados para a loja, apesar de ser uma estrutura simplificada ela possibilita que implementações adjacentes que venham a surgir de cliente para cliente sejam implementadas sem a necessidade de mudanças relevantes no Banco de Dados.



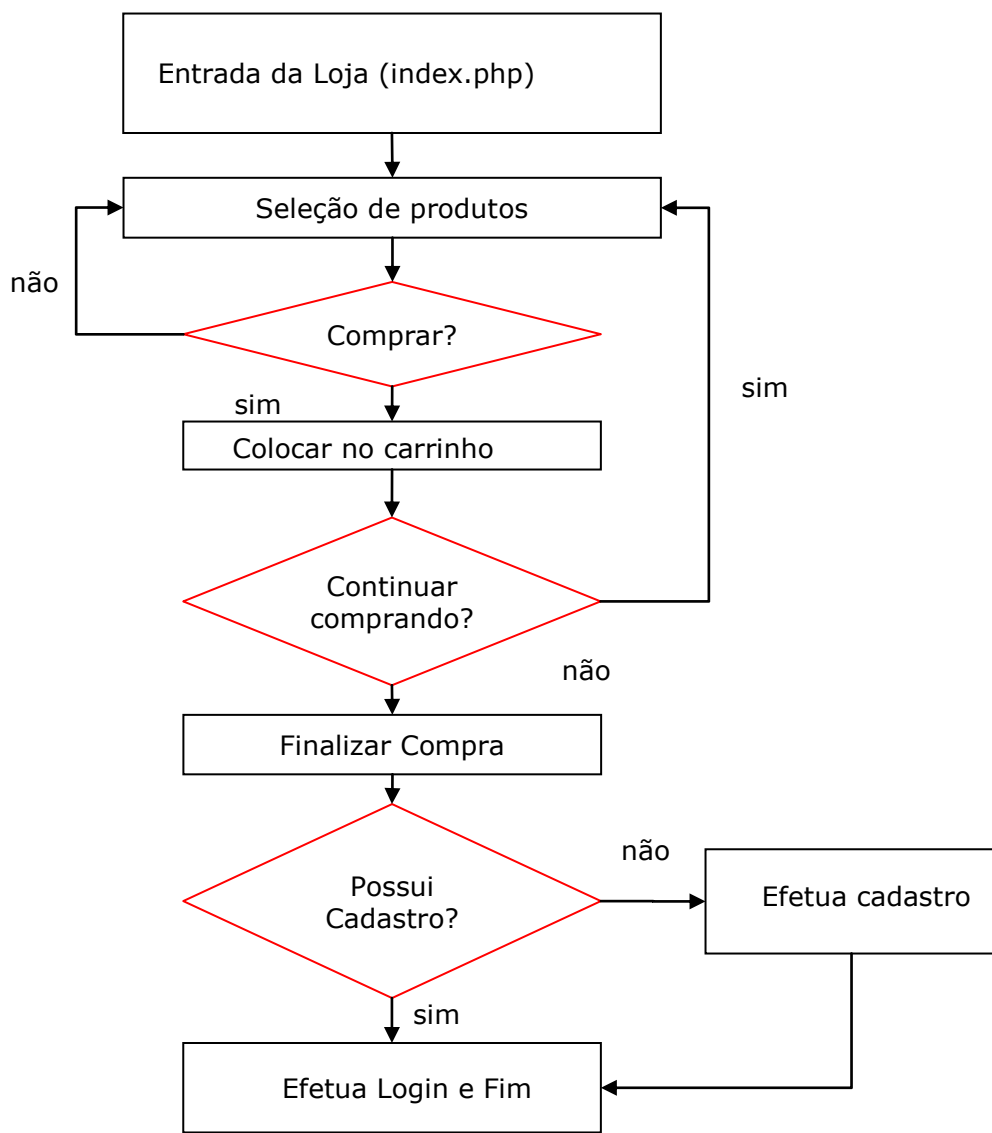
O Banco será composto pelas seguintes tabelas:

1. **tipo_produtos**: tabela responsável por armazenar os dados dos tipos de produtos existentes na loja (hardware, software, suprimentos, etc).
2. **produtos**: tabela responsável por armazenar todos os produtos que estão cadastrados na loja virtual, campos essenciais como nome, descrição e valor estão presentes, ressaltando-se o campo “destaque” o qual servirá para destacar alguns produtos que serão apresentados na página inicial do site.
3. **clientes**: relação de todos os clientes que compraram na loja, no processo final será solicitado o cadastro do cliente ou a login do mesmo caso já possua cadastro conosco.
4. **pedidos**: tabela que armazena o resumo de pedidos, coletando dados simplificados da compra como código do cliente, data do pedido e total do pedido.
5. **produtos_pedidos**: tabela relacional responsável por vincular produtos da loja com os pedidos efetuados pelos clientes já cadastrados. É importante mencionar que essa tabela além de armazenar o código do produto, armazena também o seu valor no momento da compra, evitando que futuras alterações nos preços venham a comprometer os dados já adicionados no banco.
6. **usuarios**: nessa tabela estão cadastrados os usuários que serão capazes de administrar a loja, observe que o campo senha é um VARCHAR de 50, sendo necessário esta tipagem para armazenarmos senhas criptografadas em MD5, aumentando assim a confiabilidade e segurança do ambiente.

7. newsletter_recipes: nessa tabela iremos armazenar os dados de usuários interessados em receber o informativo da loja, veja o script de envio de newsletter no final da apostila como material extra.

4.2 Processo de compra: como funciona?

Para a elaboração dos scripts deveremos verificar como será o processo de compra na loja, desde a entrada do cliente até a finalização da compra no sistema. Isso pode ser facilmente demonstrado com o digrama abaixo:



Todas as compras do cliente serão armazenadas em um vetor bidimensional de sessão, ou seja, caso o usuário feche o navegador ele irá perder todos os produtos existentes no seu carrinho. O propósito da utilização de variáveis de sessão nesse caso visa aumentar a segurança da compra e eliminar consultas desnecessárias ao banco de dados, o que ocasionaria queda de desempenho.

Para a criação do Banco de Dados podemos utilizar qualquer ferramenta já vista na formação, ou bata utilizarmos a query disponível abaixo diretamente no console do MYSQL.

```
#TABELA DE CLIENTES
CREATE TABLE IF NOT EXISTS clientes (
  id_cliente int(11) NOT NULL auto_increment,
  nome varchar(100) ,
  endereco varchar(100) ,
  cidade varchar(50) ,
  estado char(2) ,
  cep varchar(10) ,
  fone varchar(20) ,
  email varchar(100) ,
  cpf varchar(15) ,
  senha varchar(50) ,
  PRIMARY KEY (id_cliente)
);

#TABELA DE NEWSLETTER
CREATE TABLE IF NOT EXISTS newsletter_recipes (
  id_recipe int(11) NOT NULL auto_increment,
  nome varchar(100) ,
  email varchar(100) NOT NULL DEFAULT '' ,
  recebe char(3) DEFAULT 'Sim' ,
  dt_cadastro date ,
  PRIMARY KEY (id_recipe)
);

#TABELA DE PEDIDOS
CREATE TABLE IF NOT EXISTS pedidos (
  id_pedido int(11) NOT NULL auto_increment,
  id_cliente int(11) NOT NULL DEFAULT '0' ,
  dt_pedido date NOT NULL DEFAULT '0000-00-00' ,
  total_pedido float(8,2) ,
  PRIMARY KEY (id_pedido)
);
```

```
#TABELA DE PRODUTOS
CREATE TABLE IF NOT EXISTS produtos (
    id_produto int(11) NOT NULL auto_increment,
    id_tipo int(11) NOT NULL DEFAULT '0' ,
    nome varchar(100) ,
    descricao text ,
    valor float(8,2) ,
    destaque char(3) DEFAULT 'Não' ,
    foto varchar(100) ,
    PRIMARY KEY (id_produto)
);

#TABELA RELACIONAL DE PRODUTOS POR PEDIDOS
CREATE TABLE IF NOT EXISTS produtos_pedidos (
    id_pedido int(11) NOT NULL DEFAULT '0' ,
    id_produto int(11) NOT NULL DEFAULT '0' ,
    valor float(8,2) NOT NULL DEFAULT '0.00' ,
    qtd tinyint(3) unsigned
);

#TABELA DE TIPO DE PRODUTOS
CREATE TABLE IF NOT EXISTS tipo_produtos (
    id_tipo int(11) NOT NULL auto_increment,
    nome varchar(50) ,
    PRIMARY KEY (id_tipo)
);

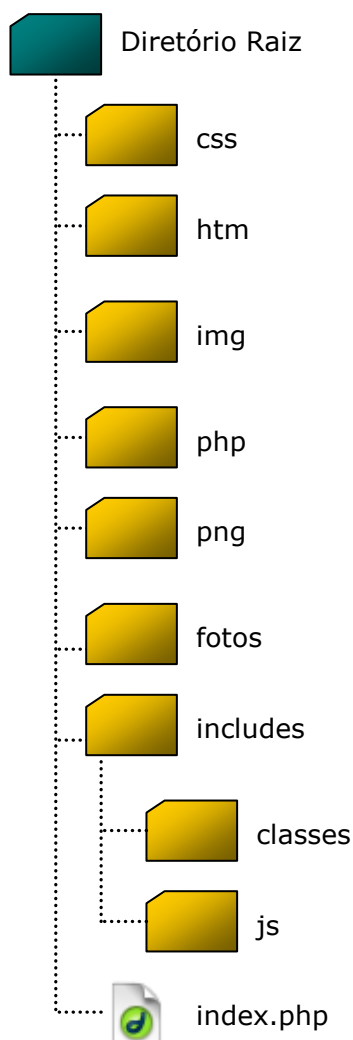
#TABELA DE USUÁRIOS
CREATE TABLE IF NOT EXISTS usuarios (
    id_usuario int(11) NOT NULL auto_increment,
    usuario varchar(20) ,
    senha varchar(50) ,
    PRIMARY KEY (id_usuario)
);
```

4.3 A estrutura de diretórios (site_map)

Com a utilização da classe Template Power notamos que a organização dos scripts é modificada, tanto na questão referente a implementação, quanto a organizacional.

Para isso podemos adotar um padrão que melhora ainda mais o armazenamento de nosso arquivos no servidor.

Observe a imagem a seguir:



Como podemos notar, o único arquivo presente na raiz do site é o `index.php`, este mesmo arquivo será responsável apenas por redirecionar o usuário do site para a pasta PHP que possuirá o arquivo responsável por carregar o layout da página.

Outro diretório que também deve receber uma atenção especial é o INCLUDES, nele estão presentes as classes de conexão com banco de dados, template power e uma classe responsável pela manipulação de dados diversos, como conversão de datas e valores.

No subdiretório JS armazenaremos scripts em javascript que serão úteis na loja virtual.

4.4 O layout (template)

Ao utilizarmos a sistemática de Templates com o template power, deveremos primeiramente definir o que será e como será dividido o template principal da página.

No nosso caso, possuiremos uma determinada área do documento que se modificará a cada clique de troca de sessão ou comportamento que for atribuído pelo usuário, a essa área daremos o nome de CONTEUDO, o restante da página nunca se modificará, ou seja, permanecerá sempre com a mesma aparência não importando a ação executada pelo visitante dentro do site.

Uma sugestão de montagem dessa estrutura segue-se abaixo:



Como podemos ver somente o centro do site será modificado, nele exibiremos informações como o carrinho de compras do usuário, os produtos que estão sendo listados, o formulário de compras, etc.

O que podemos abstrair dessa idéia é que poderemos criar um BLOCO no Template que ira unir e incorporar essas informações variadas ao site conforme formos navegando.

Unidade 5

A página inicial

Inicialmente vamos criar um arquivo chamado “**modelo_site.htm**”. Este arquivo será o responsável por unir todos os outros e conterá todo conteúdo tanto dinâmico quanto estático do site. Ele é formado basicamente por um menu superior de 3 botões, os quais apontam para o documento principal (HOME), para a listagem de produtos (PRODUTOS) e por fim para um formulário de contato (CONTATO). Ainda no topo deste mesmo documento teremos um link que será responsável por mostrar os produtos do carrinho de compras do usuário, e logo abaixo um formulário simples para o login no ambiente administrativo do site.



