Walace Soares

BRINDE
Alguns códigos-fonte e exemplos do livro disponíveis em www.editoraerica.com.br para download.

- Framework de desenvolvimento
- Conceituação do sistema
- Módulo de cadastro
- Módulo de movimentação de estoques
- Relatórios operacionais e gerenciais
- Inventário
- Desenvolvimento com orientação a objetos

Crie um Sistema Web com PHP 5 e AJAX

CONTROLE DE ESTOQUE

Crie um Sistema Web com PHP 5 e AJAX

Controle de Estoque

EDITORA AFILIADA

Seja Nosso Parceiro no Combate à Cópia Ilegal

A cópia ilegal é crime. Ao efetuá-la, o infrator estará cometendo um grave erro, que é inibir a produção de obras literárias, prejudicando profissionais que serão atingidos pelo crime praticado.

Junte-se a nós nesta corrente contra a pirataria. Diga não à cópia ilegal.

Seu Cadastro É Muito Importante para Nós

Se você não comprou o livro pela Internet, ao preencher a ficha de cadastro em nosso site, você passará a receber informações sobre nossos lançamentos em sua área de preferência.

Conhecendo melhor nossos leitores e suas preferências, vamos produzir títulos que atendam suas necessidades.

Obrigado pela sua escolha.

Fale Conosco!

Eventuais problemas referentes ao conteúdo deste livro serão encaminhados ao(s) respectivo(s) autor(es) para esclarecimento, excetuando-se as dúvidas que dizem respeito a pacotes de softwares, as quais sugerimos que sejam encaminhadas aos distribuidores e revendedores desses produtos, que estão habilitados a prestar todos os esclarecimentos.

Os problemas só podem ser enviados por:

1. E-mail: producao@erica.com.br
2. Fax: (11) 2097.4060
3. Carta: Rua São Gil, 159 - Tatuapé - CEP 03401-030 - São Paulo - SP

Walace Soares

Crie um Sistema Web com PHP 5 e AJAX

Controle de Estoque

1ª Edição

São Paulo
2009 - Editora Érica Ltda.

Dados Internacionais de Catalogação na Publicação (CIP)
(Câmara Brasileira do Livro, SP, Brasil)

Soares, Walace
Crie um sistema web com PHP 5 e AJAX: Controle de estoque / Walace Soares. -- 1. ed. -- São Paulo: Érica, 2009.

Bibliografia
ISBN: 978-85-365-0240-3

1. AJAX (Tecnologia de desenvolvimento de sites da web) 2. Aplicativos - Software - Desenvolvimento 3. Framework (Programa de computador) 4. PHP (Linguagem de programação para computadores) 5. Web sites - Desenvolvimento I. Título.

09-05238 CDD-005.12

Índices para catálogo sistemático
1. Framework para sistemas web com PHP 5 e AJAX: Ciências da computação 005.12

Conselho Editorial:

Revisão e Coordenação Editorial:	Rosana Arruda da Silva
Editoração:	Rosana Ap. A. dos Santos
Finalização de Capa:	Maurício S. de França
Revisão Gramatical:	Marlene Teresa S. Alves
	Carla de Oliveira Morais
Figuras:	Flávio Eugenio de Lima

Editora Érica Ltda.
Rua São Gil, 159 - Tatuapé
CEP: 03401-030 - São Paulo - SP
Fone: (11) 2295-3066 - Fax: (11) 2097-4060
www.editoraerica.com.br

Fabricantes

Produto: **PHP**
Fabricante: PHP Group
Site: www.php.net

Produto: **MySQL**
Fabricante: MySQL AB
Site: www.mysql.com

Produto: **PostgreSQL**
Fabricante: PostgreSQL Inc.
Site: www.postgresql.org

Produto: **Apache**
Fabricante: Apache Software Foundation
Site: www.apache.org

Requisitos de Hardware e de Software

Software

- Linux (Kernel 2.4 ou superior) ou Windows (2000/XP/Vista)
- Servidor Web Apache 2.0.63 ou superior
- MySQL 5.1 ou superior
- PostgreSQL 8.3 ou superior
- PHP 5.2.9 ou versão mais recente
- Browser Mozilla Firefox 3.10 ou superior, Internet Explorer 7 ou superior

Hardware

- Intel Pentium ou compatível, 1 GHz ou superior
- 256 MB de RAM (512 recomendado)
- CD-ROM
- 20 GB de espaço disponível em disco
- Modem e acesso à Internet

Conhecimento necessário

- Lógica de programação Orientada a Objetos (OOP)
- Qualquer banco de dados (de preferência postgreSQL ou MySQL)
- PHP 5 (classes em especial) e javascript
- AJAX
- Html e CSS

Dedicatória

Aos meus pais,
Darci e Elza.

"Amarás o Senhor teu Deus, com todo teu coração,
com toda tua alma e com toda tua mente."

Mateus - 22, 37

Agradecimentos

A todos que me ajudaram direta e indiretamente no desenvolvimento deste livro, em especial à minha família pelo apoio e compreensão.

Índice analítico

Prefácio

O desenvolvimento de sistemas é sempre um desafio, muitas vezes gratificante. Melhor ainda se tivermos uma boa base para a sua construção, pois certamente teremos mais chances de sucesso na empreitada. Quando digo boa base, refiro-me não só ao conhecimento técnico, mas principalmente ao conhecimento do negócio, das operações que devem ser realizadas, das interações, do fluxo dos processos, enfim, de tudo que é necessário para representar o sistema do mundo físico no mundo virtual.

A conversão dessa representação em um sistema computacional é ainda mais fácil se construirmos o sistema em um framework de aplicação, uma vez que a maioria das tarefas, principalmente as mais monótonas, já está pronta, restando a parte nobre do processo, ou seja, a implementação das regras de negócio.

O enfoque deste livro é justamente mostrar como construir um sistema de controle de estoque baseando-se em regras de negócio e em um framework de aplicação, que neste caso é o fw.PHP, um framework robusto e totalmente voltado ao desenvolvimento de sistemas para o ambiente web.

Começa com a definição do framework e um tutorial sobre sua utilização. Além disso implementa ferramentas adicionais para melhor aproveitamento da estrutura oferecida pelo framework, tema tratado no capítulo 1 do livro.

O capítulo 2 discute o sistema de controle de estoque, descreve as regras de negócio, a estrutura do banco de dados e tudo que é necessário para a sua implementação.

No capítulo 3 começamos a construir a base do sistema, criando os cadastros auxiliares descritos no capítulo anterior. Essa construção termina no capítulo 4, que trata do principal cadastro do sistema, o de produtos.

O capítulo 5 aborda a movimentação do estoque, tema principal do sistema, estrutura o movimento de estoque, criando a classe principal, e ainda desenvolve a classe para entrada de estoque, passo inicial de um sistema desse tipo.

Já o capítulo 6 mostra como desenvolver a classe de saída de estoque e o capítulo 7 refere-se à contraparte dos lançamentos de entrada e saída de estoque, isto é, o estorno de movimentação.

No capítulo 8 estão os principais relatórios de um sistema de controle de estoque, tais como saldo atual, estoque acima do máximo, estoque abaixo do mínimo, extrato de movimentação, razão de estoque e outros relatórios relacionados ao tema.

Encerra com a discussão sobre o módulo de inventário de estoque, no capítulo 9. Mostra como desenvolver o gerenciamento do inventário, desde a criação do inventário, passando pela seleção dos produtos, congelamento dos estoques, controle das contagens, e fecha com a apuração do inventário e o ajuste dos estoques dos produtos.

Ao final da leitura você terá em mãos um sistema de gerenciamento de estoque de produtos robusto, voltado para a plataforma web e extremamente simples de implementar, manter e expandir.

Boa leitura!

O autor

Sobre o autor

Walace Soares é capixaba, natural de Vila Velha, formado em Matemática com ênfase em Computação. Trabalha desde 1987 na área de informática e passou pelos mais diversos segmentos (indústria, comércio, serviços).

Atualmente é sócio-proprietário da MWS Serviços e Sistemas, onde se dedica ao desenvolvimento de softwares de alta tecnologia utilizando PHP, JavaScript, AJAX, Flex e outras ferramentas. Seu primeiro contato com PHP foi em 1999 (ainda na versão 3), e desde essa época vem se aplicando a conhecer e utilizar de forma cada vez mais profissional essa poderosa linguagem.

Sobre o material disponível na Internet

O material disponível na Internet contém o framework **fw.PHP** com todas os códigos-fontes e os complementos introduzidos no livro, parte do código do sistema descrito no livro e o script para criação do banco de dados no postgreSQL. É necessário ter instalados na máquina os programas Apache 2.0.63 ou superior, MySQL 5 ou PostgreSQL 8.3 ou versão superior, PHP 5.2.9 ou superior.

Sistema_Web.EXE - 1.14 MB

Procedimento para Transferência

Acesse o site da Editora Érica Ltda.: www.editoraerica.com.br. A transferência do arquivo disponível pode ser feita de duas formas:

- **Por meio do módulo pesquisa.** Localize o livro desejado, digitando palavras-chave (nome do livro ou do autor). Aparecerão os dados do livro e o arquivo para download, então dê um clique no arquivo executável que será transferido.
- **Por meio do botão Download.** Na página principal do site, clique no item **Download**. É exibido um campo no qual devem ser digitadas palavras-chave (nome do livro ou do autor). Serão exibidos o nome do livro e o arquivo para download. Dê um clique no arquivo executável que será transferido.

Procedimento para Descompactação

Primeiro passo: após ter transferido o arquivo, verifique o diretório em que se encontra e dê um duplo-clique nele. Será exibida uma tela do programa WINZIP SELF-EXTRACTOR que conduzirá você ao processo de descompactação. Abaixo do Unzip To Folder, existe um campo que indica o destino do arquivo que será copiado para o disco rígido do seu computador.

C:\Crie um Sistema Web

Segundo passo: prossiga a instalação, clicando no botão Unzip, o qual se encarrega de descompactar o arquivo. Logo abaixo dessa tela, aparece a barra de status a qual monitora o processo para que você acompanhe. Após o término, outra tela de informação surge, indicando que o arquivo foi descompactado com sucesso e está no diretório criado. Para sair dessa tela, clique no botão OK. Para finalizar o programa WINZIP SELF-EXTRACTOR, clique no botão Close.

Veja como fica o projeto que você vai desenvolver com o apoio deste livro.
Entre no site: www.editoraerica.com.br/sistema_web
Usuário: estoque
Senha: estoque

Capítulo 1

Introdução

Este livro desenvolve um sistema web para controle de estoque. A construção do sistema baseia-se no framework **fw.PHP**, que é dirigido ao desenvolvimento de sistema na web. Ele contém a infra-estrutura necessária para o desenvolvimento desse tipo de sistema, incluindo classes para o gerenciamento de acesso a banco de dados, estruturação de formulários, controle de acesso e gerência de permissões.

O sistema desenvolvido no livro contém as rotinas fundamentais para a gestão de estoque, proporcionando uma base para a expansão do sistema e adequação a necessidades específicas. Tratamos dos cadastros básicos, rotinas de entrada e saída de estoque, relatórios básicos e o gerenciamento de inventário.

Um sistema de controle de estoques é simples, porém de fundamental importância na gestão de uma empresa, pois é através dele que os tomadores de decisão sabem se é necessário aumentar ou diminuir as compras, quais produtos têm um giro maior, quais produtos estão parados há muito tempo e outras informações que ajudam na tomada de decisão.

Este primeiro capítulo mostra o básico da utilização do framework **fw.PHP** e do capítulo 2 em diante desenvolvemos as rotinas necessárias para o sistema em questão.

O que você precisa saber

Para que você acompanhe de forma satisfatória o desenvolvimento do sistema proposto pressupõe-se que tenha conhecimento razoável de:

- Lógica de programação;
- Orientação a Objeto (OO);
- Qualquer banco de dados (de preferência postgreSQL ou MySQL);
- Comandos SQL (interação com banco de dados);
- PHP 5 (especialmente classes);
- Javascript (preferencialmente com OO);

- AJAX;
- Html;
- CSS.

Framework fw.PHP

Totalmente desenvolvido em PHP, esse framework tem suporte às principais necessidades de um sistema direcionado à web. A lista a seguir mostra seus recursos principais.

- Gerenciamento de vários bancos de dados
- Classe base para gestão de entidades do sistema
- Geração automática de lista de registros
- Filtro de registros
- Geração de relatórios
- Geração automática de formulários de manutenção
- Suporte a AJAX
- Controle de usuários
- Rotinas para autenticação de usuários
- Gerenciamento de menus estruturados e dinâmicos
- Controle de permissões de acesso por usuário e menu
- Auditoria de processos
- Histórico de uso por usuário
- Controle de Favoritos

Como instalar o framework

Antes de começar a utilizar o framework, você deve executar os seguintes passos:

1. Baixar o framework do site http://www.editoraerica.com.br, da Editora Érica.
2. Instalar o framework no diretório de trabalho do servidor web (apache).
3. Criar o banco de dados inicial.
4. Incluir manualmente no banco de dados o usuário de administração do sistema.

5. Incluir a estrutura de menus do framework.
6. Configurar o framework.
7. Testar o framework.

Baixar o framework

Para baixar os arquivos do framework, basta acessar o site da Editora Érica (http://www.editoraerica.com.br) e, na página relacionada ao livro, executar a opção de download dos arquivos relacionados.

Instalar o framework

O arquivo baixado no site da Editora deve ser descompactado no diretório de trabalho do servidor web (preferencialmente Apache versão 2.0.63 ou maior).

Para não misturar o framework com outras possíveis aplicações, crie um diretório específico para conter o framework e os sistemas desenvolvidos com ele. Como padrão, vamos chamar esse diretório de **sistemas_web**. Após criar o diretório e instalar o framework, você deve ver a estrutura apresentada ao lado.

Com essa estrutura vamos desenvolver o sistema proposto no livro.