Após a montagem do esqueleto do site, teremos que definir as strings em html que serão substituídas pelo PHP no arquivo **modelo_site.php**. Como estamos trabalhando com uma estrutura que dará liberdade de alteração do layout sempre que desejado pelo cliente, devemos refletir que até mesmo os links do site serão interpretados pelo template. Então na tabela superior do documento atual iremos atribuir os seguintes valores aos links de menu:

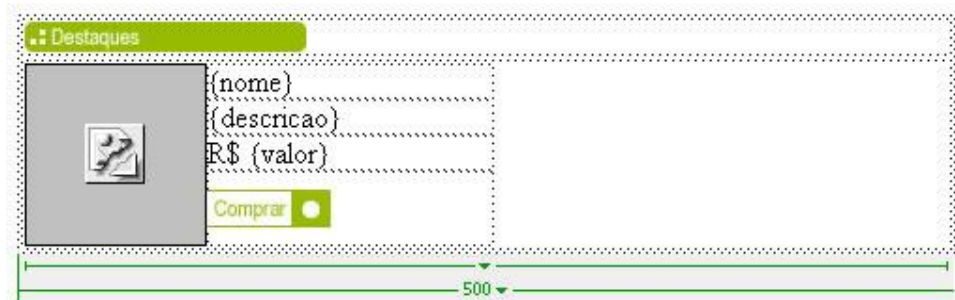
```
Link Home: <a href="{link_home}">
Link Produtos: <a href="{link_produtos}">
Link Contato: <a href="{link_contato}">
Faremos a mesma coisa no link do carrinho de compras:
Link Carrinho: <a href="{link_carrinho}">
```

Agora, na tabela de login não podemos esquecer-nos de colocar criarmos um formulário que receberá os dados de login do administrador e o redirecionará para o ambiente administrativo caso o login tenha sido efetuado com sucesso. Caso contrário, exibiremos uma mensagem nesta mesma tela informando que o usuário e a senha estão incorretos.

Para isso deixaremos uma string chamada {msg_login} na linha superior ao formulário.

No centro do site deveremos listar os produtos que estão configurados para serem destaques, para facilitar a organização criaremos um outro arquivo HTML que será mesclado com o modelo_site.htm e mostrará os resultado final conforme visto no início do projeto.

A esse arquivo daremos o nome de “**destaques.htm**”. O arquivo possuirá a seguinte formatação:



Para cada produto encontrado no banco de dados que se define como destaque, iremos executar um loop de repetição. Quando necessitamos de uma estrutura de repetição em um template logo notamos que devemos criar um novo bloco. Observe que a tabela central do layout acima apresentado é quem será repetida a cada volta do laço.

Podemos deixa-la então dentro de uma tag de loop do template como mostrado abaixo:

```
<!--START BLOCK : PRODUTOS -->
.
.
.
<!--END BLOCK : PRODUTOS -->
```

Após a montagem do arquivo modelo_site.htm e destaques.htm, deveremos agora criar o arquivo php que será capaz de interpretá-lo e exibi-lo na web, mas antes iremos implementar um classe responsável por tratar os dados que serão exibidos para o usuário.

Unidade 6

Arquivos, classes e métodos para a loja

A seguir iremos implementar alguns métodos que serão importantes para a manipulação de dados específicos na nossa loja, como o tratamento de datas e valores numéricos referentes a moeda.

Chamaremos essa classe de classUtil.php, onde os métodos acima será encapsulados, salve-a no diretório INCLUDES do site.

```
class Util {
#Método que Formata um numero decimal
function formataDecimal($numero){
    $numero = str_replace(".", "", $numero);

    $num_vet = explode(",", $numero);

    if(sizeof($num_vet) == 1){
        $numero = $numero . ",00";
    }else{
        if(strlen($num_vet[1]) < 2)
            $numero = $numero . "0";
    }
    return $numero;
}

#Método que converte um decimal para o formato do BANCO
function converteDecimal($numero){
    $converte=explode(",", $numero);
    return $converte[0].".".$converte[1];
}

#Método que converte uma data em formato Português ou
Inglês com validação
function formata_data($data){
    if (strlen($data) > 0 && $data != '0000-00-00'){
        if (ereg ("([0-9]{1,2})/([0-9]{1,2})/([0-9]{4})",
        $data)){
            list ($dia, $mes, $ano) = explode ("/", $data);
            $this->data = ($ano."-".$mes."-".$dia);
        }else{
            list ($ano, $mes, $dia) = explode ("-", $data);
            $this->data = ($dia."/".$mes."/".$ano);
        }
    }
}
```



```
    }  
    return $this->data;  
}else  
    return "";  
}  
}
```

Como iremos manipular o banco de dados é essencial que tenhamos um arquivo responsável para conectar-se ao banco, e que possua uma fácil configuração quando o site for exportado para outro servidor.

Crie um novo arquivo PHP com o nome de CONEXAO.PHP e salve-o no diretório INCLUDES do site.

```
<?php  
#Conecta com o banco de dados  
mysql_connect('localhost','root','') or die ('Erro de  
conexao');  
#Seleciona database  
mysql_select_db('ecommerce');  
?>
```

Unidade 7

O arquivo php inicial

Salve o arquivo atual, mas não o feche, crie agora um novo arquivo com extensão “.PHP” e salve-o no diretório PHP do site da loja, com o nome de “**modelo_site.php**”.

```
include('../includes/classTemplatePower.php');  
include('../includes/conexao.php');  
include('../includes/classUtil.php');
```

Para incorporarmos os dois documentos já montados anteriormente, deveremos primeiro instanciar a classe template power dentro do modelo_site.php, e logo após realizar a junção do arquivo destaques.htm no documento.

```
$util = new Util;  
$tpl = new TemplatePower('../htm/modelo_site.htm');  
$tpl->assignInclude('CONTEUDO', '../htm/destaques.htm');  
$tpl->prepare();
```

A próxima etapa é substituírmos os termos dados aos links do menu pelos documentos que serão vinculados ao site.

```
#===== MONTA OS LINKS DO MENU SUPERIOR =====  
$tpl->assign('link_home', '../php/modelo_site.php');  
$tpl->assign('link_produtos', '../php/produtos.php');  
$tpl->assign('link_contato', '../php/contato.php');  
$tpl->assign('link_carrinho', '../php/carrinho.php');
```

A etapas acima descritas farão parte da maioria dos documentos que iremos confeccionar a seguir, entretanto as consultas ao banco serão bastante diversificadas, e serão colocadas sempre após estas linhas iniciais.

Como estamos implementando o documento principal do site, e este por sua vez possui uma listagem dos produtos que estão catalogados como destaque, devemos realizar a seguinte requisição ao banco de dados via PHP, para exibirmos os registros no documento.

```
#===== SELECIONA OS DESTAQUES DO BANCO =====
$sql = "select produtos.* from produtos
        where produtos.destaque = 'Sim'
        order by rand()";
$rs = mysql_query($sql) or die('Erro de produtos');
$total = mysql_num_rows($rs);

for($x = 0; $x < $total; $x++){
    $tpl->newBlock('PRODUTOS');
    $tpl->assign('nome',mysql_result($rs, $x, 'nome'));
    $tpl->assign('descricao',mysql_result($rs, $x,
'descricao'));
    $tpl->assign('valor',$util-
>formataDecimal(mysql_result($rs, $x, 'valor')));
    $tpl-
>assign('link_comprar','../php/carrinho.php?acao=comprar&i
d_produto='.mysql_result($rs, $x, 'id_produto'));

    if(strlen(mysql_result($rs, $x, 'foto')) == 0)
        $tpl->assign('foto','../fotos/indisponivel.jpg');
    else
        $tpl->assign('foto','../fotos/'.mysql_result($rs,
$x, 'foto'));
}
$tpl->gotoBlock('_ROOT');
```

Mesmo ainda não tendo chegado nesse tópico, já iremos deixar preparado o documento para interceptar a string de mensagem de erro de login que mencionamos anteriormente.

```
if(isset($msg))
    $tpl->assign('msg_login',$msg_login);
```

E por fim executamos o método responsável por criar o documento para o usuário.

```
$tpl->printToScreen();
?>
```

7.1 Redirecionamento para a página inicial

Como foi mencionado anteriormente no diretório raiz do site existirá apenas um arquivo chamado **index.php**, o qual será responsável por redirecionar o visitante para dentro do diretório PHP que possuirá o arquivo de manipulação do template da página inicial.

Ainda com o arquivo acima aberto, crie um arquivo chamado `index.php` e salve-o no diretório raiz do site.

O arquivo `index.php` possuirá apenas a seguinte linha:

```
<?php
header('location:php/modelo_site.php');
?>
```

Unidade 8

Elaboração do arquivo CSS:

Antes de continuarmos com a implementação dos scripts, iremos confeccionar a folha de estilos do site, nesse arquivo serão adicionados alguns seletores responsáveis por manipular e formatar os links, textos e formulários do nosso site.

Vamos criar um novo arquivo CSS, e chama-lo de TOTAL.CSS:

```
body
{
    MARGIN-TOP : 0pt;
    MARGIN-LEFT: 0pt;
    SCROLLBAR-FACE-COLOR: #bfbfbf;
    SCROLLBAR-HIGHLIGHT-COLOR: #bfbfbf;
    SCROLLBAR-SHADOW-COLOR: #bfbfbf;
    SCROLLBAR-3DLIGHT-COLOR: #bfbfbf;
    SCROLLBAR-ARROW-COLOR: #395a4f;
    SCROLLBAR-TRACK-COLOR: #bfbfbf;
    SCROLLBAR-DARKSHADOW-COLOR: #bfbfbf;
}

input,select,textarea
{
    BORDER-RIGHT: #333333 1px outset;
    BORDER-TOP: #333333 1px outset;
    FONT-SIZE: 11px;
    BORDER-LEFT: #333333 1px outset;
    COLOR: #333333;
    BORDER-BOTTOM: #333333 1px outset;
    FONT-FAMILY: Verdana, Arial, Verdana;
    BACKGROUND-COLOR: #ffffff
}

.vermelho11 {
    FONT-WEIGHT: normal;
    FONT-SIZE: 11px;
    COLOR: #FF0000;
    FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;
    TEXT-DECORATION: none;
}

.branco11{
    FONT-WEIGHT: normal;
```

```
    FONT-SIZE: 11px;
    COLOR: #FFFFFFF;
    FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;
    TEXT-DECORATION: none
}
.cinzafortell {
    FONT-WEIGHT: normal;
    FONT-SIZE: 11px;
    COLOR: #777777;
    FONT-FAMILY: "Trebuchet MS";
    TEXT-DECORATION: none;
}
A.linkpreto10 {
    FONT-WEIGHT: normal;
    FONT-SIZE: 10px;
    COLOR: #000000;
    FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;
    TEXT-DECORATION: none;
}
A.linkpreto10:hover {
    FONT-WEIGHT: normal;
    FONT-SIZE: 10px;
    COLOR: #000000;
    FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;
    TEXT-DECORATION: underline;
}
A.linkbranco11 {
    FONT-WEIGHT: normal;
    FONT-SIZE: 11px;
    COLOR: #FFFFFFF;
    FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;
    TEXT-DECORATION: none;
}
A.linkbranco11:hover {
    FONT-WEIGHT: normal;
    FONT-SIZE: 11px;
    COLOR: #FFFFFFF;
    FONT-FAMILY: Verdana, Arial, Helvetica, sans-serif;
    TEXT-DECORATION: underline;
}
```

Agora que possuímos as formatações para os componentes principais do site, continuaremos com as implementações dos scripts da Loja.

Unidade 9

A listagem de produtos

No menu superior da loja, o usuário poderá também selecionar os produtos que estão disponíveis para a venda, sendo que esta seleção pode ser feita através de uma filtragem de tipos ou palavras chaves que procuram estes itens na nossa base de dados.

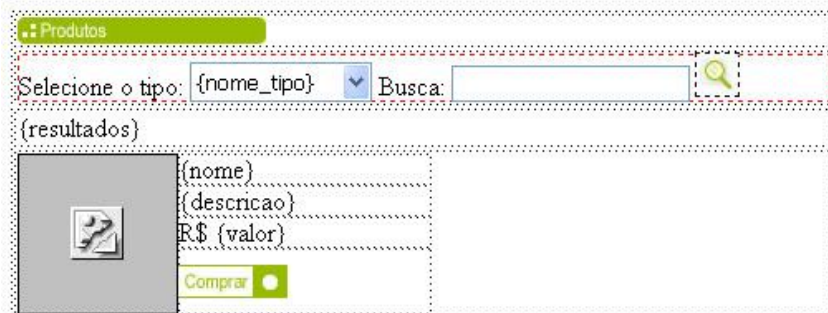


Primeiramente teremos que montar o arquivo de template da página, sendo que como podemos observar os tipos de produtos serão recuperados através de uma consulta feita na tabela TIPOS_PRODUTOS da base de dados.

Crie então um novo arquivo HTML e salve-o como **produtos.htm** na pasta htm do nosso site.

Este documento será equivalente ao confeccionado anteriormente (destaques.htm) porem possuirá um template específico para interpretá-lo, o que veremos no próximo tópico.

O layout do arquivo será como mostrado abaixo:



Dentro do campo de filtragem por tipo e por busca, teremos dois componentes para o formulário, um MENU LIST e um campo do TIPO TEXT.

O formulário submeterá os dados ao mesmo documento, ou seja, teremos que testar se foi feita alguma filtragem antes de mostrar os produtos.

A montagem do MENU LIST de tipos de produtos será feita através de blocos do template, o qual chamaremos de BLOCO TIPOS.

```
<!--START BLOCK : TIPOS -->
```

```
.
```



```
.  
.   
 <!--END BLOCK : TIPOS -->
```

Abaixo do formulário iremos mostrar a quantidade de registros encontrados na busca.

Portanto, adicionaremos a tag {resultados} no template.

A exibição dos registros, juntamente da foto e dos dados do produto serão mostrados com a mesma sistemática abordada nos destaques, utilizando-se outro loop em bloco, chamado BLOCO PRODUTOS.

```
<!--START BLOCK : PRODUTOS -->  
.   
.   
.   
<!--END BLOCK : PRODUTOS -->
```

Montado o template, vamos agora implementar o script php que será responsável por interpretar o arquivo html e exibir os produtos encontrados na tela.

Crie um arquivo novo chamado **produtos.php** e salve-o no diretório php da loja.

```
<?php  
include('../includes/classTemplatePower.php');  
include('../includes/conexao.php');  
  
$tpl = new TemplatePower('../htm/modelo_site.htm');  
$tpl->assignInclude('CONTEUDO','../htm/produtos.htm');  
$tpl->prepare();  
  
#===== MONTA OS LINKS DO MENU SUPERIOR =====  
$tpl->assign('link_home','../php/modelo_site.php');  
$tpl->assign('link_produtos','../php/produtos.php');  
$tpl->assign('link_contato','../php/contato.php');  
$tpl->assign('link_carrinho','../php/carrinho.php');  
  
#===== MOSTRA OS TIPOS DE PRODUTO PARA SELEÇÃO =====  
$sql = "select tipo_produtos.* from tipo_produtos  
        order by tipo_produtos.nome";  
$rs = mysql_query($sql) or die('Erro de tipos');  
$total = mysql_num_rows($rs);  
  
for($x = 0; $x < $total; $x++){  
    $tpl->newBlock('TIPOS');
```

```
$tpl->assign('id_tipo',mysql_result($rs, $x,
'id_tipo'));
$tpl->assign('nome_tipo',mysql_result($rs, $x,
'nome'));
}
$tpl->gotoBlock('_ROOT');
#===== SELECIONA OS PRODUTOS DO BANCO =====
if(!isset($busca)){
    $busca = '';
}else{
    $resultado = 'Foram encontrados: ';
}
#caso o usuario deixa a opção SELECIONE no tipo
if(isset($id_tipo) && $id_tipo == '00')
    unset($id_tipo);

if(!isset($id_tipo)){
    if(isset($resultado))
        $sql_mais = '';
    else
        $sql_mais = 'limit 0,3';

    $sql = "select produtos.* from produtos
            where produtos.nome like '%$busca%'
            or produtos.descricao like '%$busca%'
            order by rand()
            $sql_mais
            ";
}else{
    $sql = "select produtos.* from produtos
            where produtos.id_tipo = $id_tipo
            and (produtos.nome like '%$busca%'
            or produtos.descricao like
            '%$busca%')
            order by produtos.nome";
}

$rs = mysql_query($sql) or die('Erro de produtos');
$total = mysql_num_rows($rs);

if(isset($resultado)){
    $resultado .= $total . ' produto(s)';
    $tpl->assign('resultados',$resultado);
}
```

```
}

for($x = 0; $x < $total; $x++){
    $tpl->newBlock('PRODUTOS');
    $tpl->assign('nome',mysql_result($rs, $x, 'nome'));
    $tpl->assign('descricao',mysql_result($rs, $x,
'descricao'));
    $tpl->assign('valor',mysql_result($rs, $x, 'valor'));
    $tpl-
>assign('link_comprar','../php/carrinho.php?acao=comprar&i
d_produto='.mysql_result($rs, $x, 'id_produto'));

    if(strlen(mysql_result($rs, $x, 'foto')) == 0)
        $tpl->assign('foto','../fotos/indisponivel.jpg');
    else
        $tpl->assign('foto','../fotos/'.mysql_result($rs,
$x, 'foto'));
}
$tpl->gotoBlock('_ROOT');

$tpl->printToScreen();
?>
```

Como você pode ver na análise do código php apresentado, a variável “\$sql_mais” é uma espécie de complemento que é adicionado à query de consulta do banco caso não tenha sido efetuada nenhuma pesquisa por produtos pelo usuário, mostrando assim apenas três registros randômicos da tabela de produtos quando a seção é acessada pela primeira vez.

Unidade 10

O formulário de contato

Todo e qualquer site da web atualmente na maioria dos casos possui um formulário de contato para facilitar a comunicação entre seus usuários e a empresa.

Outra vantagem da utilização desse método de comunicação é que o envio da mensagem será feito via serviço de email do servidor, não necessitando que o usuário possua um cliente de email instalado e configurado na estação que esta acessando o website. Além do mais, a facilidade em poder documentar este contato armazenando-o em uma base de dados também é um dos pontos fortes a serem considerados.

Vamos agora criar um simples formulário de contato para a nossa loja, aproveitando-se de alguns recursos do php para realizarmos a validação do documento.

Crie um arquivo novo chamado **contato.htm** e salve-o no diretório htm do nosso site.

O layout desse documento será como mostrado abaixo:

Cuide para nomear corretamente os campos do formulário para que possamos acessá-los sem erros no nosso arquivo php.

Além dos campos mostrados no formulário, temos ainda uma variável oculta (hidden) chamada “enviar” na qual atribuímos o valor “sim” no formulário.

Montado esse arquivo vamos agora implementar dois arquivos php, primeiramente implementaremos o **contato.php** que será responsável por trabalhar com o template htm que acabamos de montar, e juntamente confeccionaremos o arquivo **mensagem.php** que servirá como uma espécie de retorno das validações que criaremos.

Crie um arquivo novo em php e salve-o como **contato.php** no diretório php do servidor, veja abaixo como ficará o script:

```
<?php
include('../includes/classTemplatePower.php');

$tpl = new TemplatePower('../htm/modelo_site.htm');
```

```
$tpl->assignInclude('CONTEUDO','../htm/contato.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_home','../php/modelo_site.php');
$tpl->assign('link_produtos','../php/produtos.php');
$tpl->assign('link_contato','../php/contato.php');
$tpl->assign('link_carrinho','../php/carrinho.php');

#===== VALIDAÇÃO DOS CAMPOS =====
#TESTA SE A VARIÁVEL OCULTA(HIDDEN) CHAMADA 'ENVIAR'
CHEGOU,
#CASO SIM, É SINAL QUE O USUÁRIO APERTOU NO SUBMIT
if(isset($enviar)){
    $msg = '';
    if(strlen($nome) == 0)
        $msg .= '<li>Nome não foi preenchido!<br>';
    if(strlen($email) == 0)
        $msg .= '<li>Email não foi preenchido!<br>';
    if(strlen($mensagem) == 0)
        $msg .= '<li>Mensagem não foi preenchida!<br>';

    if(strlen($msg) > 0){
        header("location:../php/mensagem.php?msg=$msg");
    }else{
        #Cabeçalhos para envio de email
        #REMETENTE
        $headers = "From: $email";
        #VERSAO DO SCRIPT DE MENSAGEM
        $headers .= "MIME-Version: 1.0";
        #CONTEUDO TEXTO/HTML - CHARSET = ACENTOS
        $headers .= "Content-Type: text/html; charset=ISO-
8859-1";
        #CRIPTOGRAFIA DE ENVIO
        $headers .= "Content-Transfer-Encoding: base64";

        $texto = "<font face=\"verdana\" size=\"2\">";
        $texto .= "Nome: $nome <br>";
        $texto .= "Email: $email <br>";
        $texto .= "Mensagem: $mensagem";
    }
}
```

```
mail('cicero@alfamidia.com.br', 'Contato
Site', $texto, $headers);
$msg = 'Obrigado pela sua visita, em breve
entraremos em contato';
header("location:../php/mensagem.php?msg=$msg");
}
}
$tpl->printToScreen();
?>
```

Analisando o script acima, notamos nas linhas iniciais, o teste do envio da variável “enviar”, que caso seja detectado inicia uma série de testes em cada campo do formulário, verificando se o número de caracteres presentes é maior do que 1. Caso algum dos campos não possua preenchimento é adicionada uma string de caracteres com uma informação à variável “\$msg” que logo após é testada e caso possua tamanho de caracteres maior que zero, encerra a execução dos processos e remete o usuário para um outro documento chamado **mensagem.php**. Este tipo de artifício é muito utilizado como validação no php, ou seja, aplicamos uma série de regras e testes à variáveis que queremos validar, e caso o resultado desejado não seja encontrado acabamos armazenando nosso erro ou mensagem em uma variável específica.

Logo após a série de testes ser efetuada e a consistência dos dados ter sido aprovada, passamos para um segundo passo, que é o envio das informações por email para o contato do site.

Este processo utiliza-se da função “mail” do php que exige somente que um serviço de envio de mensagens esteja habilitado e configurado no servidor que hospeda o site.

Antes do envio são criadas algumas regras de cabeçalho que permitem o uso de tags HTML dentro do email e padroniza o envio da mensagem em formato que seja interpretado corretamente pelos clientes de email.

Após a conclusão do processo é mostrada uma mensagem de agradecimento ao usuário.

Vamos agora implementar o nosso arquivo de validação, que será utilizado em todos os formulários do site.

Inicialmente crie um arquivo chamado **mensagem.htm** e salve-o no diretório htm do seu site.

O layout do arquivo deverá ficar como mostrado abaixo:



Nesse arquivo teremos apenas uma tag de template que será substituída pelo script, a tag {msg}.

Lembrando que o envio de caracteres para a substituição desta tag será feito via método GET do formulário, pois utilizamos a função `header("location:.")` do php para este processo, portanto é importante cuidar para que o limite de caracteres não ultrapasse os 255 que é o suportado pelo cabeçalho HTTP.

Crie agora um arquivo novo chamado `mensagem.php` e salve-o no diretório php do seu site.

O script é simples de ser implementado e ficará como mostrado a seguir:

```
<?php
include('../includes/classTemplatePower.php');

$tpl = new TemplatePower('../htm/modelo_site.htm');
$tpl->assignInclude('CONTEUDO', '../htm/mensagem.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_home', '../php/modelo_site.php');
$tpl->assign('link_produtos', '../php/produtos.php');
$tpl->assign('link_contato', '../php/contato.php');
$tpl->assign('link_carrinho', '../php/carrinho.php');

$tpl->assign('msg', $msg);

$tpl->printToScreen();
?>
```

Pronto, agora já possuímos o documento que irá informar ao usuário se algum campo do formulário não foi preenchido corretamente, ou se simplesmente algum processo de envio ou de cadastro foi concluído.

Veja abaixo um exemplo de retorno gerado pelo documento **mensagem.php**.



Unidade 11

A implementação do carrinho de compras

Dentre os processos de implementação de uma loja virtual, o carrinho de compras é o que mais se destaca pela necessidade de ser um script ágil e seguro para com as compras do clientes e seus dados.

Você já viu o fluxograma de processo de compras que foi apresentado anteriormente, mas antes de começarmos a implementação envolvendo os processos do fluxograma iremos conhecer um pouco sobre a lógica introdutória do processo para assim termos mais facilidade na implementação dos nossos scripts.