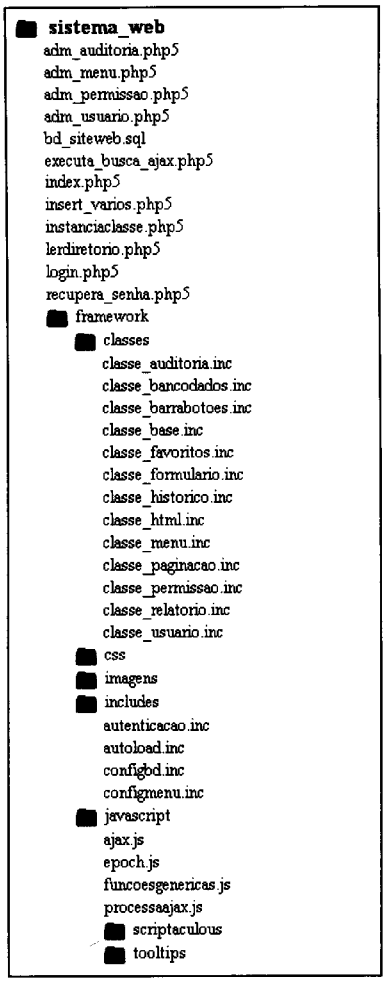


Figura 1.1

Criar o banco de dados

O próximo passo é criar o banco de dados para suportar o desenvolvimento. O framework suporta tanto mySQL quanto postgreSQL (se necessário, é possível incluir no framework o suporte a outros bancos de dados; basta um pouco de programação). Definimos o banco de dados com o nome de **sistemaweb**. Após criar o banco de dados vamos executar o script de inicialização contido no framework (o script disponível é compatível com postgreSQL, mas com algumas alterações pode ser utilizado no mySQL também). Basta executar o administrador do banco de dados, o pgAdmin, por exemplo, carregar e executar o script *bd_siteweb.sql*. Após a execução desse script você deve ver as seguintes tabelas, criadas no banco de dados:

- auditoria
- favoritos
- histórico
- menu
- permissao
- usuario

Nota: Você verá, ainda, a tabela tab_teste que serve apenas para testes no framework.

Incluir usuário administrador

Para que o framework funcione, é necessário que exista pelo menos um usuário cadastrado no banco de dados. Você precisa incluir esse usuário manualmente antes de começar a usar o sistema. O comando seguinte mostra como criar um usuário administrador com a senha predefinida.

```
INSERT INTO usuario(usuario_id, usuario_nome, usuario_login,
                    usuario_senha, usuario_email, usuario_ativo)
VALUES (1, 'Administrador', 'admin', 'SfCxXyng3gs=', '', 1);
```

Ao executar esse script no banco de dados, é criado um usuário **admin** com a senha **admin** (a senha está criptografada).

Estrutura de menus do framework

O framework necessita de uma estrutura mínima de menus para funcionar corretamente. A estrutura deve conter pelo menos o programa que gera os menus do sistema. Você pode executar essa tarefa de duas formas, a primeira é rodar o script completo mostrado aqui, a segunda forma é criar apenas a estrutura inicial que contém o menu *Administração* e o item *Gerenciador de menus*, em seguida incluir manualmente cada um dos menus do sistema.

```
INSERT INTO menu (menu_id, menu_desc, menu_prog, menu_icon, menu_pai)
    VALUES (2, 'Administração', '[menu]', '', -1);
INSERT INTO menu (menu_id, menu_desc, menu_prog, menu_icon, menu_pai)
    VALUES (3, 'Gerenciador de Menus', 'adm_menu.php5', 'menu.png', 2);
INSERT INTO menu (menu_id, menu_desc, menu_prog, menu_icon, menu_pai)
    VALUES (11, 'Gerenciador de Usuários', 'adm_usuario.php5', 'user.png', 2);
INSERT INTO menu (menu_id, menu_desc, menu_prog, menu_icon, menu_pai)
    VALUES (12, 'Permissões', 'adm_permissao.php5', 'permissoes.png', 2);
INSERT INTO menu (menu_id, menu_desc, menu_prog, menu_icon, menu_pai)
    VALUES (14, 'Auditoria', 'adm_auditoria.php5', 'audit.png', 2);
```

Configurar

O último passo antes de testar o framework é configurar o acesso ao banco de dados. A configuração deve ser realizada no arquivo *configbd.inc* que está dentro do diretório *framework/includes*. No arquivo precisamos alterar os parâmetros de conexão com o banco de dados e em geral deve-se alterar apenas o nome do banco de dados, o usuário de acesso e a senha. Isso é feito alterando as variáveis $_banco, $_usuario e $_senha. No livro usamos a seguinte configuração:

- **Banco de dados:** sistemaweb
- **Usuário:** postgres
- **Senha:** postgres

O arquivo deve ficar da seguinte forma:

Lista 1: configbd.inc

```
<?php
/**
 * Configurador para acesso ao Banco de Dados
 * Altere os valores de $_servidor, $_banco, $_usuario, $_senha e $_porta
 *
 */

/**
 * @var string define o tipo do banco de dados, padrão pgsql (opção atual mysql)
 */
$_tipobd = 'pgsql';
/**
 * @var string define o endereço do servidor do banco de dados
 */
$_servidor = "localhost";
/**
 * @var integer Porta de conexão
 */
$_porta = 5432;
/**
 * @var string define o nome do banco de dados que desejamos conectar
 */
$_banco = "sitemaweb";
/**
 * @var string informa o nome do usuário
 */
$_usuario = "postgres";
```

```
/**
 * @var string Senha do usuário
 */
$_senha = "postgres";

/**
 * Conecta com o Banco de dados e retorna uma instância de bancodados
 */
$_bd = new $_tipobd();
$_bd->SetServidor($_servidor);
$_bd->SetPorta($_porta);
$_bd->SetBanco($_banco);
$_bd->SetUsuario($_usuario);
$_bd->SetSenha($_senha);
if($_bd->Conectar()===false) {
    die("<p style='color:red;font-weight:bold;text-align:center;'>Erro na conexão
com o Banco de Dados {$_bd->getUltimoErro()}</p>");
}
?>
```

Teste de funcionamento do framework

O acesso ao framework deve ser feito pela chamada ao programa *login.php5* que está no diretório principal do sistema, ou seja, em *localhost/sistema_web* (troque localhost pelo endereço da máquina na qual está rodando o servidor web). Ao executar esse programa, devemos ver a tela exibida na Figura 1.2.

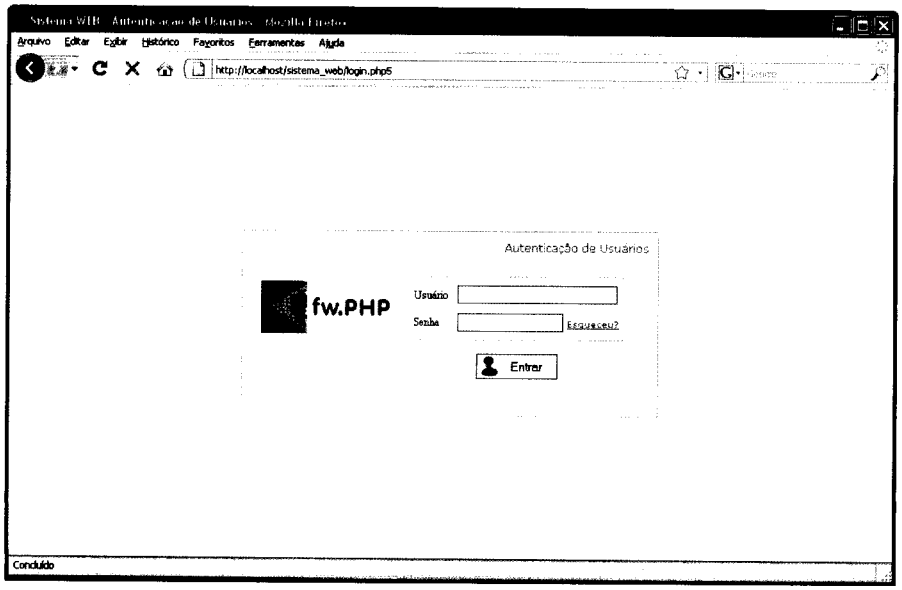


Figura 1.2

Se tudo estiver correto, após informar o usuário e a senha (admin/admin), teremos a tela da Figura 1.3.

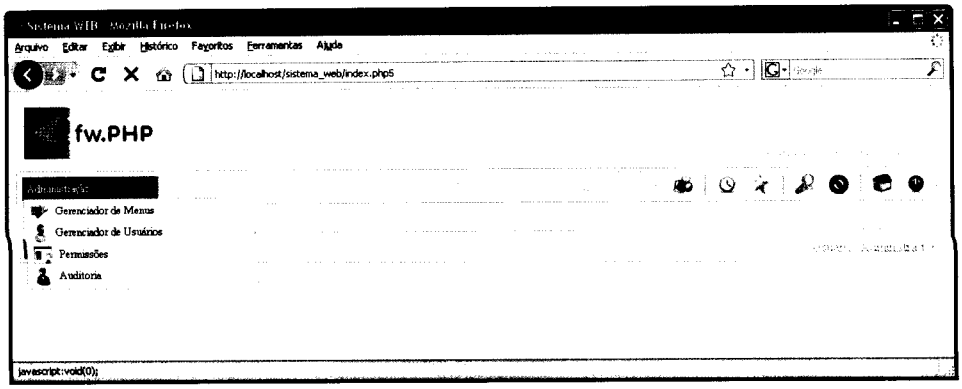


Figura 1.3

Para finalizar o teste do framework, vamos entrar no menu *Gerenciador de Menus* e incluir o menu para o sistema de estoques.

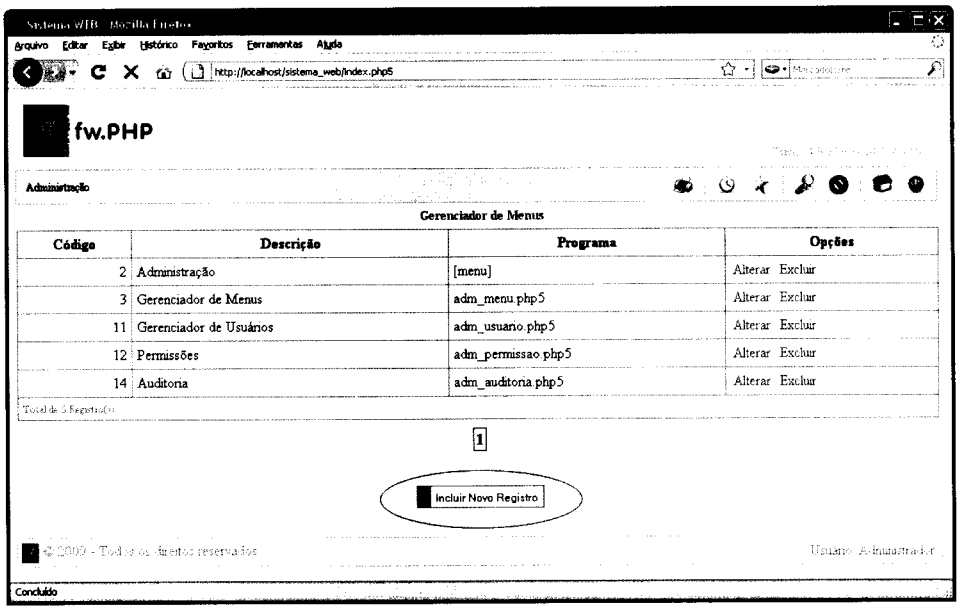


Figura 1.4

Para incluir o item do menu, basta clicar no botão *Incluir Novo registro* e digitar os dados do menu:

- **Descrição:** Estoque
- **Programa:** [menu]
- **Imagem Relacionada:** Nenhuma
- **Menu Relacionado(pai):** Nenhum

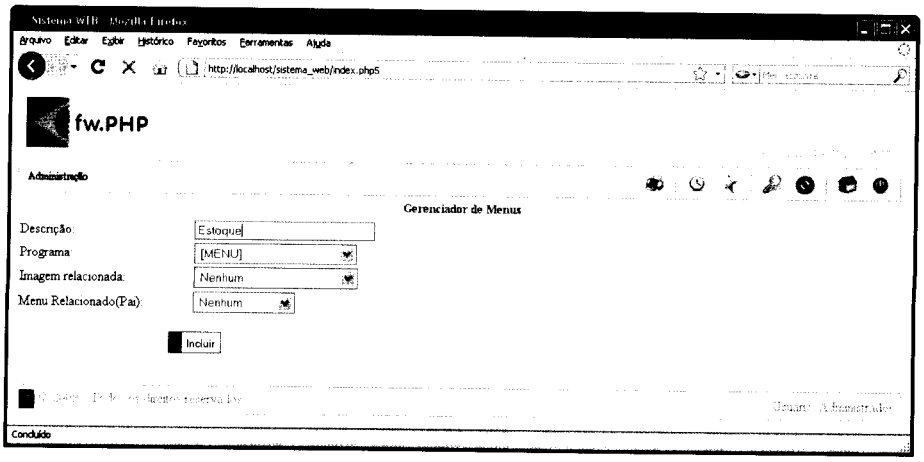


Figura 1.5

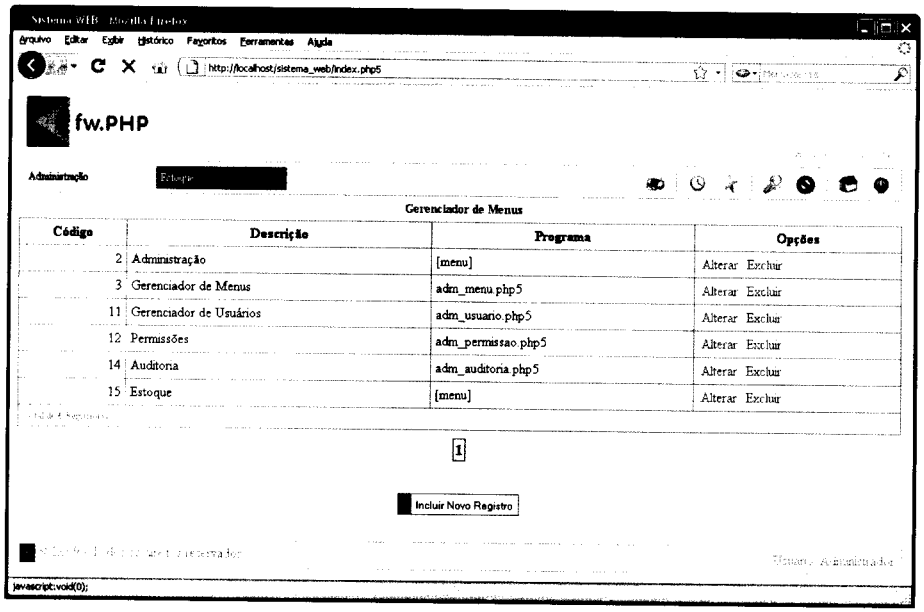


Figura 1.6

Para finalizar é só clicar no botão *Incluir*; com isso o sistema insere o novo registro no banco de dados. Porém, somente visualiza o novo item de menu quando fizer uma nova autenticação no sistema. Para isso você deve clicar no botão de saída do sistema, que é o último da barra de botões, e informar novamente seus dados de usuário (admin/admin) na página de autenticação. Ao entrar novamente no sistema, a primeira diferença que nota é que a página inicial não está mais em branco, como ocorreu anteriormente. Em vez disso você verá a página inicial do gerenciador de menus. Esta é uma das características do framework, que grava a última página que o usuário acessou no sistema e a exibe sempre que é feita a entrada no sistema ou quando o usuário clica

no botão de página inicial ou no logo do framework. A segunda diferença é a presença do menu *Estoque* que será exibido ao lado do menu *Administração*.

Os testes iniciais estão concluídos e o framework está pronto para utilização. A seguir veremos o básico de sua utilização.

Utilização do framework fw.PHP

Ele obedece ao conceito de framework de aplicação e também de framework híbrido. O primeiro conceito define que o framework está voltado a uma tarefa específica, ou seja, não é generalista, o que é verdade, pois está focado no desenvolvimento de sistemas web, tais como ERP, CRM e BI, entre outros. Já o segundo conceito define que esse framework é baseado em heranças de classes, mas também disponibiliza algumas ferramentas para uso geral no desenvolvimento dos sistemas.

É fácil usar o framework. Apenas é preciso entender a sua mecânica. Todos os processos disponibilizados estão definidos em uma série de classes dentro do framework. Para utilizá-los deve-se primeiro estender a classe básica do framework que é a **base**. Tudo se baseia em converter as entidades e seus atributos em classes estendidas da classe **base**, cujos atributos das entidades devem ser convertidos em instâncias de classes específicas para cada tipo de atributo.

Veja, por exemplo, a tabela de gerenciamento de menus, que possui os campos MENU_ID, MENU_DESC, MENU_PROG, MENU_ICON e MENU_PAI. No framework a classe que representa essa entidade é a **menu** e está definida da seguinte forma (mostra apenas uma parte do construtor da classe):

```
/**
 * Classe para gerenciamento de menus do sistema
 *
 */
class menu extends BASE {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'menu';
          $this->addCampo(new inteiro("MENU_ID","Código",10,
                    null,null,false,false,true,null,null,null,true));
          $this->addCampo(new string("MENU_DESC","Descrição",30,
                    null,null,true,true,true,5,null,null,false,$_conn));
          $this->addCampo(new string("MENU_PROG","Programa",120,
                    null,null,true,true,true,null,null,null,false,$_conn));
          $this->addCampo(new string("MENU_ICON","Imagem relacionada",120,
                    null,null,true,true,false,null,null,null,false,$_conn));
          $this->addCampo(new inteiro("MENU_PAI","Menu Relacionado(Pai)",10,
                    null,null,true,true,false,null,null,null,false,$_conn));
     }
}
```

A classe contém vários outros métodos, mas neste momento o que interessa é o construtor da classe, o qual define os atributos da tabela como instância das classes correspondentes ao tipo de cada atributo, ou seja, se o atributo é um inteiro, utiliza-se a classe **inteiro** para defini-lo (como no caso do atributo MENU_ID), se o atributo é do tipo caracteres, usamos a classe **string** para defini-lo (como no caso de MENU_DESC) e assim sucessivamente.

Para entender o funcionamento do framework, vamos tomar como exemplo a tabela de testes *carro*, a qual contém os atributos marca, modelo, ano fabricação, ano modelo e cor. O script de criação da tabela no banco de dados é exibido em seguida:

```
CREATE TABLE carro (
  codigo integer NOT NULL,
  marca character varying(50),
  modelo character varying(100),
  ano_fabricacao integer,
  ano_modelo integer,
  cor character varying(20),
  CONSTRAINT carros_pkey PRIMARY KEY (codigo)
)
```

Note que incluímos o atributo *codigo* que será usado como chave primária da tabela.

Para inserir essa entidade no framework, precisamos, em primeiro lugar, especificar uma classe que conterá as definições necessárias. Como nome da classe utilizamos a palavra mais lógica que é o nome da entidade que estamos convertendo. Desta forma, a classe será nomeada como **carro** e terá inicialmente o seguinte formato:

Lista 2: classe_carro.inc

```
<?php
/**
 * Definição da classe carro
 * que reflete a entidade carro no banco de dados
 */
class carro extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'carro';
          $this->_class_path = 'teste';
          $this->addCampo(new inteiro("CODIGO","Código",10,
                    null,null,false,false,true,null,null,null,true));
          $this->addCampo(new string("MARCA","Marca",50,
                    null,null,true,true,true,2,null,null,false,$_conn));
          $this->addCampo(new string("MODELO","Modelo",100,
                    null,null,true,true,true,null,null,null,false,$_conn));
          $this->addCampo(new inteiro("ANO_FABRICACAO","Ano de fabricação",4,
                    null,null,true,true,true,null,null,null));
          $this->addCampo(new inteiro("ANO_MODELO","Ano do modelo",4,
                    null,null,true,true,true,null,null,null));
          $this->addCampo(new string("COR","Cor",20,
                    null,null,true,true,true,null,null,null,false,$_conn));  }
}
?>
```

Com esta definição já é possível gerenciar a entidade pelo framework. Salve o arquivo contendo a classe (classe_carro.inc) no diretório *teste/classes/*. No construtor definimos o nome interno da tabela através do atributo *$_nome_tabela* e também o diretório de trabalho no atributo *$_class_path*. Esses dois atributos são de definição obrigatória para a correta utilização do framework.

Agora precisamos criar o programa que será o ponto de conexão com o sistema de menus do framework. O programa é muito simples, contendo apenas algumas poucas linhas (três linhas na verdade). Tome por exemplo o programa para gerenciamento de menus.

```
<?php
/**
 * Gerenciador de menus
 *
 */
$_classe = Array("arquivo"=>"framework/classes/classe_menu.inc",
                 "nome"=>"menu");
$_FTitulo = "Gerenciador de Menus";
return include_once("instanciaclasse.php5");
?>
```

São apenas três linhas. A primeira define onde encontrar a classe e o nome da classe que desejamos instanciar. A segunda define o título do programa e a última linha carrega o programa instanciaclasse.php5, que é responsável por carregar o que é necessário para o processamento do programa.

A mesma lógica devemos seguir para criar o programa de gerenciamento da classe **carro**.

```
<?php
/**
 * Gerenciador de carros
 *
 */
$_classe = Array("arquivo"=>"teste/classes/classe_carro.inc",
                 "nome"=>"carro");
$_FTitulo = "Gerenciador de Carros";
return include_once("instanciaclasse.php5");
?>
```

Salve este programa no diretório *teste* com o nome *man_carros.php5*. Já temos a estrutura necessária para gerenciar a tabela de carros no sistema. Basta agora incluir o menu relacionado. Se você entrar em gerenciamento de menus, clicar no botão *Incluir novo registro* e abrir a lista de programas, verá que o programa *man_carros.php5* não aparece na lista, Figura 1.7. Por padrão, o framework verifica apenas o diretório raiz para gerar a lista. Precisamos alterar esse comportamento, incluindo o diretório *teste* na estrutura do framework. Isso é simples e rápido; basta editar o arquivo *configmenu.inc* que está no diretório *framework/includes*.

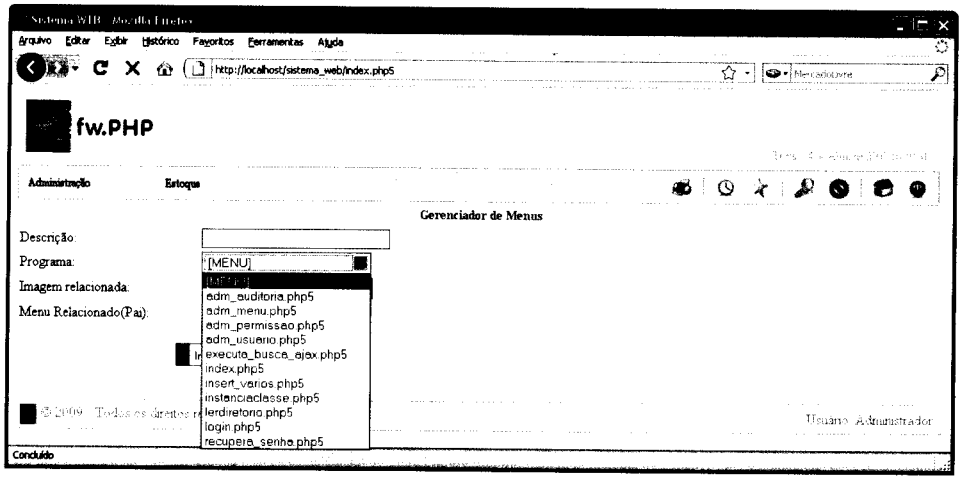


Figura 1.7

Lista 3: configmenu.inc

```
<?php
/**
 * Configurador de Sistemas existentes no framework
 * Não incluir o próprio Framework
 * O formato deve ser
 * Array('Sistema/Módulo'=>Diretório, ...)
 * Exemplo:
 *   Array(
 *             'Estoque/Cadastros'=>'estoque/cadastros/',
 *             'Estoque/Relatórios'=>'estoque/relatorios/',
 *             'Estoque/Processos'=>'estoque/processos/'
 *       );
 * A chave 'Sistema/Módulo' será exibida na inclusão de menus
 */
return    Array();
?>
```

O próprio arquivo já traz um exemplo de como deve ser definida a estrutura que será enviada ao framework. Alterando esse arquivo para conter o diretório de testes, teremos:

Lista 4: configmenu.inc (modificado)

```
<?php
/**
 * Configurador de Sistemas existentes no framework
 * Não incluir o próprio Framework
 * O formato deve ser
 * Array('Sistema/Módulo'=>Diretório, ...)
 * Exemplo:
 *   Array(
 *             'Estoque/Cadastros'=>'estoque/cadastros/',
 *             'Estoque/Relatórios'=>'estoque/relatórios/',
 *             'Estoque/Processos'=>'estoque/processos/'
 *       );
```

```
 * A chave 'Sistema/Módulo' será exibida na inclusão de menus
 */
return    Array('Teste'=>'teste/');
?>
```

Ao executar novamente a rotina de inclusão de menus, veremos que o programa *man_carros.php5* aparece na lista de programas.

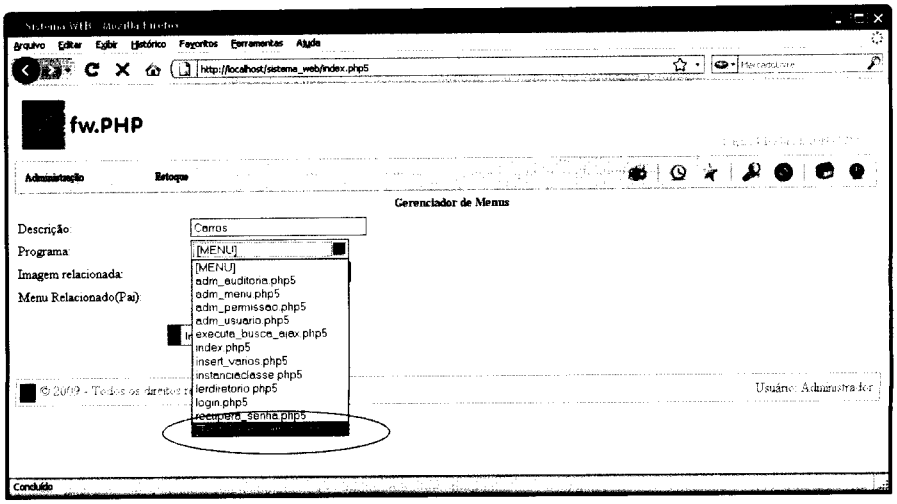


Figura 1.8

Inclua o item no menu Administração. Para visualizar o novo item, é necessário sair do sistema e entrar novamente.

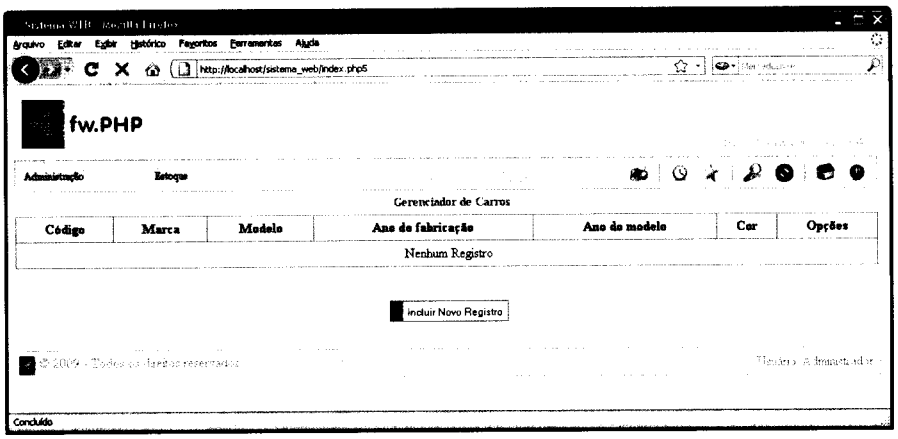


Figura 1.9

Utilização básica dos recursos do framework

Já temos a página inicial do gerenciador de carros, o que mostra que a classe está funcionando corretamente. Porém, ainda faltam alguns detalhes. Tente, por exemplo, incluir um novo carro na tabela. Clique no botão *Incluir Novo Registro*, digite os dados desejados e clique no botão *Incluir*. Ao processar o formulário e tentar incluir o novo registro, o framework retorna uma mensagem de erro, parecida com a da Figura 1.10.

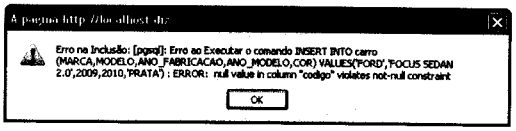


Figura 1.10

O motivo do erro é simples. No formulário de inclusão de um novo carro o atributo *codigo* não é mostrado, porque na definição desse campo dentro do framework, ele foi marcado para não ser exibido na inclusão de novos registros (na alteração também). Portanto, o campo não é incluído no comando SQL de inserção de um novo registro na tabela, o que gera o erro apresentado, uma vez que esse campo é a chave primária da tabela relacionada e como tal não pode conter o valor nulo (null).

Para solucionar este problema, devemos alterar a classe **carro** para que esse campo, antes da execução da operação de inclusão, seja inserido na lista de campos que participam dessa operação e, além disso, precisamos gerar um valor para o campo. Como esse campo é seqüencial, precisamos buscar o maior valor existente na tabela e atribuir ao campo *codigo* o valor encontrado acrescido de um. O framework possui mecanismos para resolver este problema. A classe precisa definir apenas o comando SQL necessário para a obtenção do maior valor existente. Basta sobrescrever o método *montaSQLProximoCodigo()* e o método *Incluir()* da classe **base** para que o campo associado tenha seu valor acertado. Além disso, é necessário alterar o marcador de inclusão do campo para verdadeiro (true) e então chamar o método original de inclusão.