Para o cliente, tanto a tecnologia, quanto a forma de implementação da loja é transparente, ou seja, não importa em qual linguagem, banco de dados ou metodologia a solução foi desenvolvida, e sim somente se ele conseguirá encontrar de forma rápida e eficaz o que precisa, solicitar a compra e receber o produto.

Entretanto no lado do desenvolvedor temos como a performance do banco de dados e a segurança tanto dos dados da loja quanto os dados do cliente devem ser levados em consideração.

Abaixo seguem uma série de questões relevantes que devem ser analisadas:

11.1 Como os produtos serão colocados dentro do carrinho?

O fato de colocar os produtos dentro do carrinho de compras irá utilizar de uma forma usada na maioria das estruturas de venda online na internet. Imagine que a cada minuto seu site poderá estar sendo acessado por “n” clientes, de “n” locais diferentes do planeta. Você não pode prever quantos produtos cada cliente irá comprar, e nem mesmo saber se um cliente irá finalizar a compra dele. Fatores estes que devem ser levados em consideração quando o assunto é acesso aos dados do servidor.

Se cada produto for tratado como um item e cada item inserido no carrinho fosse diretamente inserido em uma tabela do nosso banco de dados, sem nem mesmo o cliente ter concluído a compra, teríamos uma queda de performance e um estouro do limite de conexões simultâneas do banco rapidamente.

Portanto, a melhor forma de trabalharmos com o carrinho de compras online é o uso das sessões. Como você já deve ter visto uma sessão quando iniciada no php vale somente enquanto o navegador do usuário estiver em execução. Caso o mesmo seja finalizado os dados armazenados em sessão são completamente perdidos não podendo ser recuperados, isso ocasiona em mais segurança e mais performance no acesso aos dados da loja.

11.2 Como armazenar os produto no carrinho?

Os produtos que forem comprados deverão ser armazenados em variáveis de sessão, mas como é imprevisível o número de produtos que será comprado por cada cliente, não podemos possuir uma variável para cada produto selecionado, logo a primeira idéia que nos surge para sanarmos este problema é o uso de vetores, vetores de sessão.

Para compreender melhor veja a figura abaixo:



Mas e se o cliente tivesse comprado o mesmo produto mais de uma vez? Acabariamos tendo a seguinte situação:



Imagine um vetor com 120 posições, onde destas 120, existissem apenas 5 produtos distintos. Estariamos desperdiçando tempo gerenciando produtos em demasia, e no momento de inclusão destes dados no banco teríamos um algoritmo redundante e inadequado para a ocasião.

11.3 Como controlar os produtos distintos por quantidade?

Existem três formas práticas para controlarmos a quantidade de produtos distintos em nosso carrinho:

Criar um vetor adjacente: um vetor de sessão que possuirá o mesmo tamanho que o vetor de compras, porem possuindo apenas a quantidade dos produtos armazenados dentro de suas posições. A desvantagem da utilização desse método é que teremos que ter

cuidado na manipulação de dois vetores, sempre mantendo a atenção na manipulação dos dados e das posições dos mesmos.

Armazenar o produto e a quantidade dentro do mesmo vetor, porem separados por virgulas: forma simples e eficaz de solução do problema, porem deveremos ficar atentos à separação destes valores quando formos inseri-los no banco de dados, lembrando que a virgula poderia vir a atrapalhar na manipulação destes dados sendo simplesmente tratada como um numero inteiro.

Utilizar um vetor bidimensional: a forma mais clássica e eficaz, nos permite que tenhamos separados os valores de quantidade de produtos, permitindo acesso direto a cada um deles sem muito esforço de implementação.

Selecionaremos então a ultima alternativa, criando um vetor de sessão bidimensional para o controle de produtos do carrinho e suas respectivas quantidades, lembrando que como o acesso é feito via sessão do navegador, cada usuário conectado no site possuirá o seu carrinho de compras com seus produtos e quantidades, sendo estes levados para o banco de dados somente na finalização da compra.

Veja abaixo a estrutura de um vetor bidimensional:

\$VETOR_CARRINHO				
LINHAS	PRODUTO 1	PRODUTO 2	PRODUTO 5	PRODUTO 9
	2	50	23	1
COLUNAS				

Para acessar as informações destes dados manteremos a mesma sintaxe de acesso de um vetor comum, porem informando sempre a COLUNA e a LINHA que iremos manipular, como se fosse um jogo de batalha naval.

Veja o exemplo abaixo, onde acessamos o produto 5 e sua quantidade no vetor:

```
#COLUNA 0 LINHA 0 = PRODUTO
$VETOR_CARRINHO[0][0] = "Produto 1";
#COLUNA 0 LINHA 1 = QUANTIDADE
$VETOR_CARRINHO[0][1] = 1;
```

11.4 O produto será armazenado pelo nome do vetor?

Os nomes dos produtos podem de acordo com a estrutura do banco podem possuir até 100 caracteres, é desnecessário que armazenemos todo esse dado no vetor, por isso a melhor maneira para evitarmos essa situação é armazenarmos a chave primária do produto, o identificador único de cada produto da loja.

Quando formos percorrer o vetor iremos simplesmente consultar o id do produto no banco de dados e assim montarmos o resumo da compra.

Unidade 12

O script do carrinho de compras

Como você já deve ter notado, nos scripts de listagem de produtos e no script inicial em que aparecem os destaques do site, foi criado um botão COMPRAR que direciona o usuário para um documento chamado **carrinho.php**. Este script será responsável por manipular todas as operações referentes ao carrinho de compras da loja.

Existem quatro operações principais que serão feitas neste script:

Inserção: insere produtos no vetor de sessão do carrinho de compras.

Atualização: atualiza a quantidade e o calculo geral de valores do carrinho de compras.

Exclusão: exclui determinado produto do vetor de sessão do carrinho de compras.

Mostrar: sempre que o script for executado seu processo final será mostrar os produtos existentes no carrinho de compras.

A tela do carrinho de compras será exibida sempre que o usuário comprar um produto, ou ao clicar no ícone MEU CARRINHO localizado no topo do site junto do cabeçalho.

Veja a seguir a tela final do carrinho de compras:

Carinho de Compras

Você tem : 11 produto(s) no carrinho!

Produto	Valor Un.	Qtd.	Total	Excluir
Mouse Optico	65,78	11	723,58	X

TOTAL GERAL : 723,58

:: Atualizar Carrinho ::

<< Continuar Comprando Finalizar Compras >>

Como você pode observar o usuário pode além de excluir o produto atualizar a quantidade de itens no carrinho, isso evita que ele necessite clicar mais de uma vez sobre o produto que seja comprar para incrementar a quantidade de itens.

Após esse procedimento o usuário decide se continuará comprando, ou se deseja finalizar a compra.

Caso seja selecionada a primeira opção o mesmo será direcionado para a tela de seleção de produtos, caso contrário será apresentado um formulário para o usuário se logar ou se cadastrar, finalizando assim as compras.

Vamos então montar a tela de template do carrinho de compras, crie um arquivo novo em formato HTML e salve-o como **carrinho.htm** no diretório htm do site.

O layout do documento deverá ser da seguinte forma:

Onde teremos um formulário que possuirá uma tabela interna para a apresentação dos produtos.

No topo desta tabela inicialmente será apresentada uma contagem de itens do carrinho, utilizando-se a tag {itens}, onde substituiremos pela quantidade de itens disponíveis no vetor de sessão das compras multiplicado pela quantidade de produtos.

Da mesma forma que usávamos blocos para exibir os produtos nas telas anteriores, utilizaremos um loop de bloco neste template para exibir cada registro do nosso vetor. Chamaremos esse loop de BLOCO PRODUTOS e dentro dele teremos quatros tags de template: {nome_produto}, {valor}, {qtd} e {total_prod}.

Como possuímos na linha inicial do vetor de sessão o id_produto do produto que o usuário inseriu no carrinho de compras, fica fácil acessarmos os outros dados acima mencionados recuperando-os do banco de dados através de uma simples query de consulta.

A tag de template {total_geral} fica de fora deste bloco e é assignada no final da repetição através de uma variável auxiliar que contabiliza os dados de preço X quantidade no momento da exibição dos produtos do carrinho na tela.

Vamos agora implementar o script de carrinho de compras para interpretar o template já montado anteriormente.

Crie um arquivo novo em php e salve-o como carrinho.php dentro do diretório php do seu site.

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');
include('../includes/classUtil.php');

$util = new Util;
$tpl = new TemplatePower('../htm/modelo_site.htm');
$tpl->assignInclude('CONTEUDO', '../htm/carrinho.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
```

```
$tpl->assign('link_home','../php/modelo_site.php');
$tpl->assign('link_produtos','../php/produtos.php');
$tpl->assign('link_contato','../php/contato.php');
$tpl->assign('link_carrinho','../php/carrinho.php');

#===== MANIPULAÇÃO DO CARRINHO DE COMPRAS =====
#VERIFICA SE ESTA SENDO ENVIADA ALGUMA OPERAÇÃO NO
CARRINHO
session_start();

if(isset($acao)){
    switch($acao){
        case 'comprar':
            if(!isset($VETOR_CARRINHO)){
                session_register("VETOR_CARRINHO");
                $VETOR_CARRINHO[0][0] = $id_produto;
                $VETOR_CARRINHO[0][1] = 1;
            }else{
                #PERCORE O CARRINHO VERIFICANDO SE O
PRODUTO A SER INSERIDO
                #JA NAO EXISTE DENTRO, CASO SIM:
INCREMENTA A QTD. EM +1
                #CASO NAO, INSERE NO FIM DO VETOR
                $tamanho = sizeof($VETOR_CARRINHO);

                for($x = 0; $x < $tamanho;$x++){
                    if($VETOR_CARRINHO[$x][0] ==
$id_produto){
                        $VETOR_CARRINHO[$x][1] =
($VETOR_CARRINHO[$x][1]) + 1;
                        $x = $tamanho;
                        $ins = 'sim';
                    }
                }

                if(!isset($ins)){
                    $VETOR_CARRINHO[$tamanho][0] =
$id_produto;
                    $VETOR_CARRINHO[$tamanho][1] =
1;
                }
            }
        }
    }
}
```

```
        break;

        case 'atualizar':
            for($x = 0; $x < sizeof($qtd); $x++){
                $VETOR_CARRINHO[$x][1] = $qtd[$x];
            }
            break;

        case 'excluir':
            $tamanho = sizeof($VETOR_CARRINHO);
            if($posicao != $tamanho-1){
                for($x = $posicao; $x < $tamanho-1;
                $x++){
                    $VETOR_CARRINHO[$x][0] =
$VETOR_CARRINHO[$x+1][0];
                    $VETOR_CARRINHO[$x][1] =
$VETOR_CARRINHO[$x+1][1];
                }
            }
            array_pop($_SESSION['VETOR_CARRINHO']);
            break;

    }
}

#MOSTRA TODOS OS PRODUTOS DO CARRINHO
#CRIAR UMA VARIÁVEL DE TOTALGERAL VAZIA;
$total_geral = '';
$qtd_geral = '';
#POREM VERIFICA SE EXISTE ALGO PARA MOSTRAR ANTES
if(isset($VETOR_CARRINHO)){
    for($x = 0; $x < sizeof($VETOR_CARRINHO); $x++){
        $tpl->newBlock('PRODUTOS');
        $sql = "select produtos.* from produtos
                where produtos.id_produto =
                ".$VETOR_CARRINHO[$x][0];
        $rs = mysql_query($sql) or die('Erro de
produtos');
        $tpl->assign('nome_produto',mysql_result($rs, 0,
'nome'));
        $tpl->assign('valor',$util-
>formataDecimal(mysql_result($rs, 0, 'valor')));
        $tpl->assign('qtd',$VETOR_CARRINHO[$x][1]);
    }
}
```

```
$tpl->assign('link_excluir','../php/carrinho.php?acao=excluir&posicao=$x');

#PEGA A QUANTIDADE DA POSICAO $X DO VETOR E
MULTIPLICA PELO
#SEU VALOR NO BANCO DE DADOS
$total_prod = $VETOR_CARRINHO[$x][1] *
mysql_result($rs, 0, 'valor');
$tpl->assign('total_prod',$util->formataDecimal($total_prod));
$total_geral += $total_prod;