Alterando a classe **carro**, teremos:

Lista 5: classe_carro.inc

```
<?php
/**
 * Definição da classe carro
 * que reflete a entidade carro no banco de dados
 */
class carro extends base {
    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'carro';
        $this->_class_path = 'teste';
        $this->addCampo(new inteiro("CODIGO","Código",10,
                    null,null,false,false,true,null,null,null,true));
        $this->addCampo(new string("MARCA","Marca",50,
```

```
                    null,null,true,true,true,2,null,null,false,$_conn));
        $this->addCampo(new string("MODELO","Modelo",100,
                    null,null,true,true,true,null,null,null,false,$_conn));
        $this->addCampo(new inteiro("ANO_FABRICACAO","Ano de fabricação",4,
                    null,null,true,true,true,null,null,null));
        $this->addCampo(new inteiro("ANO_MODELO","Ano do modelo",4,
                    null,null,true,true,true,null,null,null));
        $this->addCampo(new string("COR","Cor",20,
                    null,null,true,true,true,null,null,null,false,$_conn));
    }

    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(CODIGO) AS CODIGO FROM {$this->_nome_tabela}";
    }

    /**
     * Busca o próximo código para o menu
     *
     * @return mixed
     */
    public function incluir() {
        if(($_cod=$this->getProximoCodigo(true))=='[ERRO]') {
            return false;
        } else {
            $this->getCampo("CODIGO")->setValor($_cod);
            $this->getCampo("CODIGO")->setIncluir(true);
            return parent::incluir();
        }
    }
}
?>
```

Executando novamente a operação de inclusão, veremos que a operação é bem-sucedida.

Utilização dos recursos do framework para melhorar o controle da classe

O que fizemos até agora foi apenas utilizar a estrutura básica do framework, porém ele disponibiliza vários recursos avançados para melhorar a interface entre sistema e usuários.

Um exemplo é o estabelecimento de uma lista de opções para determinados campos da classe. Digamos que desejamos limitar o campo MARCA a uma lista com as principais marcas de carros disponíveis no Brasil.

Para incluir este comportamento na classe, precisamos em primeiro lugar alterar o formato de exibição do campo no formulário, que neste caso deve ser uma lista de seleção, o que equivale ao tipo SELECT no formulário web.

A alteração é feita pela alteração do atributo comportamento_form, o que é realizado com o método virtual *setComportamento_form()*. Depois precisamos definir a lista de valores que será exibida no campo relacionado, o que é feito pelo método virtual *setValor_fixo()*. A lista de valores deve estar no formato padrão do framework que é Array(Array('label'=>titulo,'valor'=>valor),...). Alterando a classe **carro**, teremos:

Lista 6: classe_carro.inc

```
<?php
/**
 * Definição da classe carro
 * que reflete a entidade carro no banco de dados
 */
class carro extends base {
    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'carro';
        $this->_class_path = 'teste';
        $this->addCampo(new inteiro("CODIGO","Código",10,
                    null,null,false,false,true,null,null,null,true));
        $this->addCampo(new string("MARCA","Marca",50,
                    null,null,true,true,true,2,null,null,false,$_conn));
        $this->addCampo(new string("MODELO","Modelo",100,
                    null,null,true,true,true,null,null,null,false,$_conn));
        $this->addCampo(new inteiro("ANO_FABRICACAO","Ano de fabricação",4,
                    null,null,true,true,true,null,null,null));
        $this->addCampo(new inteiro("ANO_MODELO","Ano do modelo",4,
                    null,null,true,true,true,null,null,null));
        $this->addCampo(new string("COR","Cor",20,
                    null,null,true,true,true,null,null,null,false,$_conn));
        $this->getCampo("MARCA")->SetComportamento_form("select");
        $this->getCampo("MARCA")->setValor_fixo(
            Array(
                Array('valor'=>'AUDI','label'=>'AUDI'),
                Array('valor'=>'BMW','label'=>'BMW'),
                Array('valor'=>'CHEVROLET','label'=>'CHEVROLET'),
                Array('valor'=>'CITROËN','label'=>'CITROËN'),
                Array('valor'=>'FIAT','label'=>'FIAT'),
                Array('valor'=>'FORD','label'=>'FORD'),
                Array('valor'=>'HONDA','label'=>'HONDA'),
                Array('valor'=>'MERCEDES','label'=>'MERCEDES'),
                Array('valor'=>'MITSUBISHI','label'=>'MITSUBISHI'),
                Array('valor'=>'PEUGEOT','label'=>'PEUGEOT'),
                Array('valor'=>'RENAULT','label'=>'RENAULT'),
                Array('valor'=>'TOYOTA','label'=>'TOYOTA'),
                Array('valor'=>'VOLKSWAGEN','label'=>'VOLKSWAGEN')
            )
        );
    }

    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(CODIGO) AS CODIGO FROM {$this->_nome_tabela}";
    }
```

```
    /**
     * Busca o próximo código para o menu
     *
     * @return mixed
     */
    public function incluir() {
        if(($_cod=$this->getProximoCodigo(true))=='[ERRO]') {
            return false;
        } else {
            $this->getCampo("CODIGO")->setValor($_cod);
            $this->getCampo("CODIGO")->setIncluir(true);
            return parent::incluir();
        }
    }
}
?>
```

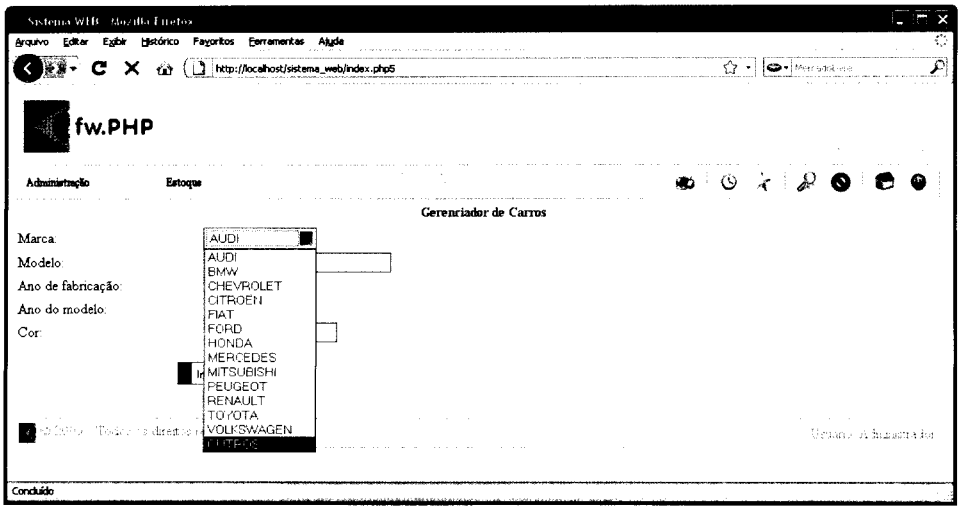


Figura 1.11

Outro recurso interessante disponível no framework é a possibilidade de limitar os valores inseridos nos campos do formulário. No caso de valores numéricos podemos estabelecer tanto o limite inferior quanto o limite superior, já para campos caractere (string) informamos apenas a quantidade mínima de caracteres aceitos, uma vez que a máxima já está definida no tamanho do campo.

Para campos do tipo data podemos informar uma data qualquer no formato padrão, ou seja, dia-mês-ano ou então a palavra reservada NOW que significa a data atual.

Para campos numéricos temos o recurso extra de informar um outro campo como referência tanto para o limite inferior quanto para o limite superior.

No caso da classe **carro** podemos realizar duas melhorias neste contexto. A primeira é somente aceitar carro com no máximo dez anos de fabricação, ou seja, o ano de fabricação (ANO_FABRICAO) deve ser maior ou igual ao ano atual menos 10, além disso o ano de fabricação não pode ser maior que o ano atual. A segunda alteração é que o ano do modelo (ANO_MODELO) não pode ser menor que o ano de fabricação, além disso não vamos aceitar que o ano do modelo seja maior que o ano atual somado 2.

A alteração na classe é simples, e pode ser realizada no momento da criação do objeto que representa o campo ou posteriormente com a utilização dos métodos virtuais *setMinimo()* e *setMaximo()*. Optaremos neste caso pela segunda forma.

Precisamos alterar apenas o construtor da classe, apresentado na listagem seguinte:

```
public function __construct(BancoDados $_conn) {
     parent::__construct($_conn);
     $this->_nome_tabela = 'carro';
     $this->_class_path = 'teste';
     $this->addCampo(new inteiro("CODIGO","Código",10,
               null,null,false,false,true,null,null,null,true));
     $this->addCampo(new string("MARCA","Marca",50,
               null,null,true,true,true,2,null,null,false,$_conn));
     $this->addCampo(new string("MODELO","Modelo",100,
               null,null,true,true,true,null,null,null,false,$_conn));
     $this->addCampo(new inteiro("ANO_FABRICACAO","Ano de fabricação",4,
               null,null,true,true,true,null,null,null));
     $this->addCampo(new inteiro("ANO_MODELO","Ano do modelo",4,
               null,null,true,true,true,null,null,null));
     $this->addCampo(new string("COR","Cor",20,
               null,null,true,true,true,null,null,null,false,$_conn));
     $this->getCampo("MARCA")->SetComportamento_form("select");
     $this->getCampo("MARCA")->setValor_fixo(
          Array(
               Array('valor'=>'AUDI','label'=>'AUDI'),
               Array('valor'=>'BMW','label'=>'BMW'),
               Array('valor'=>'CHEVROLET','label'=>'CHEVROLET'),
               Array('valor'=>'CITROËN','label'=>'CITROËN'),
               Array('valor'=>'FIAT','label'=>'FIAT'),
               Array('valor'=>'FORD','label'=>'FORD'),
               Array('valor'=>'HONDA','label'=>'HONDA'),
               Array('valor'=>'MERCEDES','label'=>'MERCEDES'),
               Array('valor'=>'MITSUBISHI','label'=>'MITSUBISHI'),
               Array('valor'=>'PEUGEOT','label'=>'PEUGEOT'),
               Array('valor'=>'RENAULT','label'=>'RENAULT'),
               Array('valor'=>'TOYOTA','label'=>'TOYOTA'),
               Array('valor'=>'VOLKSWAGEN','label'=>'VOLKSWAGEN'),
               Array('valor'=>'OUTROS','label'=>'OUTROS')
          )
     );
     $this->getCampo("ANO_FABRICACAO")->SetMinimo(date('Y')-10);
     $this->getCampo("ANO_FABRICACAO")->SetMaximo(date('Y'));
     $this->getCampo("ANO_MODELO")->SetMinimo('[ANO_FABRICACAO]');
     $this->getCampo("ANO_MODELO")->SetMaximo(date('Y')+2);
}
```

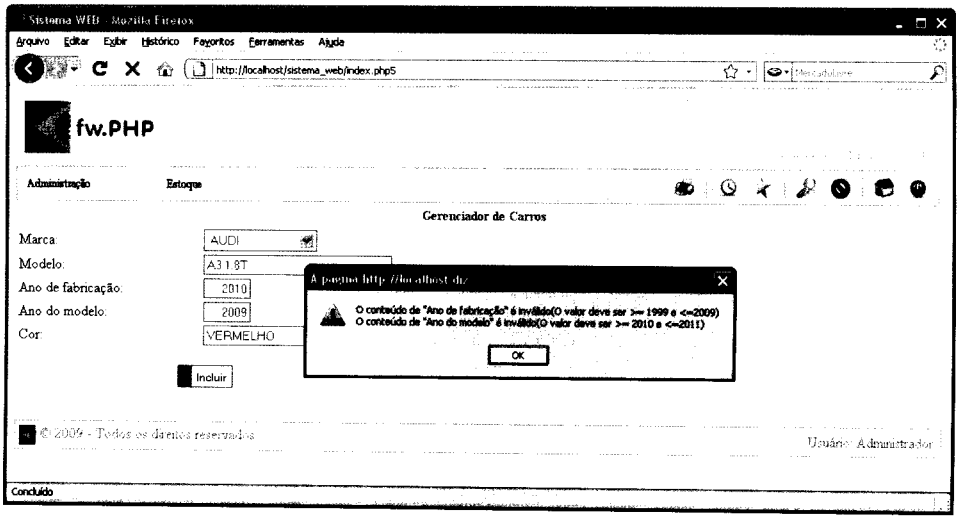


Figura 1.12

O framework disponibiliza também um modelo para gerar uma lista de opções conforme o conteúdo digitado em um determinado campo de referência. Basta declarar que o comportamento form do campo é 'ajax'. Tome como exemplo o campo MODELO. Para acrescentar essa funcionalidade ao campo precisamos apenas executar o método virtual *setcomportamento_form()* informando 'ajax' como parâmetro.

```
$this->getCampo("MODELO")->setComportamento_form("ajax");
```

Ao encontrar esta situação, o framework gera uma chamada à biblioteca *script.aculo.us* criando uma instância para Ajax.Autocomplete. O único problema é que não podemos vincular diretamente a busca a outro campo, que é o caso, ou seja, desejamos retornar os modelos conforme a marca selecionada. Para contornar este problema, devemos alterar a chamada javascript para incluir uma função *callback* em Ajax.Autocomplete (este é um dos parâmetros da criação do objeto). É preciso um pouco de javascript e um pequeno truque para "enganar" o framework. O truque está no fato de que podemos utilizar parâmetros extras na geração da chamada e por essa brecha vamos incluir essa referência (um outro meio mais formal, como veremos adiante, é a alteração do framework para tratamento desta situação).

```
$this->getCampo("MODELO")->setajax_params_extras(
        "',callback: function(v,qstr) {" .
        "return '&'+qstr+'&metodo=retornaModelos&marca='" .
        "+document.getElementById('MARCA').options[" .
        "document.getElementById('MARCA').selectedIndex].value;},x:'");
```

Temos definido que o método a ser chamado é *retornaModelos()* da classe **carro**. Além disso a marca selecionada também será enviada para o processamento no servidor web.

Vamos criar o método que retorna os modelos dos carros conforme a marca. Como é apenas um teste, deixaremos os modelos fixos por marca e trataremos algumas marcas (no mundo real devemos ter uma tabela de modelos por marca e o método buscaria a lista de modelos dentro da tabela).

```
public function retornaModelos() {
     $_opcoes = Array();
     switch($_POST['marca']) {
     case 'AUDI':       $_opcoes = Array(
                              Array('valor'=>'A3 1.6','label'=>'A3 1.6'),
                              Array('valor'=>'A3 1.8','label'=>'A3 1.8'),
                              Array('valor'=>'A3 1.8T',
                                   'label'=>'A3 1.8 Turbo 150cv'),
                              Array('valor'=>'A3 1.8T 180',
                                   'label'=>'A3 1.8 Turbo 180CV'),
                              Array('valor'=>'A3 2.0',
                                   'label'=>'A3 2.0 Aspirado'),
                              Array('valor'=>'A4 1.8T',
                                   'label'=>'A4 1.8 Turbo')
                        );
                        break;
     case 'VOLKSWAGEN': $_opcoes = Array(
                              Array('valor'=>'GOL 1.0','label'=>'GOL 1.0'),
                              Array('valor'=>'GOL 1.6','label'=>'GOL 1.6'),
                              Array('valor'=>'GOL 1.8','label'=>'GOL 1.8'),
                              Array('valor'=>'GOL 1.8 GT','label'=>'GOL GT 1.8'),
                              Array('valor'=>'GOLF 1.6','label'=>'GOLF 1.6'),
                              Array('valor'=>'GOLF 1.8 GTS','label'=>'GOL 1.8 GTS')
                        );
                        break;
     case 'FIAT':       $_opcoes = Array(
                              Array('valor'=>'PUNTO 1.4','label'=>'PUNTO 1.4'),
                              Array('valor'=>'PUNTO 1.8 SPORTING',
                                   'label'=>'PUNTO 1.8 Sporting'),
                              Array('valor'=>'PUNTO 1.4 T-JET',
                                   'label'=>'PUNTO T-Jet')
                        );
                        break;
     }
     $_resultado = Array();
     foreach($_opcoes as $_opcao) {
          if(stripos($_opcao['label'],$_POST['valor'])!==false) {
               $_resultado[] = $_opcao;
          }
     }
     return $this->retornaListAjax($_resultado);
}
```

Outro ponto interessante é a possibilidade de controle dos filtros de registros. A forma padrão de geração dos filtros faz com que o framework gere os campos com os mesmos valores definidos na classe para a geração de formulários. Esta situação gera um pequeno inconveniente com campos cujo comportamento no formulário é definido como 'select', pois uma vez que a lista é fixa, não existe a possibilidade de listar todos os registros, desconsiderando esse campo.

Há basicamente três formas para acertar esse comportamento. A primeira é retirar o campo das opções de filtragem, o que é feito através do método virtual *setFiltro(false)*, em que o parâmetro *false* indica que o campo não deve aparecer no formulário de filtragem. As duas outras formas envolvem o método existente no framework para preparação da classe antes da exibição do filtro, método chamado *prefiltro()*. Nesse método podemos fazer duas ações para contornar o problema. A primeira é alterar o comportamento do campo no formulário, alterando para o tipo padrão (text).

```
public function prefiltro() {
     $this->getCampo("MARCA")->SetComportamento_form("text");
}
```

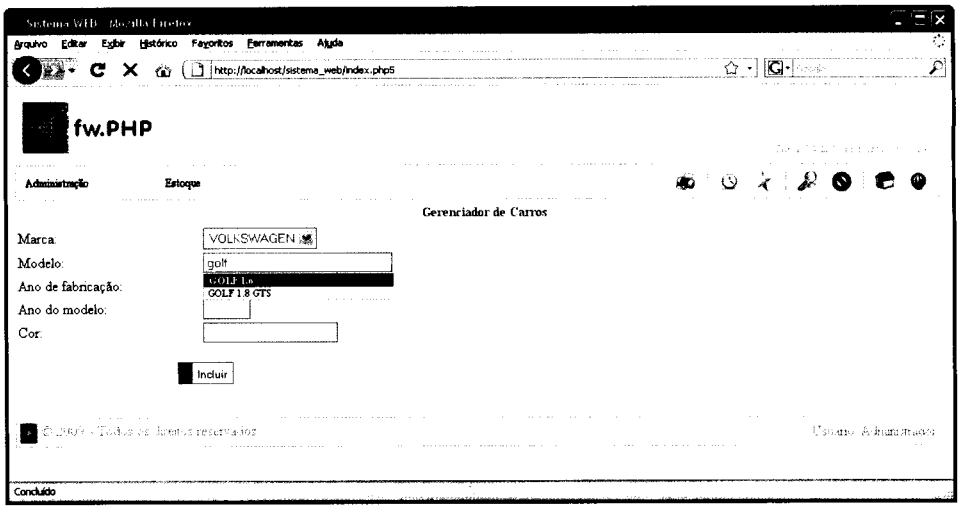


Figura 1.13

O problema dessa solução é que liberamos o usuário para digitar qualquer coisa que deseje, em vez de restringirmos a seleção a uma lista predeterminada. A segunda ação que podemos executar nesse método, e também a mais recomendada, é incluir uma nova opção na lista existente que será visualizada apenas no formulário de filtragem.

```
public function prefiltro() {
     $this->getCampo("MARCA")->setValor_fixo(array_merge(
                         Array(Array('label'=>"TODOS",'valor'=>'-')),
                               $this->getCampo("MARCA")->getValor_Fixo()));
}
```

Com isso teremos, além das opções normais, a opção extra "Todos".

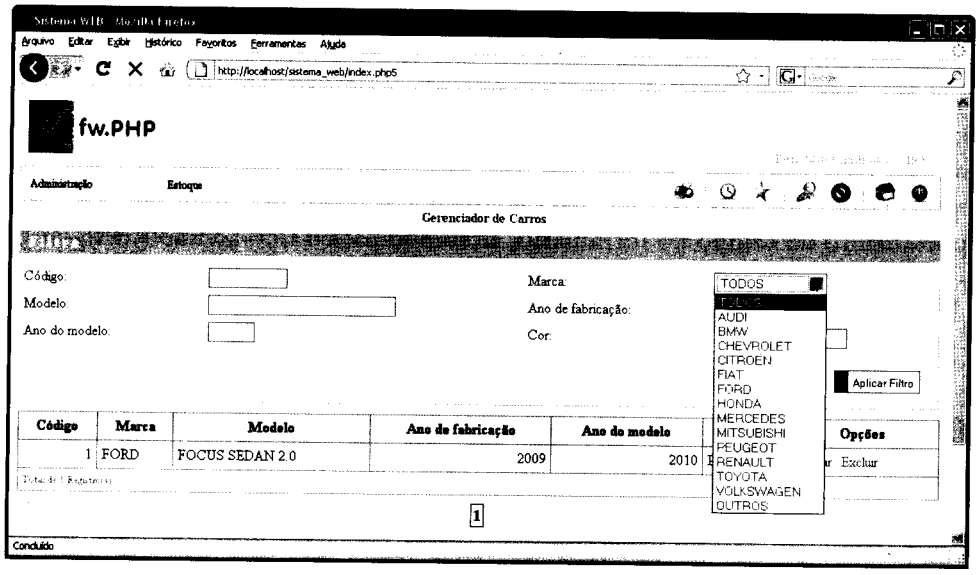


Figura 1.14

A opção para contornar o problema exige que o valor enviado pelo formulário seja tratado antes que o filtro seja executado, ou seja, antes da construção do comando SQL com as limitações definidas. Para tratar o valor, devemos sobrescrever o método *validavalor()*, o qual recebe dois parâmetros, um para identificar o campo e outro para o conteúdo do filtro. No exemplo não desejamos filtrar o campo MARCA quando o valor enviado for igual a '-', pois esse valor indica que todas as marcas devem ser consideradas. O método sobrescrito terá o seguinte formato:

```
protected function validaValor($_campo,$_valor) {
     if($_campo=='MARCA'&&$_valor=='-') {
          return false;
     }
     return true;
}
```

O framework disponibiliza vários outros recursos, tais como controle de impressão, histórico, favoritos, validação obrigatória ou não de campos de um formulário. Todos esses recursos são estudados no livro, alguns de forma mais detalhada.

Customizações necessárias ao framework

Sempre que desenvolvemos um sistema utilizando um framework, percebemos a necessidade de algumas customizações para atender às exigências específicas do processo de desenvolvimento. Uma necessidade já percebemos, que é a implementação de chamadas "callback" no processamento AJAX. Além dela, precisamos das seguintes customizações para a correta utilização do framework:

- Geração centralizada do usuário e data da última alteração;
- Processamento dos campos do formulário antes do envio para o browser;
- Geração do formulário com várias colunas;
- Foco no primeiro elemento do formulário;
- Busca com resultado no formato XML;
- Personalização da função atualizaFormulário;
- Filtro forçado para a classe;
- Geração personalizada de opções na lista de registros;
- Geração de relatórios;
- Mostrar e esconder botões;
- Exibição de mensagem personalizada.

Implementação de callback em ajax.autocomplete

Como visto anteriormente, o framework não dispõe de um meio direto para a implementação de uma chamada "callback" durante o processamento AJAX para a busca de dados dinâmicos no servidor. Para contornar este problema, desenvolvemos um meio de enganar o framework. O problema é que esses meios podem ficar muito complexos com o passar do tempo e dificultar o trabalho. Desta forma, vamos implementar este conceito no framework formalmente.

A classe **campo** deve ser alterada para que suporte um atributo para as chamadas callback e introduza, na geração do código html correspondente, o parâmetro callback na instanciação do objeto ajax.autocomplete.

A primeira alteração é simples. Precisamos apenas adicionar um novo atributo à classe **campo**, o qual chamaremos, por motivos óbvios, de $_callback.

```
/**
 * Classe abstrata para gerenciamento de Campos do framework
 *
 */
abstract class campo {
    protected $_nome;
    protected $_titulo;
    protected $_tipo;
    protected $_tamanho;
    protected $_formato;
    protected $_relacao;
    protected $_incluir;
    protected $_alterar;
    protected $_listar;
    protected $_minimo;
    protected $_maximo;
    protected $_camporef;
```

```
protected $_valor;
protected $_valor_anterior;
protected $_pk;
protected $_valor_fixo=null;
protected $_marcar=false;
protected $_comportamento_form=null;
protected $_url_ajax = 'executa_busca_ajax.php5';
protected $_ajax_params_extras = '';
protected $_classe = '';
protected $_arquivo_classe = '';
protected $_obrigatorio = true;
protected $_filtro = true; // Exibe o campo no filtro (default=sim)
protected $_relatorio = true; // Exibe o cmapo no relatório (default=sim)
// Customização
protected $_callback = null; // Define uma chamada callback ajax.autocomplete
```

A segunda alteração é no método *toForm()* para que o parâmetro callback seja tratado na geração do código html. Vamos alterar somente o ponto do método no qual ocorre a instanciação de ajax.autocomplete, introduzindo, quando necessário, o parâmetro callback na lista de parâmetros do construtor da classe.

```
public function toForm() {
    switch(strtolower($this->_comportamento_form)) {
    ...
    default:        $_cpofrm = new formInputTexto($this->_nome,
                                $this->_tipo,$this->_tamanho>30
                                    ? 30
                                    : $this->_tamanho,
                                $this->_tamanho);
                    // Ajax.Autocompleter
                    if(strtolower($this->_comportamento_form)=='ajax') {
                        $_cpofrm->addSubTag(new tag(new tipotag("SPAN"),
                            Array(new atributo("ID",
                                        "aguarde_{$this->_nome}"),
                            new atributo("STYLE","display:none;"
                            ))));
                        $_cpofrm->getLastSubTag()->addSubTag(
                            new tag(new tipotag("IMG",false),
                                Array(new atributo("SRC",
                                    "framework/imagens/spinner.gif"),
                                    new atributo("ALIGN",
                                                "ABSMIDDLE"))));
                        $_cpofrm->addSubTag(new tag(new tipotag("DIV"),
                            Array(new atributo("ID",
                                        "resultado_{$this->_nome}"),
                                new atributo("CLASS","autocomplete"))));
                        $_cpofrm->addSubTag(new tag(
                            new tipotag("SCRIPT"),
                            null,"new Ajax.Autocompleter('" .
                            "{$this->_nome}'," .
                            "'resultado_{$this->_nome}'," .
                            "'{$this->_url_ajax}', " .
                            "{paramName: 'valor'," .
                            "minChars: 2," .
                            "updateElement: atualizaFormulario, " .
                            "indicator: 'aguarde_{$this->_nome}'," .
                            "parameters: 'target=" .
                            "{$this->_nome}&classe={$this->_classe}&" .
                            "arquivo_classe={$this->_arquivo_classe}" .
                            "{$this->_ajax_params_extras}'" .
                            ($this->_callback!==null&&
```

```
                                        $this->_callback!=""
                                            ? ", callback: {$this->_callback}"
                                            : "") . "});"));
                                } else {
                                    $_cpofrm->setVerificaCampo($this->_tipo);

                                }
            ...
        }
    }
```

Geração centralizada de dados do usuário e data da última alteração

A maioria das tabelas do sistema, senão todas, tem dois campos comuns, um para determinar o usuário que realizou a última operação no registro, seja de inclusão ou alteração dos dados (no caso da exclusão, somente a trilha de auditoria pode ser usada), e um segundo campo que indica a data e a hora em que a manutenção foi realizada. Esses campos contêm sempre a alteração mais recente. Para rastrear todas as manutenções realizadas, deve-se utilizar a trilha de auditoria.

Como esses campos são comuns, não faz sentido criar um método em cada classe. Em vez disso, vamos criar um método na classe **base** com os ajustes nesses dois campos. Cada classe precisa somente executar o método antes da inclusão e alteração dos registros (uma outra forma seria ter um parâmetro na classe **base** que indique se o método deve ser executado ou não).

A alteração na classe **base** é simples, como mostrado a seguir.

```
protected function SetUsuarioeData() {
    try {
        $this->getCampo("USUARIO_LOGIN")->setValor(
                                            $_SESSION['USUARIO_LOGIN']);
        $this->getCampo("DATA_ULTIMA_ALTERACAO")->setValor(
                                            date("d-m-Y H:i:s"));
        $this->getCampo("DATA_ULTIMA_ALTERACAO")->setIncluir(true);
        $this->getCampo("USUARIO_LOGIN")->setIncluir(true);
        $this->getCampo("DATA_ULTIMA_ALTERACAO")->setAlterar(true);
        $this->getCampo("USUARIO_LOGIN")->setAlterar(true);
        return true;
    } catch(Exception $_e) {
        return false; // provavelmente um dos campos não existe
    }
}
```

A classe precisa apenas implementar nas rotinas de inclusão e alteração a chamada ao método. O exemplo seguinte mostra como fazer isso.

```
public function incluir() {
    $this->setUsuarioeData();
    return parent::incluir();
}

public function Alterar() {
    $this->setUsuarioeData();
    return parent::alterar();
}
```

Processamento dos campos do formulário antes do envio para o browser

Em algumas situações é necessário um processamento extra dos campos do formulário, por exemplo, para a inclusão de um novo atributo ou evento relacionado a um ou mais campos do formulário. Esse processamento deve ser flexível, permitindo ao desenvolvedor realizar a customização conforme a atividade realizada, ou seja, o desenvolvedor deve ter condições de saber se o formulário que está sendo gerado é de manutenção do cadastro ou de filtro de registros, tendo assim condições de decidir quando implementar as alterações nos campos do formulário.