$qtyd_geral += $VETOR_CARRINHO[$x][1];
}
$tpl->gotoBlock('_ROOT');
}else{
    $msg = "Voce não possui produtos no seu carrinho!";
    header("location:../php/mensagem.php?msg=$msg");
}
#ATRIBUI A VER TOTAL_GERAL NO FINAL DO CARRINHO
$tpl->assign('total_geral',$util->formataDecimal($total_geral));
$tpl->assign('itens',$qtyd_geral);
#REFERENTE AOS BOTOS DE FINALIZAR OU CONTINUAR COMPRA
$tpl->assign('link_continuar','../php/produtos.php');
$tpl->assign('link_finalizar','../php/finaliza.php');

$tpl->printToScreen();
?>
```

Note que o script baseia-se simplesmente na manipulação de vetores, tendo como particularidade o vetor bidimensional que criamos no ato da inserção do primeiro produto no carrinho.

O final do arquivo você verá os processos de exibição do vetor do carrinho na tela, o qual como mencionado anteriormente é executado sempre que o script for executado não importando a operação que foi realizada.

A variável “ação” é testada sempre no início do script, portanto, quando o usuário clicar em comprar, atualizar ou excluir a variável muda de valor. Outro ponto importante que deve ser observado é a utilização dos métodos de conversão que criamos na classUtil.php, que tem extrema importância na exibição de valores convertidos como é o caso do preço dos produtos.

Unidade 13

Finalização da compra

Após a escolha dos produtos para a compra o cliente deverá clicar em FINALIZAR COMPRA na tela do carrinho de compras que montamos acima. Este procedimento ira direcioná-lo para outro script o qual chamaremos de **finaliza.php**.

Este script será responsável por perguntar para o usuário se ele já possui cadastro ou se deseja se cadastrar, no primeiro caso os dados de login do usuario serão recuperados do banco de dados, evitando que um novo cadastro seja preenchido, no segundo caso o usuario deverá preencher os campos obrigatório juntamente de uma senha que servirá para futuras compras na loja.

Veja abaixo como ficará o layout do arquivo de template **finaliza.htm**:

Finalização de Compra

JA POSSUO CADASTRO

CPF: Senha:

DESEJO ME CADASTRAR

Nome:

Endereço:

Cidade:

Estado:

CEP: Fone:

Email:

CPF: Senha:

Crie um novo arquivo do tipo HTML e salve-o como finaliza.htm no diretório htm do seu site.

Este arquivo possui dois formulários dentro de uma única tabela, ambos os formulários possuem sua ACTION para o mesmo arquivo, porem passando ações diferentes via GET para o destino.

O primeiro formulário possuirá a seguinte sintaxe na sua ACTION:

```
<form id="form1" name="form1" method="post"
action="../php/finaliza_compra.php?login=sim">
```

O segundo será da seguinte forma:

```
<form id="form3" name="form3" method="post"
action="../php/finaliza_compra.php?login=nao">
```

Da mesma maneira que utilizamos a validação via php no script de contato, iremos utilizar esse artifício para o login ou para o cadastro.

Para isso deveremos implementar o script que utilizará o template que acabamos de montar. Crie um novo arquivo php e salve-o como **finaliza.php** o diretório php da loja.

Como você deve ter notado o documento anterior envia os dados dos formulários para o `finaliza_compras.php`, estes dados de acordo com a ação executaram determinado bloco de processos dentro do script, portanto nosso script atual servirá simplesmente para exibir o template.

Veja o script a seguir:

```
<?php
include('../includes/classTemplatePower.php');

$tpl = new TemplatePower('../htm/modelo_site.htm');
$tpl->assignInclude('CONTEUDO','../htm/finaliza.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_home','../php/modelo_site.php');
$tpl->assign('link_produtos','../php/produtos.php');
$tpl->assign('link_contato','../php/contato.php');
$tpl->assign('link_carrinho','../php/carrinho.php');

$tpl->printToScreen();
?>
```

O foco principal da finalização ficará no script da ACTION dos formulários, este script não utiliza de templates para ser executado, sendo elaborado em php puro. Como é de costume as estruturas de testes sempre executam o teste mais simples em sua primeira execução, deixando os processos mais trabalhosos para o final.

Iremos nesse arquivo verificar se o usuário esta inicialmente tentando efetuar login, verificaremos se o login esta correto e por fim testaremos se a ação executada é um cadastro de um novo usuário a loja.

Crie um novo arquivo php e salve-o como `finaliza_compras.php` no diretório php do site da loja.

A sintaxe do script finaliza_compras.php deverá ficar como mostrada abaixo:

```
<?php
include('../includes/conexao.php');
session_start();

if(isset($login)){
    if($login == 'sim'){
        #VERIFICA SE OS DADOS DO USUARIO EXISTEM NA TABELA DE
CLIENTES
        $sql = "select clientes.* from clientes
                where clientes.cpf = '$cpf' and
clientes.senha = '$senha'";
        $rs = mysql_query($sql) or die(mysql_error());
        $total = mysql_num_rows($rs);

        if($total > 0){
            #SE VERDADEIRO PEGA O ID_CLIENTE PARA A
INSERÇÃO DOS PRODUTOS COMPRADOS NO BANCO DE DADOS
            $id_cliente = mysql_result($rs, 0,
'id_cliente');
        }else{
            $msg = "LOGIN inválidos";

            header("location:../php/mensagem.php?msg=$msg");
        }
    }else{
        #INSERE O CLIENTE NA TABELA DE CLIENTES
        #REALIZANDO A VALIDAÇÃO DOS CAMPOS
        $msg = '';
        if(empty($nome))
            $msg .= "<li>Nome não preenchido!";
        if(empty($endereco))
            $msg .= "<li>Endereco não preenchido!";
        if(empty($cidade))
            $msg .= "<li>Cidade não preenchida!";

        if(empty($cep))
            $msg .= "<li>CEP não preenchido!";
        if(empty($fone))
            $msg .= "<li>Fone não preenchido!";
        if(empty($email))
            $msg .= "<li>Email não preenchido!";
    }
}
```

```
if(empty($cpf))
    $msg .= "<li>CPF não preenchido!";
if(empty($senha))
    $msg .= "<li>SENHA não preenchida!";

#VERIFICA SE NAO EXISTE NENHUM CLIENTE JA USANDO
ESTE CPF
$sql = "select clientes.cpf from clientes
        where clientes.cpf = '$cpf'";
$rs = mysql_query($sql) or die(mysql_error());
$total = mysql_num_rows($rs);

if($total > 0){
    $msg .= "<li><strong>Já existe um cliente
cadastrado com esse CPF!</strong>";
}

if(strlen($msg) > 0){
    header("location:../php/mensagem.php?msg=$msg");
}else{
    #INSERE O CLIENTE NO BANCO
    $sql = "insert into clientes
(nome,endereco,cidade,cep,fone,email,cpf,senha)
        values
('$nome','$endereco','$cidade','$cep','$fone','$email','$c
pf','$senha')";
    mysql_query($sql) or die(mysql_error());

    #RECUPERA O ID DESSE NOVO CLIENTE PARA USO
NA TABELA DE PEDIDOS
    $sql = "select clientes.id_cliente from
clientes
        order by
clientes.id_cliente desc
        limit 0,1";
    $rs = mysql_query($sql) or
die(mysql_error());
    $id_cliente = mysql_result($rs, 0,
'id_cliente');
}
}
```

```
}

#POR SEGURANÇA VERIFICA SE ATÉ AQUI JA SE TEM UM VALOR
PARA ID_CLIENTE
if(isset($id_cliente)){

    #MANIPULA O PEDIDO DO CLIENTE
    $dt_pedido = date("Y-m-d");
    $sql = "insert into pedidos (id_cliente,dt_pedido)
            values ($id_cliente,'$dt_pedido')";
    mysql_query($sql) or die(mysql_error());

    $sql = "select pedidos.id_pedido from pedidos
            order by pedidos.id_pedido desc
            limit 0,1";
    $rs = mysql_query($sql) or die(mysql_error());

    $id_pedido = mysql_result($rs, 0, 'id_pedido');

    #INSERE OS PRODUTOS NA TABELA DE PEDIDOS_PRODUTOS
    $total = 0;
    for($x = 0; $x < sizeof($VETOR_CARRINHO); $x++){
        $id_produto = $VETOR_CARRINHO[$x][0];
        $qtd = $VETOR_CARRINHO[$x][1];

        $sql_prod = "select produtos.valor from produtos
                    where produtos.id_produto
= $id_produto";

        $rs_prod = mysql_query($sql_prod) or
die(mysql_error());

        $valor = mysql_result($rs_prod, 0, 'valor');

        #INSERE DENTRO DO BANCO
        $sql_ped = "insert into produtos_pedidos

(id_pedido,id_produto,valor,qtd) values

($id_pedido,$id_produto,'$valor',$qtd)";
        mysql_query($sql_ped) or die(mysql_error());
        $total += $qtd * $valor;      #CALCULA O TOTAL
```

```
}  
#ATUALIZA O CAMPO TOTAL DA TABELA PEDIDOS  
$sql_ped = "update pedidos set pedidos.total_pedido =  
'$total'  
                                where pedidos.id_pedido =  
$id_pedido";  
mysql_query($sql_ped) or die(mysql_error());  
  
#APOS CONCLUIR A OPERAÇÃO - MATA A SESSAO E ENVIA O  
USUARIO PARA TELA DE FIM  
session_destroy();  
header("location:../php/obrigado.php");  
}  
?>
```

Analisando o script acima notaremos a presença dos processos de validação, e por fim a destruição da sessão do usuário caso as compras tenham sido armazenadas no banco de dados com sucesso. Esta finalização consiste em destruir a sessão por completo, mesmo que o usuário não finalize a execução do navegador os dados presentes no carrinho de compras são eliminados por questão de segurança.

Vamos agora construir o último template do site, chamaremos este arquivo de obrigado.htm, tendo como base o layout abaixo:



Crie também o arquivo obrigado.php que será responsável por exibir a tela de template mostrada acima.

O script possuirá a seguinte sintaxe:

```
<?php
```

```
include('../includes/classTemplatePower.php');

$tpl = new TemplatePower('../htm/modelo_site.htm');
$tpl->assignInclude('CONTEUDO', '../htm/obrigado.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_home', '../php/modelo_site.php');
$tpl->assign('link_produtos', '../php/produtos.php');
$tpl->assign('link_contato', '../php/contato.php');
$tpl->assign('link_carrinho', '../php/carrinho.php');

$tpl->printToScreen();
?>
```

Concluímos então o processo de compras e exibição de produtos na loja, ou seja, todas as funcionalidades disponíveis para o **CLIENTE** foram concluídas.

Vamos agora partir para outro passo importante no sistema de E-Commerce, a administração da loja em um ambiente seguro e confiável.

Unidade 14

A administração da loja

Todo site, tanto voltado para a área de E-Commerce, ou para outro foco possui a necessidade de um ambiente administrativo, um local onde um usuário que não possua conhecimentos de programação e desenvolvimento possa alimentar de forma rápida e fácil o conteúdo do site.

Iremos agora elaborar essa camada para a nossa loja virtual.

14.1 Módulos do ambiente

Os seguinte módulos dinâmicos deverão ser implementados:

- ☐ Cadastro de tipos de produtos;
- ☐ Cadastro de Produtos;
- ☐ Acompanhamento de pedidos.

14.2 Implementando de forma segura

Da mesma maneira que utilizamos uma sessão segura para manter o carrinho de compras do usuário, deveremos criar uma sessão segura para o login no ambiente administrativo.

O login será feito através do documento inicial do site, como você deve lembrar existe um formulário ao lado direito do site que solicita USUARIO e SENHA.

Ajuste então o ACTION deste formulário para o destino **login.php**.

Neste novo arquivo iremos verificar se os dados digitados estão presentes na tabela de usuários do banco de dados, caso contrario uma mensagem de erro é gerada e direcionada para o documento **modelo_site.php** como mostra a imagem abaixo:

Seja bem vindo!

 Gerenciador

Área restrita

Erro de login!!!

Usuário:

Senha:

Caso os dados sejam aceitos o usuário possui seu ID registrado em sessão e é então direcionado para a tela a seguir:



Para evitar o acesso irrestrito aos documentos de manipulação de conteúdo teremos que verificar se o ID do usuário está registrado a cada acesso.

Vamos criar um novo arquivo HTML e salva-lo no diretório htm do nosso site como **modelo_admin.htm**.

O layout deste arquivo deverá ser como mostrado abaixo:



No centro deste documento possuiremos um bloco de inclusão chamado conteúdo:

```
<!--INCLUDE BLOCK : CONTEUDO -->
```

Veja que os dados de login como usuario, data e hora são mostrados no topo do documento, abaixo é criado um menu que irá linkar os outros documentos do ambiente administrativo.

Para os itens do menu daremos os seguintes endereço para as ancoras:

```
Tipos de produtos: <a href="{link_tipos}">
Produtos: <a href="{link_produtos}">
Pedidos: <a href="{link_pedidos}">
Logout: <a href="{link_logout}">
```

Vamos então implementar o script responsável pelo login do usuário no sistema, crie um arquivo novo e salve-o como **login.php** dentro do diretório php do site.

Este script fará a conexão com o banco de dados e irá verificar a consistência dos dados de login, veja a sintaxe do algoritmo abaixo:

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');

#==== VERIFICA SE O USUARIO EXISTE NO BANCO DE DADOS =====
if(isset($usuario) && isset($senha)){
    $sql = "select usuarios.* from usuarios
           where usuarios.usuario = '$usuario'
           and senha = '$senha'";
```

```
$rs = mysql_query($sql) or die(mysql_error() . 'Erro de
usuario');
$total = mysql_num_rows($rs);

#SE ENCONTRAR UM REGISTRO LIBERA PARA O ADMINISTRADOR,
SENAO VOLTA
if($total > 0){
    session_start();
    $ID_USUARIO = mysql_result($rs, 0, 'id_usuario');
    session_register("ID_USUARIO");
    header("location:../php/modelo_admin.php");
}else{
    $msg_login = 'Erro de login!!!';

    header("location:../php/modelo_site.php?msg_login=$msg_
login");
}
}
?>
```

Caso ocorra tudo certo, a sessão é registrada e o modelo_site.php é apresentado.