Para realizar esse processamento vamos incluir na classe **base** do framework um novo método que chamaremos de *processaCampoFormulario()*, o qual será executado sempre que um formulário for gerado no framework, seja na página de manutenção (inclusão ou alteração) ou no formulário para filtragem dos registros.

O novo método vai receber três parâmetros para que seja possível o correto tratamento dos campos:

- Nome do campo
- Instância da classe tag
- Nome do método que está em execução

O primeiro parâmetro informa ao método o campo que está sendo tratado. Com isso temos que o método será executado n vezes, em que n é igual ao número de campos definidos na classe. Desta forma, o desenvolvedor pode selecionar os campos que devem ser tratados e em quais situações.

O segundo parâmetro é a instância da classe **tag** (na verdade, é uma instância da classe **formCampo**, mas para melhor generalização do método, a classe **tag** foi escolhida) passada como referência. Você sabe o que é passagem por referência? O padrão da passagem de parâmetros para funções e métodos no PHP é por valor, ou seja, as alterações realizadas dentro da função não são refletidas na variável original enviada como argumento dela. Para que as alterações reflitam na variável original, devemos utilizar a passagem de parâmetros por referência. Com isso todas as alterações da variável dentro da função refletem na variável original. Para declarar um parâmetro como passado por referência devemos inserir o símbolo '&' antes da declaração do parâmetro, por exemplo, &$_param.

O terceiro e último parâmetro do método é o nome do método que originou a chamada, por exemplo, *base::getFormulario*. Com esta informação o desenvolvedor pode decidir em quais métodos chamadores o processamento deve ser executado.

Na classe **base** o método *processaCampoFormulario()* é apenas uma casca vazia, não implementando nenhuma alteração nos campos do formulário. Para que o tratamento seja executado, devemos implementar o que for necessário dentro das classes específicas.

A seguir encontra-se a codificação do método dentro da classe **base**:

```
/**
 * Processa um campo do formulário antes de sua inclusão
 *
 * @param string $_nome
 * @param tag    $_campo
 * @param string $_metodo
 */
protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
     // Deve ser implementado na classe destino
}
```

Precisamos, ainda, implementar as chamadas ao método dentro da classe **base**. Temos dois métodos que executam a rotina de processamento do formulário, *getFormulario()* e *getFormularioFiltro()*. Vamos alterar os pontos nos dois métodos em que os campos são inseridos no formulário. Em cada método há dois pontos nos quais é necessário inserir a chamada ao método *processaCampoFormulario()*. O primeiro é quando o método está tratando campos com comportamento form igual a 'radio' e o segundo é no tratamento dos demais campos do formulário. A seguir temos os dois métodos com suas alterações.

```
public function getFormulario($_funcao="INC") {
     $_fcn  = ($_funcao!='ALT' ? "incluir" : 'alterar');
     $_form = new formulario("FRM_{$this->_nome_tabela}","index.php5");
     $_form->addSubTag(new tag(new tipotag("SCRIPT"),
                               null,"objForm.construct();"));
     $_form->addSubTag(new formInputHidden("SCRIPT_NAME",
                                           "{$_SERVER['PHP_SELF']}"));
     $_form->addSubTag(new formInputHidden("ACAO","SALVAR"));
     $_form->addSubTag(new formInputHidden("FUNCAO",$_funcao));
     if($_funcao!='INC') {
          // Devemos gravar as chaves
          foreach($this->filtrarCampos("pk") as $_pk) {
               $_form->addSubTag(new formInputHidden($_pk->getNome(),
                                                     $_pk->getValor()));
          }
          if($_funcao=='ALT') {
               // Vamos gerar os valores Atuais dos campos
               foreach($this->filtrarCampos("alterar") as $_campo) {
                    $_form->addSubTag(new formInputHidden(
                                           "{$_campo->getNome()}_OLD",
                                           $_campo->toHTML()));
               }
          }
     }
     $_tab  = new tag(new tipotag("TABLE"),
                      Array(new atributo("BORDER",0),
                            new atributo("CELLPADDING",2),
                            new atributo("CELLSPACING",0)));
     $_tr = new tipotag("TR");
     $_td = new tipotag("TD");
     // fcn=ALT
     if($_funcao=='ALT') {
          // Vamos listar as chaves
          foreach($this->filtrarCampos("pk") as $_pk) {
               $_tab->addSubTag(new tag($_tr));
               $_tab->getLastSubTag()->addSubTag(new tag($_td,
```

```
                    Array(new atributo("STYLE","color:Navy;width:200px;")),
                    "{$_pk->getTitulo()}:"));
            $_tab->getLastSubTag()->addSubTag(new tag($_td));
            $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
                    new tipotag("SPAN"),
                    Array(new atributo("STYLE","color:#ff0011;")),
                    $_pk->toHTML()));
        }
    }
    foreach($this->filtrarCampos($_fcn) as $_campo) {
        $_tab->addSubTag(new tag($_tr));
        $_tab->getLastSubTag()->addSubTag(new tag($_td,
                Array(new atributo("STYLE","color:Navy;width:200px;")),
                "{$_campo->getTitulo()}:"));
        $_tab->getLastSubTag()->addSubTag(new tag($_td));
        // fcn=Exc
        if($_funcao=='EXC') {
            $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
                new tipotag("SPAN"),
                Array(new atributo("STYLE","color:#ff0011;")),
                $_campo->toHTML()));
        } else {
            if(strtolower($_campo->getComportamento_form())=='radio'&&
               is_array($_campo->getValor_fixo())) {
                foreach($_campo->getValor_Fixo() as $_opcao) {
                    $_cpoform = new formInputRadio($_campo->getNome(),
                                $_opcao['valor'],$_opcao['label'],null,null,
                                ($_opcao['marcar']===true||
                                $_campo->getValor()==$_opcao['valor']));
                    $this->processaCampoFormulario($_campo->getNome(),
                                                   $_cpoform,
                                                   __METHOD__);
                    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                                                                $_cpoform);
                    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                             new tag(new tipotag('BR',false)));
                }
            } else {
                $_cpoform = $_campo->toForm();
                $this->processaCampoFormulario($_campo->getNome(),
                                               $_cpoform,
                                               __METHOD__);
                $_tab->getLastSubTag()->getLastSubTag()->addSubTag($_cpoform);
            }
            if(strtolower($_campo->getComportamento_form())=='file') {
                $_form->addAtributo(new atributo("ENCTYPE",
                                                 "multipart/form-data"));
            }
        }
    }
    $_tab->addSubTag(new tag($_tr));
    $_tab->getLastSubTag()->addSubTag(new tag($_td,
                Array(new atributo("COLSPAN",2),
                    new atributo('ALIGN','CENTER'),
                    new atributo("ID","BTNOK"))));
    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new formInputSubmit("OK",
                ($_funcao=='INC'
                    ? ' Incluir '
                    : ($_funcao=='ALT'
                        ? ' Alterar'
                        : ' Confirmar Exclusão '))));
    $_form->addSubTag($_tab);
    return utf8_encode($_form->toHTML(false));
}
```

```
/**
 * Gera o formulário para filtro de conteúdo
 *
 * @return mixed
 */
public function getFormularioFiltro() {
     $this->preFiltro();
     $_filtro = $this->getFiltro();
     $_form = new formulario("FILTRO_{$this->_nome_tabela}",
                             "Javascript:void(0);");
     $_form->deleteAtributo(4);
     $_form->addAtributo(new atributo("ONSUBMIT",
                        "ObjProcAjax.runPost('{$_POST['programa']}','CORPO'," .
                        "retornaFiltro(this));processaFiltro();return false;"));
     $_tab  = new tag(new tipotag("TABLE"),
                       Array(new atributo("BORDER",0),
                             new atributo("CELLPADDING",2),
                             new atributo("CELLSPACING",0),
                             new atributo("WIDTH","100%")));
     $_tr = new tipotag("TR");
     $_td = new tipotag("TD");
     $_tab->addSubTag(new tag($_tr));
     $_tab->getLastSubTag()->addSubTag(new tag($_td,
              Array(new atributo("COLSPAN",4),
                    new atributo("STYLE","text-align:left;font-weight:bold;" .
                                 "color:#ffffff;background-color:#aaccff;" .
                                 "border-bottom:1px solid #c0c0c0;" .
                                 "padding-left:10px;")),
              "F i l t r o"));
     $_tab->addSubTag(new tag($_tr));
     $_tab->getLastSubTag()->addSubTag(new tag($_td,
              Array(new atributo("COLSPAN",4),
                new atributo("STYLE","height:10px;font-size:1px;"))," "));
     // Colunas ...
     $_colunas = 3;
     foreach($this->filtrarCampos('filtro') as $_chv=>$_campo) {
          if($_colunas>2) {
               $_tab->addSubTag(new tag($_tr));
               $_colunas = 1;
          }
          ++$_colunas;
          if(isset($_filtro[$_chv])) {
               $_campo->setValor(utf8_decode($_filtro[$_chv]));
          }
          $_tab->getLastSubTag()->addSubTag(new tag($_td,
               Array(new atributo("STYLE","color:Navy;width:200px;padding:5px;")),
               "{$_campo->getTitulo()}:"));
          $_tab->getLastSubTag()->addSubTag(new tag($_td));
          if(strtolower($_campo->getComportamento_form())=='radio'&&
             is_array($_campo->getValor_fixo())) {
               foreach($_campo->getValor_Fixo() as $_opcao) {
                    $_cpoform = new formInputRadio($_campo->getNome(),
                                   $_opcao['valor'],$_opcao['label'],null,null,
                                   $_opcao['marcar']===true||
                                   $_campo->getValor()==$_opcao['valor']);
                    $_tab->getLastSubTag()->getLastSubTag()->addSubTag($_cpoform);
                    $this->processaCampoFormulario($_campo->getNome(),
                                                   $_cpoform,
                                                   __METHOD__);
                    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                              new tag(new tipotag('BR',false)));
               }
          } else {
               $_cpoform = $_campo->toForm();
```

```
                $this->processaCampoFormulario($_campo->getNome(),
                                               $_cpoform,
                                               __METHOD__);
                $_tab->getLastSubTag()->getLastSubTag()->addSubTag($_cpoform);
            }
        }
        $_tab->addSubTag(new tag($_tr));
        $_tab->getLastSubTag()->addSubTag(new tag($_td,
                Array(new atributo("COLSPAN",4),
                    new atributo('STYLE','text-align:right;padding-right:20px;'),
                    new atributo("ID","BTNOK"))));
        $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                new formInputSubmit("OK"," Aplicar Filtro "));
        $_form->addSubTag($_tab);
        echo utf8_encode($_form->toHTML(false));
}
```

O código a seguir mostra um exemplo de utilização dessa nova funcionalidade. Neste exemplo testamos se o método é o de geração do formulário e se o campo que está sendo enviado tem o nome 'COR'. Em caso afirmativo, é inserido um evento no campo.

```
protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
    if(stripos(strtolower($_metodo),'getFormulario')!==false) {
        if($_nome=='COR') {
            $_ev = new Eventos();
            $_ev->setOnclick("alert('Estamos aqui...'+this.value);");
            $_campo->setEventos($_ev);
        }
    }
}
```

Geração do formulário com várias colunas

O padrão atual do framework é que cada campo da classe ocupe uma linha da tabela gerada para o formulário. Para classes com poucos campos, digamos até quatro ou cinco campos, não há nenhum problema nesta solução, porém quando uma classe possui muitos campos, dez, quinze ou mais, a página fica muito comprida, o que em geral não é esteticamente agradável, pois obriga o usuário a rolar a página para ver parte do formulário. Para contornar este problema vamos introduzir no framework a possibilidade de ter mais de um campo por linha da tabela.

Esta nova facilidade do framework exige a criação de um atributo e a alteração do método que gera o formulário. Essas alterações são executadas na classe **base**, pois é nela que o formulário é gerado. A seguir veja a lista das alterações necessárias:

- Incluir um novo atributo para determinação do número de campos em cada linha da tabela gerada para o formulário. Caso o atributo não seja alterado, seu valor padrão será de um campo por linha da tabela.

➲ Alterar o método geraFormulario() para que o novo atributo seja utilizado na geração da tabela. Antes de utilizar o atributo definido, o método verifica se ele contém um valor válido, ou seja, se o valor é maior ou igual a 1 e menor ou igual a 10 (com isso limitamos a dez campos por linha da tabela).