Atenção: evite registrar dados confidenciais em variáveis de sessão, no caso acima estaos registrando a chave primaria do usuário no banco de dados, não sendo aconselhado registrar dados como o nome de LOGIN ou a SENHA, a não ser que estes estejam criptografados.

Após a execução do script de login, surge a necessidade de criarmos o script do modelo_admin.php, que será a tela de recepção do ambiente administrativo da loja.

Crie um arquivo novo php com o nome de **modelo_admin.php** e salve-o no diretório php da loja.

Este script apenas assignará os valores ao menu do topo e os dados de login do usuario.

Veja a seguir a sintaxe do arquivo:

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');
include('../php/ver_sessao.php');

$tpl = new TemplatePower('../htm/modelo_admin.htm');
$tpl->assignInclude('CONTEUDO','../htm/inicio_admin.htm');
$tpl->prepare();
```

```
#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_tipos','../php/tipos_admin.php');
$tpl->assign('link_produtos','../php/produtos_admin.php');
$tpl->assign('link_pedidos','../php/pedidos_admin.php');
$tpl->assign('link_logout','../php/logout.php');

#MOSTRA O NOME DO USUARIO NO INICIO
$sql = "select usuarios.usuario from usuarios
        where usuarios.id_usuario = $ID_USUARIO";
$rs = mysql_query($sql) or die ('Erro de usuario');

$tpl->assign('usuario',mysql_result($rs,0,'usuario'));
$tpl->assign('data',date("d/m/Y"));
$tpl->assign('hora',date("h:i"));

$tpl->printToScreen();
?>
```

Dentro do bloco CONTEUDO que criamos anteriormente estamos vinculando um arquivo chamado inicio_admin.htm, este arquivo possui uma mensagem de boas vindas, uma sugestão interessante seria colocarmos um aviso de novos contatos e novos pedidos que entraram o dia atual.

Vamos implementá-lo a seguir, criando o arquivo inicio_admin.htm e salvando-o no diretório htm do site.

O layout do arquivo deverá ficar como está sendo mostrado abaixo:

```
Bem vindo ao ambiente administrativo da Loja Virtual

Você sta logado como: {usuario}

Data: {data}
```

Como este arquivo é um bloco vinculado ao documento modelo_admin.htm não precisamos elaborar o documento php que o interpretará.

Unidade 15

Validando a sessão

Como foi mencionado anteriormente, não basta validar o login e a senha do usuário somente no primeiro acesso ao ambiente administrativo, precisamos que a cada url que for restrita revalidar esses dados, como fica inviável solicitarmos os dados a cada clique, criaremos um documento que verifica se a sessão foi iniciada e se a variável de sessão foi criada.

Chamaremos esse arquivo **ver_sessao.php** e salvaremos ele na pasta php do nosso site. A lógica deste algoritmo é simples, ele verifica a existência da variável, caso ela não seja identificada o usuário é direcionado para a tela de login do site novamente.

Veja abaixo a sintaxe do arquivo:

```
<?php
session_start();
if(!isset($ID_USUARIO)){
    $msg = 'Erro de login!';
    header("location:../php/modelo_site.php?msg=$msg");
}
?>
```

Iremos incluir este arquivo em cada documento do administrador.

15.1 Tipos de produtos

Os tipos de produtos são essenciais para uma divisão mais adequadas de itens da loja, ao clicar no link: Tipos de produtos do ambiente administrativo será exibida uma listagem de tipos cadastrados no banco de dados.

Vamos então criar um novo arquivo de template em HTML chamado **tipos_admin.htm** e salve-lo na pasta htm do nosso site.

A aparência deste template deverá ser como mostrada abaixo:

TIPOS DE PRODUTOS		Novo
Nome do Tipo	Ação	
{nome_tipo}	 	

Neste arquivo teremos um loop de tipos chamado BLOCO TIPOS que mostrará cada item da tabela tipo_produtos da nossa base de dados. Um fator interessante é usarmos uma tag

de template chamada {cor} que ficará presentes na tag HTML <TR> da linha que exibe o nome do tipo. Essa tag será substituída por valores hexadecimais de cores intercaladas a cada volta do loop, dando um efeito de separação entre cada registro do banco facilitando assim a leitura dos dados.

Existem ainda três links importantes neste documento, que deverão ser substituídos pelas seguintes tags:

```
Link editar: <a href="{link_editar}">
Link excluir: <a href="{link_excluir}">
Link novo: <a href="{link_novo}">
```

Para interpretar este arquivo iremos agora criar o seu script php, chamado tipo_admin.php e salvando-o no diretório php do nosso servidor.

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');
include('../php/ver_sessao.php');

$tpl = new TemplatePower('../htm/modelo_admin.htm');
$tpl->assignInclude('CONTEUDO','../htm/tipos_admin.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_tipos','../php/tipos_admin.php');
$tpl->assign('link_produtos','../php/produtos_admin.php');
$tpl->assign('link_pedidos','../php/pedidos_admin.php');
$tpl->assign('link_logout','../php/logout.php');

#===== LISTA OS TIPOS DE PRODUTOS =====
$sql = "select tipo_produtos.* from tipo_produtos
        order by tipo_produtos.nome";
$rs = mysql_query($sql) or die('Erro de tipos');
$total = mysql_num_rows($rs);

for($x = 0; $x < $total; $x++){
    $tpl->newBlock('TIPOS');

    #DIVIDE $X POR 2, SE FOR IMPAR ATRIBUI UM VALOR A COR,
    E A LINHA FICA DE UM JEITO
    #CASO CONTRARIO A COR EQUIVALE A OUTRO VALOR
    if($x%2)
        $tpl->assign('cor','#F3F3F3');
    else
```

```

        $tpl->assign('cor','#FFFFFF');

        #PARA CADA BOTAO (EDITAR , EXCLUIR) COLOCAR O LINK
        $tpl-
>assign('link_editar',"../php/tipos_admin_editar.php?acao=
editar&id_tipo=" . mysql_result($rs, $x, 'id_tipo'));
        $tpl-
>assign('link_excluir',"../php/tipos_admin_editar.php?acao
=excluir&id_tipo=" . mysql_result($rs, $x, 'id_tipo'));

        $tpl->assign('nome_tipo',mysql_result($rs, $x,
'nome'));
    }
    $tpl->gotoBlock('_ROOT');

    $tpl-
>assign('link_novo',"../php/tipos_admin_editar.php?acao=no
vo");

    $tpl->printToScreen();
?>

```

15.2 Inserindo e editando um tipo

Ao clicar no botão NOVO uma variável chamada ação é remetida ao documento **tipos_admin_editar.php**, essa variável é analisada e acaba por fazer com que o formulário venha vazio.

Primeiramente vamos criar o documento que conterá o formulário de cadastro e edição de tipos, crie um novo documento HTML e salve-o como **tipos_admin_editar.htm** no diretório htm do nosso site.

O layout do arquivo devera seguir este padrão:

EDIÇÃO / CADASTRO DE TIPOS DE PRODUTOS	
Nome	<input type="text" value="{nome_tipo}"/>
<input type="button" value=":: Salvar ::"/>	

A tag {nome_tipo} ficará dentro do atributo VALUE do campo texto do formulário, sendo que esta vira em branco em caso de inserção e preenchida no caso de uma atualização de registro.

Vamos agora criar o arquivo responsável pela inserção, edição e exclusão de tipos de produtos no site.

Crie um arquivo em php com o nome de **tipos_admin_editar.php** e salve-o no diretório php do site.

Dentre os valores analisados para a variável “ação” neste documento, existe também o de SALVAR, onde verificamos se esta sendo uma edição (UPDATE) ou inserção (INSERT) na base de dados, para isso simplesmente passamos no ACTION do nosso formulário o id_tipo do registro em caso de edição. Caso o id_tipo tenha sido passado saberemos que se trata da edição de um registro.

Veja abaixo a sintaxe do script em php:

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');
include('../php/ver_sessao.php');

$tpl = new TemplatePower('../htm/modelo_admin.htm');
$tpl->assignInclude('CONTEUDO','../htm/tipos_admin_editar.htm');
;
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_tipos','../php/tipos_admin.php');
$tpl->assign('link_produtos','../php/produtos_admin.php');
$tpl->assign('link_pedidos','../php/pedidos_admin.php');
$tpl->assign('link_logout','../php/logout.php');

#===== VERIFICA QUAL A OPERAÇÃO SERÁ REALIZADA =====
switch($acao){
    case 'novo':
        $tpl->assign('nome_tipo','');
        $tpl->assign('acao','../php/tipos_admin_editar.php?acao=salvar');
        break;
    case 'editar':
        $sql = "select tipo_produtos.* from tipo_produtos
                where tipo_produtos.id_tipo =
$id_tipo";
        $rs = mysql_query($sql) or die(mysql_error() .
'Erro de tipo');
        $tpl->assign('nome_tipo',mysql_result($rs, 0,
'nome'));
```

```
#TPL ASSIGN NA ACAO DO FORMULARIO, MOSTRANDO QUE
É UMA EDIÇÃO
$tpl-
>assign('acao',"../php/tipos_admin_editar.php?acao=salvar&
id_tipo=$id_tipo");
break;
case 'excluir':
    #A EXCLUSAO SÓ PODE SER FEITA SE NAO EXISTIR
    NENHUM PRODUTO USANDO O TIPO, CASO EXISTA NAO DEIXAR!!!
    #PRIMEIRO SELECIONA-SE TODOS OS PRODUTOS DESSE
    TIPO, SE ACHAR ALGUM NAO FAZ O DELETE
    $sql = "select produtos.* from produtos
            where produtos.id_tipo =
$id_tipo";
    $rs = mysql_query($sql) or die(mysql_error());
    $total = mysql_num_rows($rs);

    if($total > 0){
        $msg = 'Existem produtos usando este
tipo!';

        header("location:../php/tipos_admin.php?msg=$msg");
    }else{
        $sql = "delete from tipo_produtos where
tipo_produtos.id_tipo = $id_tipo";
        mysql_query($sql) or die(mysql_error());
        header("location:../php/tipos_admin.php");
    }
break;
case 'salvar':
    $msg = '';
    #VALIDAÇÃO DOS CAMPOS
    if(strlen($nome_tipo) == 0)
        $msg = '<li>Nome invalido!';

    if(strlen($msg) > 0){

        header("location:mensagem_admin.php?msg=$msg");
        exit;
    }else{
        if(isset($id_tipo)){
```

```
#SE O ID_TIPO VEIO JUNTO É SINAL QUE
É UM EDICAO (UPDATE NO BANCO), CAOS CONTRARIO É INSERÇÃO
$sql = "update tipo_produtos set nome
= '$nome_tipo'
                                where
tipo_produtos.id_tipo = $id_tipo";
mysql_query($sql) or
die(mysql_error());
    }else{
        $sql = "insert into tipo_produtos
(nome) values ('$nome_tipo')";
        mysql_query($sql) or
die(mysql_error());
    }
    header('location:../php/tipos_admin.php');
}
break;
}

$tpl->printToScreen();
?>
```

Como estamos manipulando os dados através de um formulário e precisamos ter consistência dos dados que estão sendo inseridos na base de dados, realizamos o processo de validação no bloco em que desejamos salvar os dados.

Por questão de apresentação, não podemos usar o mesmo arquivo de mensagem que usamos no ambiente do usuário, pois o layout é um pouco diferente, então acabamos por ter de criar um script e um template específico para exibir as mensagens retornadas no ambiente administrativo.

Como já sabemos a funcionalidade deste script iremos apenas aproveitar o já existente e duplica-lo com o nome de mensagem_admin.htm e mensagem_admin.php respectivamente. Estas e outras vantagens do uso do template facilitam a nossa implementação e o ganho de tempo em diversos casos.

Ajustaremos apenas o layout do arquivo mensagem_admin.htm para uma largura maior. Já no script ajustaremos a linha de bloco e de template.

Veja abaixo o script mensagem_admin.php como deverá ser implementado:

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');
include('../php/ver_sessao.php');
```

```
$tpl = new TemplatePower('../htm/modelo_admin.htm');
$tpl->assignInclude('CONTEUDO', '../htm/mensagem_admin.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_tipos', '../php/tipos_admin.php');
$tpl->assign('link_produtos', '../php/produtos_admin.php');
$tpl->assign('link_pedidos', '../php/pedidos_admin.php');
$tpl->assign('link_logout', '../php/logout.php');

$tpl->assign('msg', $msg);




$tpl->printToScreen();
?>
```

15.3 Produtos

Após a conclusão do cadastro de tipos de produtos, deveremos então partir para o próximo passo, o cadastro e manutenção dos produtos da loja. Estes scripts serão similares ao do modulo de tipos, porem com uma particularidade: o upload de imagens no servidor. Para realizarmos o upload de arquivos em php deveremos levar duas coisas em consideração:

- O tipo de arquivo deverá ser verificado, imagens suportadas somente em formato gif, jpg e png.
- O diretório que será salvo as imagens deverá estar com direitos de escrita liberado para o usuário comum.
- A imagem nunca é salva com o nome original, renomeamos o arquivo para um nome consistente e que se baseie na sua identificação no banco de dados.
- A primeira tela a ser exibida quando clicamos em produtos do site é a listagem de produtos, de qualquer forma aproveitaremos o script de listagem de tipos, e criaremos um novo arquivo chamado: **produtos_admin.htm** salvando-o no diretório htm do servidor.

O layout do arquivo deverá ficar como mostrado abaixo:

PRODUTOS			
{msg}			
	Nome do Produto	Valor	Ação
	{nome_produto}	R\$ {valor}	 

Teremos aqui um bloco chamado BLOCO PRODUTOS e o atributo de cor da linha atribuído da mesma forma que fizemos na listagem de tipos de produtos anteriormente.

Os links presentes são:

```
Link editar: <a href="{link_editar}">
Link excluir: <a href="{link_excluir}">
Link novo: <a href="{link_novo}">
```

Para interpretarmos o template, vamos criar um novo arquivo chamado **produtos_admin.php** e salva-lo no diretório php do nosso site.

Veja abaixo a sintaxe do script mencionado:

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');
include('../php/ver_sessao.php');

$tpl = new TemplatePower('../htm/modelo_admin.htm');
$tpl->assignInclude('CONTEUDO','../htm/produtos_admin.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_tipos','../php/tipos_admin.php');
$tpl->assign('link_produtos','../php/produtos_admin.php');
$tpl->assign('link_pedidos','../php/pedidos_admin.php');
$tpl->assign('link_logout','../php/logout.php');

#===== LISTA OS PRODUTOS =====
$sql = "select produtos.* from produtos
        order by produtos.nome";
$rs = mysql_query($sql) or die('Erro de produtos');
$total = mysql_num_rows($rs);

for($x = 0; $x < $total; $x++){
    $tpl->newBlock('PRODUTOS');

    #DIVIDE $X POR 2, SE FOR IMPAR ATRIBUI UM VALOR A COR
    if($x%2)
        $tpl->assign('cor','#F3F3F3');
    else
        $tpl->assign('cor','#FFFFFF');
```

```
#PARA CADA BOTAO (EDITAR , EXCLUIR) COLOCAR O LINK
$tpl-
>assign('link_editar',"../php/produtos_admin_editar.php?acao=editar&id_produto=" . mysql_result($rs, $x, 'id_produto'));
$tpl-
>assign('link_excluir',"../php/produtos_admin_editar.php?acao=excluir&id_produto=" . mysql_result($rs, $x, 'id_produto'));

$tpl->assign('nome_produto',mysql_result($rs, $x, 'nome'));
$tpl->assign('valor',mysql_result($rs, $x, 'valor'));
}
$tpl->gotoBlock('_ROOT');

$tpl-
>assign('link_novo',"../php/produtos_admin_editar.php?acao=novo");

$tpl->printToScreen();
?>
```

15.4 Inserindo e editando um produto

O processo de inserção, edição e exclusão de um produto funcionará da mesma forma que vimos com os tipos, sendo que a variável “ação” possuirá quatro estados que serão analisados no processo de submissão do formulário de manutenção dos produtos. O ultimo deles é o processo de salvar os dados do registro o banco, onde iremos validar as informações que estão sendo passadas antes de finalizar o processo, e caso haja alguma inconsistência nesses valores iremos direcionar o usuário para o arquivo mensagem_admin.php informando que está errado nesses itens.

Outro fator importante que veremos no script é a utilização de funções para a manipulação de arquivos via php e o tratamento que damos para vinculá-los a um registro específico na base de dados.

Crie um novo arquivo php e nomeie de **produtos_admin.php**, salvando-o no diretório php do servidor.

A sintaxe do arquivo é mostrada a seguir:

```
<?php
include('../includes/classTemplatePower.php');
```

```
include('../includes/conexao.php');
include('../php/ver_sessao.php');

$tpl = new TemplatePower('../htm/modelo_admin.htm');
$tpl->assignInclude('CONTEUDO','../htm/produtos_admin_editar.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_tipos','../php/tipos_admin.php');
$tpl->assign('link_produtos','../php/produtos_admin.php');
$tpl->assign('link_pedidos','../php/pedidos_admin.php');
$tpl->assign('link_logout','../php/logout.php');

#===== VERIFICA SE É UMA EDIÇÃO DE PRODUTO E FAZ COM QUE O TIPO JA VENHA SELECIONADO
if(isset($acao) && $acao == 'editar'){
    $sql = "select produtos.id_tipo from produtos
            where produtos.id_produto = $id_produto";
    $rs = mysql_query($sql) or die(mysql_error());
    $id_tipo = mysql_result($rs, 0, 'id_tipo');
}

#===== MONTA A CAIXA DE SELEÇÃO DE TIPOS NO TOPO DA PAGINA =====
$sql = "select tipo_produtos.* from tipo_produtos
        order by tipo_produtos.nome";
$rs = mysql_query($sql) or die(mysql_error());
$total = mysql_num_rows($rs);

for($x = 0; $x < $total; $x++){
    $tpl->newBlock('TIPOS');
    $tpl->assign('nome_tipo',mysql_result($rs, $x, 'nome'));
    $tpl->assign('id_tipo',mysql_result($rs, $x, 'id_tipo'));

    if(isset($id_tipo) && $id_tipo == mysql_result($rs, $x, 'id_tipo'))
        $tpl->assign('marcar', 'selected');
}
```

```
$tpl->gotoBlock('_ROOT');

#===== VERIFICA QUAL A OPERAÇÃO SERÁ REALIZADA =====
switch($acao){
    case 'novo':
        $tpl->assign('nome_produto','');
        $tpl->assign('descricao','');
        $tpl->assign('valor','');
        $tpl->assign('acao',"../php/produtos_admin_editar.php?acao=salvar");
        break;
    case 'editar':
        $sql = "select produtos.* from produtos
                where produtos.id_produto = $id_produto";
        $rs = mysql_query($sql ) or die(mysql_error() . 'Erro de produto');
        $tpl->assign('nome_produto',mysql_result($rs, 0, 'nome'));
        $tpl->assign('descricao',mysql_result($rs, 0, 'descricao'));
        $tpl->assign('valor',mysql_result($rs, 0, 'valor'));

        if(mysql_result($rs, 0, 'destaque') == 'Sim')
            $tpl->assign('seleciona','checked');

        #VERIFICA SE O PRODUTO TEM FOTO CASO POSSUA FAZ COM QUE ESTA APAREÇA NA TELA
        if(strlen(mysql_result($rs, 0, 'foto'))>0)
            $tpl->assign('foto',"<img src=\"../fotos/".mysql_result($rs,0,'foto')."\" width=\"85\">");

        #TPL ASSIGN NA ACAO DO FORMULARIO, MOSTRANDO QUE É UMA EDIÇÃO
        $tpl->assign('acao',"../php/produtos_admin_editar.php?acao=salvar&id_produto=$id_produto");
        break;
    case 'excluir':
        #ANTES DE EXCLUIR O PRODUTO EXCLUI A SUA FOTO
```



```
$sql = "select produtos.foto from produtos
      where produtos.id_produto =
$id_produto";
$rs = mysql_query($sql) or die(mysql_error());
$foto = mysql_result($rs, 0, 'foto');
#VERIFICA SE TINHA FOTO CADASTRADA PARA O PRODUTO
A SER EXCLUIDO
if(strlen($foto)>0)
    unlink("../fotos/$foto");

#APOS O PROCESSO EXCLUI O PRODUTO
$sql = "delete from produtos where
produtos.id_produto = $id_produto";
mysql_query($sql) or die(mysql_error());
header("location:../php/produtos_admin.php");
break;
case 'salvar':
    $msg = '';
    #VALIDAÇÃO DOS CAMPOS DO FORMULARIO
    if($id_tipo == '00')
        $msg = '<li>Selecione um tipo de produto!';

    if(strlen($nome_produto) == 0)
        $msg .= '<li>Nome inválido!';

    if(strlen($descricao) == 0)
        $msg .= '<li>Descrição inválida!';

    if(strlen($valor) == 0 || !is_numeric($valor))
        $msg .= '<li>Valor inválido';

    if(strlen($msg) > 0){

        header("location:mensagem_admin.php?msg=$msg");
        exit;
    }else{
        if(isset($id_produto)){
            #SE O ID_PRODUTO VEIO JUNTO É SINAL
            QUE É UM EDICAO (UPDATE NO BANCO), CAOS CONTRARIO É
            INSERÇÃO

            #VERIFICA SE O DESTAQUE VEIO MARCADO
            E ATUALIZA NO BANCO DE DADOS TAMBEM ;)
            if(isset($destaque))
```

```

                                $sql_destaque = ",destaque =
'Sim'";
                                else
                                $sql_destaque = '';

                                $sql = "update produtos set
                                    id_tipo =
'S$id_tipo',
                                    nome =
'$nome_produto',
                                    descricao =
'$descricao',
                                    valor = '$valor'
                                $sql_destaque
                                where produtos.id_produto
= $id_produto";
                                mysql_query($sql) or
die(mysql_error());

                                #NO MOMENTO DA INSERÇÃO VERIFICA SE
VEIO FOTO

                                if(strlen($foto) > 0){
                                    #ACEITA SÓ SE FOR ARQUIVO JPG
OU GIF

                                    if($foto_type == 'image/pjpeg'
|| $foto_type == 'image/gif'){
                                        #PEGA OS TRES ULTIMOS
CARACTERES DO ARQUIVO PARA FORMAR A EXTENSÃO
                                        $extensao =
explode(".", $foto_name);

                                        $tam = sizeof($extensao);
                                        $extensao =
$extensao[$tam-1];

                                        $nome_arquivo = 'foto_'.
$id_produto . "." . $extensao;