Chamaremos o novo atributo da classe **base** de $_num_colunas_form. A seguir observe uma visão parcial da classe **base** com as alterações necessárias.

```
/**
 * Classe Base do framework, deverá ser estendida nas classes filhas
 *
 */
class BASE {
    protected $_nome_tabela;
    protected $_campos = Array();
    protected $_msgextra = "";
    /**
     * @var BancoDados
     */
    protected $_conn = null;
    protected $_class_path = "framework";
    protected $_exibe_opcoes = true;
    protected $_num_colunas_form = 1; // Padrão é 1 coluna

    ...
    public function getFormulario($_funcao="INC") {
        $_fcn  = ($_funcao!='ALT' ? "incluir" : 'alterar');
        $_form = new formulario("FRM_{$this->_nome_tabela}","index.php5");
        $_form->addSubTag(new tag(new tipotag("SCRIPT"),null,
                                  "objForm.construct();"));
        $_form->addSubTag(new formInputHidden("SCRIPT_NAME",
                                  "{$_SERVER['PHP_SELF']}"));
        $_form->addSubTag(new formInputHidden("ACAO","SALVAR"));
        $_form->addSubTag(new formInputHidden("FUNCAO",$_funcao));
        if($_funcao!='INC') {
            // Devemos gravar as chaves
            foreach($this->filtrarCampos("pk") as $_pk) {
                $_form->addSubTag(
                    new formInputHidden($_pk->getNome(),$_pk->getValor()));
            }
            if($_funcao=='ALT') {
                // Vamos gerar os valores Atuais dos campos
                foreach($this->filtrarCampos("alterar") as $_campo) {
                    $_form->addSubTag(
                        new formInputHidden("{$_campo->getNome()}_OLD",
                                            $_campo->toHTML()));
                }
            }
        }
        $_tab  = new tag(new tipotag("TABLE"),
                         Array(new atributo("BORDER",0),
                               new atributo("CELLPADDING",2),
                               new atributo("CELLSPACING",0)));
        $_tr = new tipotag("TR");
        $_td = new tipotag("TD");
        // fcn=ALT
        if($_funcao=='ALT') {
            // Vamos listar as chaves
            foreach($this->filtrarCampos("pk") as $_pk) {
                $_tab->addSubTag(new tag($_tr));
                $_tab->getLastSubTag()->addSubTag(new tag($_td,Array(
```

```
                    new atributo("STYLE","color:Navy;width:200px;")),
                    "{$_pk->getTitulo()}:"));
          $_tab->getLastSubTag()->addSubTag(new tag($_td));
          $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
               new tag(new tipotag("SPAN"),
                         Array(new atributo("STYLE","color:#ff0011;")),
                         $_pk->toHTML()));
     }
}
$_num_colunas = ($this->_num_colunas_form<1||$this->_num_colunas_form>5
                    ? 1
                    : $this->_num_colunas_form);
$_cols = $_num_colunas+1;
foreach($this->filtrarCampos($_fcn) as $_campo) {
     if($_cols>$_num_colunas) {
          $_tab->addSubTag(new tag($_tr));
          $_cols = 1;
     }
     ++$_cols;
     $_tab->getLastSubTag()->addSubTag(new tag($_td,
          Array(new atributo("STYLE",
                         "color:Navy;width:200px;padding-left:5px;")),
          "{$_campo->getTitulo()}:"));
     $_tab->getLastSubTag()->addSubTag(new tag($_td));
     // fcn=Exc
     if($_funcao=='EXC') {
          $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
               new tipotag("SPAN"),
               Array(new atributo("STYLE","color:#ff0011;")),
               $_campo->toHTML()));
     } else {
          if(strtolower($_campo->getComportamento_form())=='radio'&&
             is_array($_campo->getValor_fixo())) {
               foreach($_campo->getValor_Fixo() as $_opcao) {
                    $_cpoform = new formInputRadio($_campo->getNome(),
                                   $_opcao['valor'],$_opcao['label'],
                                   null,null,
                                   $_opcao['marcar']===true||
                                   $_campo->getValor()==$_opcao['valor']));
                    $this->processaCampoFormulario($_campo->getNome(),
                                                   $_cpoform,
                                                   __METHOD__);
                    $_tab->getLastSubTag()->
                              getLastSubTag()->addSubTag($_cpoform);
                    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                              new tag(new tipotag('BR',false)));
               }
          } else {
               $_cpoform = $_campo->toForm();
               $this->processaCampoFormulario($_campo->getNome(),
                                              $_cpoform,__METHOD__);
               $_tab->getLastSubTag()->
                         getLastSubTag()->addSubTag($_cpoform);

          }
          if(strtolower($_campo->getComportamento_form())=='file') {
               $_form->addAtributo(new atributo("ENCTYPE",
                                        "multipart/form-data"));
          }
     }
}
$_tab->addSubTag(new tag($_tr));
$_tab->getLastSubTag()->addSubTag(new tag($_td,
          Array(new atributo("COLSPAN",2),
```

```
                    new atributo('ALIGN','CENTER'),
                    new atributo("ID","BTNOK"))));
        $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                new formInputSubmit("OK",($_funcao=='INC'
                    ? ' Incluir '
                    : ($_funcao=='ALT'
                        ? ' Alterar'
                        : ' Confirmar Exclusão '))));
        $_form->addSubTag($_tab);
        return utf8_encode($_form->toHTML(false));
    }
    ...
}
```

Para definir o número de colunas que desejamos no formulário de cadastramento gerado pelo framework, basta alterar o novo atributo. Veja o exemplo em seguida.

```
$this->_num_colunas_form = 2;
```

O resultado é exibido na Figura 1.15.

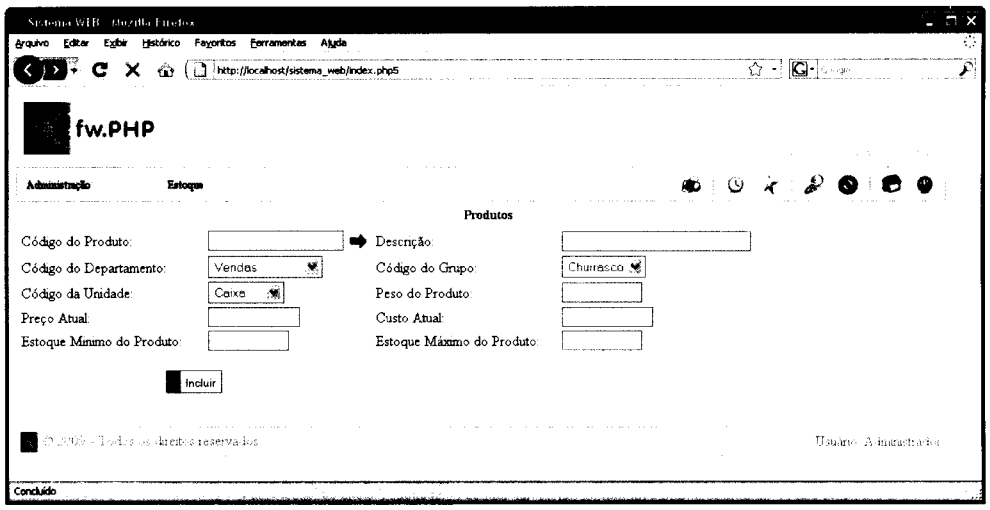


Figura 1.15

Foco no primeiro elemento do formulário

Quando o formulário para cadastramento é gerado, seja uma inclusão ou alteração, é desejável que o primeiro elemento disponível do formulário receba o foco, ou seja, que o cursor esteja nesse elemento, evitando que o usuário tenha de clicar no campo para começar o processo de digitação. Esta funcionalidade não existe originalmente no framework **fw.PHP**, por isso vamos incluí-lo agora.

É preciso apenas alterar o método *getFormulario()* da classe **base**, incluindo umas poucas linhas. Devemos executar os seguintes passos:

1. Descobrir qual o primeiro elemento disponível do formulário.
2. No final do formulário inserir o comando javascript para focar esse elemento.

O primeiro passo é simples, uma vez que sabemos o ponto exato de geração dos elementos do formulário. Basta que o nome do campo seja guardado em uma variável auxiliar para o estabelecimento do foco mais adiante. Chamaremos essa variável de $_foco, sendo seu valor inicial nulo (null).

O segundo passo, também simples, deve ser executado após a geração completa do formulário. Basta inserir um novo marcador html, do tipo SCRIPT, e colocar o comando apropriado para que o campo receba o foco:

```
document.getElementById(<nome>).focus();
```

Lembre-se de que, em algumas situações, pode não haver um elemento para receber o foco, por exemplo, ao executar a opção de exclusão. Para contornar esta situação, devemos verificar, antes de gerar o comando javascript, se a variável auxiliar $_foco contém um valor diferente de nulo (null).

O método *getFormulario()* da classe **base** com as alterações necessárias para gerar o foco no primeiro elemento disponível do formulário é exibido em seguida:

```
public function getFormulario($_funcao="INC") {
     $_fcn  = ($_funcao!='ALT' ? "incluir" : 'alterar');
     $_form = new formulario("FRM_{$this->_nome_tabela}","index.php5");
     ...
     $_num_colunas = ($this->_num_colunas_form<1||$this->_num_colunas_form>5
                      ? 1
                      : $this->_num_colunas_form);
     $_cols = $_num_colunas+1;
     $_foco = null;
     foreach($this->filtrarCampos($_fcn) as $_campo) {
          if($_cols>$_num_colunas) {
               $_tab->addSubTag(new tag($_tr));
               $_cols = 1;
          }
          ++$_cols;
          $_tab->getLastSubTag()->addSubTag(new tag($_td,
                    Array(new atributo("STYLE","color:Navy;width:200px;" .
                                               "padding-left:5px;")),
                    "{$_campo->getTitulo()}:"));
          $_tab->getLastSubTag()->addSubTag(new tag($_td));
          // fcn=Exc
          if($_funcao=='EXC') {
               $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
                    new tipotag("SPAN"),
                    Array(new atributo("STYLE","color:#ff0011;")),
                    $_campo->toHTML()));
          } else {
               if($_foco===null) {
                    $_foco = $_campo->getNome();
               }
               if(strtolower($_campo->getComportamento_form())=='radio'&&
                  is_array($_campo->getValor_fixo())) {
                    foreach($_campo->getValor_Fixo() as $_opcao) {
                         $_cpoform = new formInputRadio($_campo->getNome(),
```

```
                                        $_opcao['valor'],$_opcao['label'],
                                        null,null,
                                        ($_opcao['marcar']===true||
                                        $_campo->getValor()==$_opcao['valor']));
                    $this->processaCampoFormulario($_campo->getNome(),
                                        $_cpoform,__METHOD__);
                    $_tab->getLastSubTag()->getLastSubTag()->
                                        addSubTag($_cpoform);
                    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                                        new tag(new tipotag('BR',false)));
                }
            } else {
                $_cpoform = $_campo->toForm();
                $this->processaCampoFormulario($_campo->getNome(),
                                        $_cpoform,__METHOD__);
                $_tab->getLastSubTag()->getLastSubTag()->addSubTag($_cpoform);
            }
            if(strtolower($_campo->getComportamento_form())=='file') {
                $_form->addAtributo(new atributo("ENCTYPE",
                                        "multipart/form-data"));
            }
        }
    }
    $_tab->addSubTag(new tag($_tr));
    $_tab->getLastSubTag()->addSubTag(new tag($_td,
                Array(new atributo("COLSPAN",2),
                    new atributo('ALIGN','CENTER'),
                    new atributo("ID","BTNOK"))));
    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
            new formInputSubmit("OK",($_funcao=='INC'
                        ? ' Incluir '
                        : ($_funcao=='ALT'
                            ? ' Alterar'
                            : ' Confirmar Exclusão '))));
    $_form->addSubTag($_tab);
    if($_foco!==null) {
        $_form->addSubTag(new tag(new tipotag("SCRIPT"),null,
                            "document.getElementById('{$_foco}').focus();"));
    }
    return utf8_encode($_form->toHTML(false));
}
```

Busca com resultado no formato XML

Em algumas situações é necessário que o resultado de uma solicitação ao servidor web retorne os dados no formato XML, visando a algum tipo de manipulação pelo código javascript no browser.

Para implementar esse recurso no framework, é necessário que sejam efetuadas alterações tanto no lado servidor, basicamente na classe **base**, quanto no lado do cliente, pois é preciso informar explicitamente na chamada AJAX que os dados retornados estão no formato XML.

No lado servidor, tudo que precisamos é implementar um método que, após realizar a consulta no banco de dados, gere o resultado no formato XML. O formato a ser gerado é:

```
<?xml version="1.0"?>
<BUSCA>
     <LINHA>
          .. dados do registro 1
     </LINHA>
     ...
     <LINHA>
          .. dados do registro n
     </LINHA>
</BUSCA>
```

Vamos utilizar a classe DOM do PHP para a geração do XML de retorno. A seguir temos a implementação do método *buscaXML()* na classe **base**.

```
public function buscarXML() {
     $_campos = explode(",",$_POST['campos']);
     $_valores= explode(",",$_POST['valores']);
     $_arr = Array();
     foreach($_campos as $_k=>$_campo) {
          $_arr[$_campo] = $_valores[$_k];
     }
     $_xml = new DOMDocument('1.0');
     $_no  = $_xml->createElement('BUSCA');
     $_no  = $_xml->appendChild($_no);
     if($this->Buscar($_arr)!==false) {
          // Montar o XML
          while($this->proximo()!==false) {
               $_linha = $_xml->createElement('LINHA');
               $_no->appendChild($_linha);
               foreach($this->_campos as $_chv=>$_campo) {
                    $_filho = $_xml->createElement($_chv);
                    $_valor = $_xml->createTextNode(
                                        utf8_encode($_campo->toHTML()));
                    $_filho->appendChild($_valor);
                    $_linha->appendChild($_filho);
               }
          }
     }
     header('Content-Type: text/xml;');
     echo $_xml->saveXML();
}
```

Um detalhe importante é que devemos alterar o tipo de dado enviado ao browser. Isso é feito pelo comando *header()* do PHP. Neste caso vamos enviar o parâmetro 'Content-Type' com o valor 'text/xml'.

No lado do cliente é necessário criar um método para processamento de requisições AJAX no modo XML. A classe para chamada AJAX já suporta o processamento do resultado no formato XML. Basta que o atributo modo seja alterado para 'X'. Com isso, em vez de a classe enviar para processamento o resultado como texto, envia como XML, o que ocorre por meio do atributo httprequest.responseXML em vez de httprequest.responseText.

A implementação do processamento no formato XML é feita no método runPostXML da classe processaAjax. Sua codificação está listada a seguir.

```
runPostXML:    function(url,params,after) {
                    var ax=new AJAX();
                    ax.url=url;
                    ax.obj=this;
                    ax.metodo='POST';
                    ax.params=params;
                    ax.modo='X';
                    ax.after=after;
                    ax.processaresultado=function(xml) {
                                                this.after(xml);
                                            };
                    ax.conectar();
               }
```

Sua utilização é idêntica à do método que retorna o resultado no formato texto. A diferença está na função que processa o retorno. Veja o exemplo a seguir:

```
ObjProcAjax.runPostXML('executa_busca_ajax.php5',params,objBP.processa);
processa:
function(xml){
          var l = xml.childNodes[0].childNodes[0];
          var vlr = l.getElementsByTagName('PRODUTO_DESC')[0];
          document.getElementById('PRODUTO_DESC').value =
                              (vlr.hasChildNodes()
? vlr.firstChild.nodeValue
: '');
          var vlr = l.getElementsByTagName('PRODUTO_ESTOQUE')[0];
          document.getElementById('PRODUTO_ESTOQUE').value =
 (vlr.hasChildNodes()
? vlr.firstChild.nodeValue
: '0,000');
          var vlr = l.getElementsByTagName('PRODUTO_CUSTOMEDIO')[0];
          document.getElementById('PRODUTO_CUSTOMEDIO').value =
 (vlr.hasChildNodes()
? vlr.firstChild.nodeValue
:  '0,00');
          objBP.FOk=true;
          // Se for uma saida, colocar valor_unitario=preco_atual ou
custo_atual????
          if(this.FEntrada!==true&&this.FEstorno!==true) {
               document.getElementById('HISTORICO_VALOR_UNIT').value =
        document.getElementById('PRODUTO_CUSTOMEDIO').value;
               document.getElementById('CUSTO_SAIDA').value =
                        document.getElementById('PRODUTO_CUSTOMEDIO').value;
          };
          document.getElementById('LOCAL_CODIGO').focus();
    }
```

Personalizar função atualizaFormulario

No desenvolvimento de um sistema, às vezes desejamos personalizar um dos comportamentos-padrão do framework. Um exemplo é quando desejamos definir um tratamento diferente para o retorno do método de busca dinâmica existente no framework, o qual utiliza a ferramenta AutoComplete da classe scriptaculous. No formato original do framework fw.PHP o tratamento é sempre feito pela função *atualizaFormulario* definida no arquivo funcoesgenericas.js. Essa função tem um

comportamento-padrão eficaz na maioria das vezes, porém há situações em que precisamos incrementá-lo ou alterá-lo.