                                        #COPIA O ARQUIVO ENVIADO
PARA O DIRETORIO ESPECIFICADO

                                        copy($foto, "../fotos/$nome_arquivo");

```

```
#ATUALIZA O REGISTRO COM
O NOME DA FOTO
$sql = "update produtos
set foto = '$nome_arquivo'
where
produtos.id_produto = $id_produto";
mysql_query($sql) or
die(mysql_error());
}
}
}else{
    $sql = "insert into produtos
(nome,valor,descricao,id_tipo) values
('$nome_produto','$valor','$descricao','$id_tipo)";
    mysql_query($sql) or
    die(mysql_error());

    #NO MOMENTO DA INSERÇÃO VERIFICA SE
    VEIO FOTO

    #CASO TENHA VINDO INICIA O PROCESSO
    DE NOMEAÇÃO DO ARQUIVO E COPIA PARA O DIRETORIO

    if(strlen($foto) > 0){
        #motangem do nome do arquivo
        atraves do id do produto recém inserido
        $sql = "select
produtos.id_produto from produtos
order by
produtos.id_produto desc
limit 0,1";
        $rs = mysql_query($sql) or
        die(mysql_error());
        $id_produto = mysql_result($rs,
0, 'id_produto');

        #ACEITA SÓ SE FOR ARQUIVO JPG
        OU GIF
        if($foto_type == 'image/pjpeg'
|| $foto_type == 'image/gif'){
            #PEGA OS TRES ULTIMOS
            CARACTERES DO ARQUIVO PARA FORMAR A EXTENSÃO
            $extensao =
explode(".", $foto_name);
```

```

$tam = sizeof($extensao);
$extensao =

$extensao[$tam-1];

$nome_arquivo = 'foto_'.
$id_produto . "." . $extensao;

#COPIA O ARQUIVO ENVIADO
PARA O DIRETORIO ESPECIFICADO

    copy($foto, "../fotos/$nome_arquivo");

#ATUALIZA O REGISTRO COM
O NOME DA FOTO

    $sql = "update produtos
    set foto = '$nome_arquivo'
    where
    produtos.id_produto = $id_produto";
    mysql_query($sql) or
    die(mysql_error());
    }
    }
    }

    header('location:../php/produtos_admin.php');
    }
    break;
}

$tpl->printToScreen();
?>
```

É importante ressaltar o processo de exclusão do registro do banco, em que não basta simplesmente eliminar o registro, e sim eliminar junto o arquivo de imagem que está publicado no diretório de fotos, evitando assim o acúmulo de arquivos em desuso no site o que acaba por ocupar espaço desnecessário no servidor de hospedagem.

15.5 Controle de pedidos

Todas as compras efetuadas na loja serão armazenadas na base de dados, assim teremos um controle de quando foram realizadas as compras, por quem foram feitas e qual o valor que custou cada pedido.

Observe que na estrutura de modelagem do banco foram criadas duas tabelas para o controle de pedidos: PEDIDOS e PRODUTOS_PEDIDOS, na primeira teremos apenas um resumo do pedido, e na segunda os itens incluídos em cada compra. Teremos de realizar uma leitura simultânea das duas tabelas para montarmos nosso relatório.

Para iniciarmos a implementação deste módulo iremos agora criar o arquivo de template para o script, crie um arquivo novo HTML e salve-o no diretório htm do servidor com o nome de **pedidos_admin.htm**.

O layout deste arquivo deverá ficar semelhante ao mostrado abaixo:

PEDIDOS				
Nome do Cliente	Produtos do Pedido		Total do Pedido	Data
{nome_cliente}	{nome_produto}	R\$ {valor}	R\$ {total}	{dt_pedido}

Neste arquivo em particular teremos de montar dois blocos encadeados, sendo o primeiro chamado de BLOCO PEDIDOS e o segundo (aninhado como filho do primeiro) de BLOCO PRODUTOS.

No documento HTML a estrutura ficará como mostrada a seguir:

```
<!-- START BLOCK : PEDIDOS -->
.
.
.
    <!-- START BLOCK : PRODUTOS -->
    .
    .
    .
<!-- END BLOCK : PRODUTOS -->
.
.
.
<!-- END BLOCK : PEDIDOS -->
```

Iremos percorrer a tabela de PEDIDOS e a cada pedido encontrado iremos percorrer a tabela de PRODUTOS_PEDIDOS, gerando um relatório como mostrado abaixo:

PEDIDOS

Nome do Cliente	Produtos do Pedido		Total do Pedido	Data
João da Silva	Impressora HP Deskjet	R\$ 399.00	R\$ 4788.00	10/04/2007
João da Silva	Impressora HP Deskjet	R\$ 399.00	R\$ 399.00	15/03/2007
Paulo Silva	Memória DIMM	R\$ 59.99	R\$ 599.90	09/03/2007
João da Silva	Disco Rígido	R\$ 560.00		
	Impressora HP Deskjet	R\$ 123.00	R\$ 650.00	17/01/2007
	Mouse Óptico	R\$ 65.00		

Para interpretar o layout recém montado, iremos criar um novo arquivo php chamado **pedidos_admin.php** e salve-lo no diretório php do nosso site.

Este script conterà todas as queries de consulta para montar o relatório acima, levando em conta que deveremos chamar os métodos da classe útil para convertermos as datas e os valores para o formato compreensível e formatado.

Veja abaixo a sintaxe lógica do script:

```
<?php
include('../includes/classTemplatePower.php');
include('../includes/conexao.php');
include('../includes/classUtil.php');
include('../php/ver_sessao.php');

$util = new Util;

$tpl = new TemplatePower('../htm/modelo_admin.htm');
$tpl->assignInclude('CONTEUDO','../htm/pedidos_admin.htm');
$tpl->prepare();

#===== MONTA OS LINKS DO MENU SUPERIOR =====
$tpl->assign('link_tipos','../php/tipos_admin.php');
$tpl->assign('link_produtos','../php/produtos_admin.php');
$tpl->assign('link_pedidos','../php/pedidos_admin.php');
$tpl->assign('link_logout','../php/logout.php');

#===== SELECIONA OS PEDIDOS REALIZADOS E VINCULA AOS
RESPECITOVs CLIENTES E PRODUTOS
$sql = "select pedidos.*, clientes.* from pedidos,
clientes
```

```
        where pedidos.id_cliente =
clientes.id_cliente
        order by pedidos.dt_pedido desc";
$rs = mysql_query($sql) or die(mysql_error());
$total = mysql_num_rows($rs);

for($x = 0; $x < $total; $x++){
    $tpl->newBlock('PEDIDOS');
    $tpl->assign('nome_cliente',mysql_result($rs, $x,
'clientes.nome'));
    $tpl->assign('total',$util-
>formataDecimal(mysql_result($rs, $x,
'pedidos.total_pedido')));
    $tpl->assign('dt_pedido',$util-
>converteData(mysql_result($rs, $x,
'pedidos.dt_pedido')));

    #DENTRO DESSE MESMO BLOCO RECUPERA OS PRODUTOS DO
PEDIDO
    $sql2 = "select produtos_pedidos.*, produtos.* from
produtos_pedidos,produtos
        where produtos_pedidos.id_produto =
produtos.id_produto
        and produtos_pedidos.id_pedido = " .
mysql_result($rs, $x, 'id_pedido') . " order by
        produtos.nome";
    $rs2 = mysql_query($sql2) or die(mysql_error());
    $total2 = mysql_num_rows($rs2);

    for($y = 0; $y < $total2; $y++){
        $tpl->newBlock('PRODUTOS');
        $tpl->assign('nome_produto',mysql_result($rs2,
$y, 'nome'));
        $tpl->assign('valor',$util-
>formataDecimal(mysql_result($rs2, $y, 'valor')));
    }
}
$tpl->gotoBlock('_ROOT');

$tpl->printToScreen();
?>
```

Como mencionado ao entrarmos o bloco de pedidos automaticamente a cada volta do loop entramos no bloco de produtos, gerando assim o relatório de pedidos.

Unidade 16

Efetuando logout, finalizando a sessão

Após o usuário do ambiente administrativo do site ter efetuado as consultas e manutenções necessárias, precisamos finalizar de maneira segura a sua sessão, para isso criaremos um simples script de logout.

Neste arquivo a sessão é destruída, e o usuário é direcionado para a tela de login novamente.

Crie um novo arquivo php e salve-o como **logout.php** dentro do diretório php do site.

A estrutura do algoritmo deverá ficar como mostrado a seguir:

```
<?php
session_start();
session_destroy();
header('location:../php/modelo_site.php');
?>
```

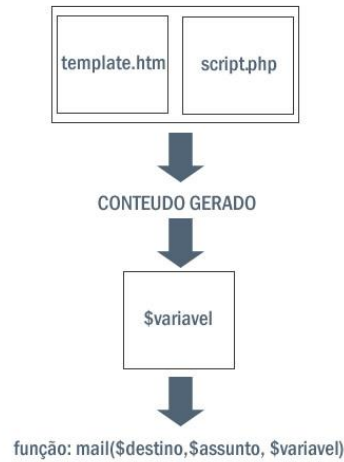
Com os passos acima seguidos podemos ter uma solução flexível e segura para uma loja virtual, lembrando que com a estrutura que montamos desde o banco de dados até os templates e scripts fica fácil agregar novas funcionalidades aos módulos utilizando-se de um tecnologia gratuita e altamente robusta volta para o ambiente web.

16.1 Extra - Implementando uma newsletter

Com a utilização da internet como uma nova forma de negócios e propaganda, a divulgação virtual se torna algo essencial para termos de forma rápida e automatizada o envio de mailing para um determinado grupo de usuários.

Utilizando a tecnologia de template podemos fazer isso de forma mais fácil e rápida do que dos meios habituais. Todo código lido e gerado pela junção do HTML com o script php é retornado para o usuário através do método `$tpl->printToScreen`, entretanto se não quiséssemos mostrar o conteúdo na tela e sim armazená-lo em uma variável poderíamos chamar o método `$tpl->getOutputContent`.

Veja o digrama abaixo, o qual explica a lógica do envio do documento por email:



Veja abaixo um exemplo prático de um script php que recebe um template chamado `newsletter.htm` e o envia por email:

Arquivo **newsletter.htm**:



Arquivo **newsletter.php**:

```
<?php
$headers = "From: Loja Virtual
<newsletter@lojavirtual.com.br>\n";
$headers .= "Content-Type: text/html; charset=iso-8859-1\n";
$headers .= "MIME-Version: 1.0\r\n";
//=====
//==> Instanciar TemplatePower para juntar o PHP + HTML
//=====
=
$tpl = new TemplatePower('../htm/newsletter.htm');
```

```
$tpl->prepare();

$tpl->assign('data',date("d/m/Y"));
$tpl->assign('titulo','Newsletter E-Commerce');
$tpl->assign('descricao','Esta é uma newsletter
exemplo!');

$message = $tpl->getOutputContent();

mail($destino, $titulo, $message, $headers);
?>
```

Analisando o script acima vemos que o comportamento é o mesmo, entretanto nada na tela é retornado e sim somente enviado por email.

É importante lembrar que no arquivo HTML, as imagens e o arquivo CSS deverão ser inseridos com o caminho completo: [HTTP://www.seudominio.com.br](http://www.seudominio.com.br) pois no cliente de email do usuário o navegador precisará do endereço completo de exibição das imagens.

Você pode incrementar o script fazendo com que os receptores do endereço sejam coletados do banco de dados, juntamente das informações que serão agregadas na newsletter. Uma sugestão interessante seria criar um módulo que permitisse a inclusão de produtos para anuncio na newsletter, fazendo assim que a divulgação de determinados produtos em promoção ou em lançamento na sua loja fosse enviado por email.

16.2 Formas de pagamento (boleto bancário e cartão de crédito)

Uma das duvidas cruciais no desenvolvimento de um sistema de e-commerce é a forma de pagamento da compra. Antigamente era necessário que o desenvolvedor implementasse uma serie de scripts para gerar códigos de barras e codificações específicas para a transação via cartão de crédito, atualmente existem dezenas de serviços que simplesmente solicitam o envio dos dados da compra como valor, nome do cliente e um código de identificação para determinada URL fornecida pela empresa.

Antes de começar a desenvolver uma aplicação que forneça esses dados ou que gere um boleto bancário automaticamente procure se informar dos serviços disponibilizados pelas empresas credenciadas e bancos autorizados para o fornecimento deste serviço. Assim você estará evitando retrabalho e futuros problemas na implementação do seu script de venda.