Para alterar esse comportamento, devemos mudar o formato fixo existente no framework (na montagem da chamada a Ajax.autocompleter o atributo updateElement sempre recebe a função atualizaFormulario) por outro dinâmico, e o padrão deve ser a função atualizaFormulario. Para executar essa alteração, devemos criar na classe **campo** o atributo updateelement, cujo padrão de criação é 'atualizaFormulario', em seguida alterar o método *toForm()* da mesma classe para que o atributo updateElement receba, em vez da constante atualizaFormulario, o conteúdo do atributo updateeleement.

```
/**
 * Classe abstrata para gerenciamento de Campos do framework
 *
 */
abstract class campo {
     protected $_nome;
     ...
     // Customização
     protected $_callback = null; // Define uma chamada callback ajax.autocomplete
     protected $_updateelement = "atualizaFormulario";
     ...
     public function toForm() {
          switch(strtolower($this->_comportamento_form)) {
          ...
          default:
               $_cpofrm = new formInputTexto($this->_nome,$this->_tipo,
                                        $this->_tamanho>30
                                             ? 30
                                             : $this->_tamanho,$this->_tamanho);
               // Ajax.Autocompleter
               if(strtolower($this->_comportamento_form)=='ajax') {
                    $_cpofrm->addSubTag(new tag(new tipotag("SPAN"),
                              Array(new atributo("ID","aguarde_{$this->_nome}"),
                                   new atributo("STYLE","display:none;"))));
                    $_cpofrm->getLastSubTag()->addSubTag(new tag(
                              new tipotag("IMG",false),
                              Array(new atributo("SRC",
                                             "framework/imagens/spinner.gif"),
                                   new atributo("ALIGN","ABSMIDDLE"))));
                    $_cpofrm->addSubTag(new tag(new tipotag("DIV"),
                              Array(new atributo("ID","resultado_{$this->_nome}"),
                                   new atributo("CLASS","autocomplete"))));
                    $_cpofrm->addSubTag(new tag(new tipotag("SCRIPT"),null,
                              "new Ajax.Autocompleter('{$this->_nome}'," .
                              "'resultado_{$this->_nome}', '{$this->_url_ajax}'," .
                              "{paramName: 'valor'," .
                              "minChars: 2, " .
                              "updateElement: {$this->_updateelement}, " .
                              "indicator: 'aguarde_{$this->_nome}'," .
                              "parameters: 'target=" .
                              "{$this->_nome}&classe={$this->_classe}&" .
                              "arquivo_classe={$this->_arquivo_classe}" .
                              "{$this->_ajax_params_extras}'" .
                              ($this->_callback!==null&&$this->_callback!=""
                                             ? ", callback: {$this->_callback}"
                                             : "") . "});"));
                     } else {
```

```
                                $_cpofrm->setVerificaCampo($this->_tipo);

                            }
            ...
        }
        if($this->_valor!==null&&in_array(strtolower(
           $this->_comportamento_form),Array('password','file','ajax',null))) {
             $_cpofrm->addAtributo(new atributo('VALUE',$this->toHTML()));
        }
        return $_cpofrm;
    }
    ...
}
```

A utilização desta nova funcionalidade é bem simples. Veja o exemplo a seguir:

```
$this->getCampo("PRODUTO_CODIGO")->setUpdateElement(
     "function(li) {atualizaFormulario(li);objBP.buscar();}");
```

Filtro forçado para a classe

Quando lidamos com o desenvolvimento de um sistema qualquer, sempre ocorrem situações nas quais desejamos controlar o que será exibido na tela de seleção de registros, por exemplo, limitando a seleção ao conteúdo de algum campo. Isso pode ser feito manualmente pelo usuário com a utilização do filtro de registro, mas em alguns momentos desejamos forçar o filtro, por exemplo, quando temos tabelas relacionadas cujo conteúdo a ser exibido depende sempre do que foi selecionado em outra tabela (como notas fiscais e itens de nota).

Para inserir esta funcionalidade no framework, precisamos alterar a classe **base** nos seguintes pontos:

1. Criação do método para determinar o filtro fixo;
2. Alteração do método *listar()* para que o filtro fixo seja considerado no momento da busca dos registros;
3. Alteração do método *imprimirRelatorio()* para que o filtro fixo seja enviado para a classe **relatorio**.

É necessário ainda alterar o método *imprimir()* da classe **relatorio** para que o filtro fixo seja considerado no momento da geração dos registros para impressão.

base

```
/**
 * Define um Filtro Fixo para a Busca
 *
 * @return mixed
 */
protected function FiltroFixo() {
     return NULL;
```

```
}

/**
 * Exibe uma lista de registros da tabela relacionada
 *
 * @return string
 */
public function listar() {
    ...
    if(($_filtroSQL=$this->FiltroFixo())!==null) {
        if($_filtro!==null) {
            $_filtroSQL .= " AND ({$_filtro})";
        }
    } else {
        $_filtroSQL = $_filtro;
    }
    if(!isset($_GET['pagina'])) {
        $_strSQL = $this->montaSELECT("COUNT(*) AS TOTAL",$_filtroSQL);
        $_total_registros = 0;
        if($this->_conn->executaSQL($_strSQL)!==false) {
            $_dados = $this->_conn->proximo();
            $_total_registros = $_dados['TOTAL'];
        }
    } else {
        $_total_registros = $_GET['total_registros'];
    }
    $_limit = $this->_conn->getLimit(paginacao::getLimit(
                    isset($_GET['pagina'])
                            ? $_GET['pagina']
                            : 1));
    $_strSQL = $this->montaSELECT(implode(",",
                            array_keys($_camposlista)),
                            $_filtroSQL,
                            $this->getOrdem(),
                            $_limit);
    ...
}

/**
 * Imprime o Relatório
 *
 */
public function imprimirRelatorio() {
    $_rel = new relatorio($this);
    echo $_rel->Imprimir($this->FiltroFixo());
}
```

relatorio

```
/**
 * Imprime o relatório da classe, em geral um cadastro
 *
 * @param mixed $_filtroSQL
 * @return string
 */
public function Imprimir($_filtroSQL=null) {
    $this->addCabecalho();
    $_camposlista = $_POST['campos'];
    $_camposrel   = $this->_classe->filtrarCampos("relatorio");
    foreach($_camposrel as $_chv=>$_campo) {
        if(stripos(",{$_camposlista},",",{$_chv},")===false) {
            unset($_camposrel[$_chv]);
        }
```

```
        }
        $_filtro_html = null;
        if(($_filtro=$this->_classe->getFiltro())!==null&&
           $_POST['filtro']=='S') {
             $_filtro_html = utf8_decode(implode(", ",$_filtro["HTML"]));
             $_filtro = implode(" AND ",$_filtro["SQL"]);
        } else {
             $_filtro = null;
        }
        if($_filtroSQL!==null) {
             if($_filtro!==null) {
                  $_filtroSQL .= " AND ({$_filtro})";
             }
        } else {
             $_filtroSQL = $_filtro;
        }
        $_ordem = ($_POST['ordem']!=""
                           ? $_POST['ordem'] . ","
                           : "") .
                  implode(",",array_keys($this->_classe->filtrarCampos("pk")));
        $_strSQL = $this->_classe->montaSELECT($_camposlista,
                           $_filtroSQL,$_ordem,null,null,"relatorio");
        ...
    }
```

A definição do filtro fixo deve ser feita na classe em que ele for necessário, por meio da reescrita do método *filtroFixo()*. Acompanhe um exemplo:

```
/**
 * Classe para movimentação de entradas
 *
 */
class entradas extends historicomovimento {
    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->getCampo("HISTORICO_VALOR_UNIT")->setMinimo(0.01);
        $this->getCampo("HISTORICO_VALOR_TOTAL")->setMinimo(0.01);
    }

    /**
     * Seleciona apenas os tipos de movimento de entrada
     *
     * @param string $_where
     */
    protected function buscaTiposMovimentos($_where=null) {
        parent::buscaTiposMovimentos("TIPOMOV_TIPO='E' AND TIPOMOV_ESTORNO='N'");
    }

    protected function FiltroFixo() {
        return "TIPOMOV_TIPO='E'";
    }
    ...
}
```

Geração personalizada de opções na lista de registros

O framework disponibiliza na página de listagem dos registros, que em geral é a página inicial dos programas, seja de cadastro ou de outros processos, as opções-padrão para alteração e exclusão de um determinado registro e ainda a opção de inclusão de um

novo registro. Às vezes é preciso inibir a exibição dessas opções ou de parte delas (exibir as opções de alteração e exclusão, mas não exibir a de incluir). Também é bastante comum implementar outras opções para uma determinada classe. Vamos implementar essas opções no framework, assim teremos as possíveis situações:

1. Opções de alterar e excluir inibidas
2. Opção de inclusão de novo registro inibida
3. Personalização das opções disponíveis
4. Personalização da opção de inclusão

O framework implementa somente a opção de inibir todas as opções da tela (inclusão, alteração e exclusão). Vamos alterar o framework para que haja maior maleabilidade no processo. Para atender aos itens 1 e 2, precisamos criar um atributo na classe **base**, o qual determina se a opção de inclusão de um novo registro será mostrada ou não.

Já para implementar os itens 3 e 4 precisamos desenvolver dois métodos na classe **base** e alterar o método *listar()*.

Teremos um método para a exibição das opções relacionadas a cada registro listado (alterar, excluir e demais opções) e um método que disponibiliza a personalização da opção de inclusão de um novo registro.

```
/**
 * Classe Base do framework, deverá ser estendida nas classes filhas
 *
 */
class BASE {
    protected $_nome_tabela;
    protected $_campos = Array();
    protected $_msgextra = "";
    /**
     * @var BancoDados
     */
    protected $_conn = null;
    protected $_class_path = "framework";
    protected $_exibe_opcoes = true;
    protected $_num_colunas_form = 1; // Padrão é 1 coluna
    protected $_exibe_botao_incluir = false;
    ...
    /**
     * Gera as opções na lista de registros
     * Na forma padrão, gera os links para Alterar e Excluir
     *
     * @param string $_pk
     * @param tag $_tab
     */
    protected function getOpcoesLista($_pk,tag & $_tab,tipospadrao $_tipos) {
        if($this->_exibe_opcoes===true) {
            $_lnkalt = new tag($_tipos->getTipo("A"),
                        Array(new atributo("HREF","javascript:void(0);"),
                            new atributo("ONCLICK",
                                "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?" .
                                "ACAO=ALT{$_pk}','CORPO');")),
```

```
                    " Alterar ");
          $_lnkexc = new tag($_tipos->getTipo("A"),
                    Array(new atributo("HREF","javascript:void(0);"),
                       new atributo("ONCLICK",
                             "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?" .
                             "ACAO=EXC{$_pk}','CORPO');")),
                    " Excluir ");
          $_tab->getLastSubTag()->addSubTag(new tag($_tipos->getTipo('TD'),
                    Array(new atributo("STYLE","border:1px solid #a0a0a0;")),
                    $_lnkalt->toHTML() . $_lnkexc->toHTML()));
     }
}

/**
 * Insere o botão de inclusão na lista de registros
 *
 * @param tag $_div
 * @param tipospadrao $_tipos
 */
protected function getBotaoIncluir(tag & $_div,tipospadrao $_tipos) {
     if($this->_exibe_opcoes===true||$this->_exibe_botao_incluir===true) {
          $_div->addSubTag(new tag($_tipos->getTipo("BR")));
          $_div->addSubTag(new tag($_tipos->getTipo("P"),
                    Array(new atributo("ALIGN","CENTER"))));
          $_div->getLastSubTag()->addSubTag(new tag(
                    $_tipos->getTipo('BUTTON'),
                    Array(new atributo("ONCLICK",
                         "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?" .
                         "ACAO=INC','CORPO');")),
                    " Incluir Novo Registro "));
     }
}

/**
 * Exibe uma lista de registros da tabela relacionada
 *
 * @return string
 */
public function listar() {
     ...
     foreach($_camposlista as $_campo) {
          $_tr->addSubTag(new tag($_tipos->getTipo('TH'),
               Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
               $_campo->getTitulo()));
     }
     $_numcols = sizeof($_camposlista);
     if($this->_exibe_opcoes===true) {
          $_tr->addSubTag(new tag($_tipos->getTipo('TH'),
               Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
               "Opções"));
          ++$_numcols;
     }
     $_tab->addSubTag($_tr);
     if($this->_conn->executaSQL($_strSQL)!==false&&
        $this->_conn->getNumRows()>0) {
          while($this->proximo()) {
               $_tab->addSubtag(new tag($_tipos->getTipo('TR'),
                    Array(new atributo("ONMOUSEOVER",
                                  "this.className='detalhe';"),
                         new atributo("ONMOUSEOUT","this.className='';"))));
               $_pk = "";
               foreach($_camposlista as $_campo) {
```

```
                    if($_campo->getPK()===true) {
                        $_pk .= "&{$_campo->getNome()}={$_campo->toHTML()}";
                    }
                    $_tab->getLastSubTag()->addSubTag(new tag(
                              $_tipos->getTipo('TD'),
                         Array(new atributo("STYLE",
                                   "border:1px solid #a0a0a0;"),
                              $_campo->getAtributosExtras()),
                         $_campo->toHTML()));
               }
               $this->getOpcoesLista($_pk,$_tab,$_tipos);
          }
          $_trf = new tag($_tipos->getTipo('TR'));
          $_trf->addSubTag(new tag($_tipos->getTipo("TD"),
                    Array(new atributo("COLSPAN",$_numcols),
                           new atributo("STYLE","color:#a0a0a0;" .
                                        "font-size:11px;" .
                                        "border:1px solid #a0a0a0;")),
                    "Total de {$_total_registros} Registro(s)"));
          $_tab->addSubTag($_trf);
     } else {
          $_trf = new tag($_tipos->getTipo('TR'));
          $_trf->addSubTag(new tag($_tipos->getTipo("TD"),
               Array(new atributo("COLSPAN",$_numcols),
                 new atributo("STYLE","border:1px solid #a0a0a0;" .
                                      "text-align:center")),
               "Nenhum Registro"));
          $_tab->addSubTag($_trf);

     }
     $_html = "";
     $_div = new tag($_tipos->getTipo('DIV'));
     $_div->addSubTag($_tab);
     if($_filtro_html!==null) {
          $_div->addSubTag(new tag($_tipos->getTipo("DIV"),
               Array(new atributo("STYLE","padding:10px;color:#c0c0c0;" .
                                          "font-size:12px;")),
               "Filtro Aplicado: {$_filtro_html}"));
     }
     $_pag = new tag($_tipos->getTipo("P"),
                    Array(new atributo("ALIGN","CENTER")));
     $_pag->setValor(implode(" ",paginacao::paginar($_total_registros,
                             (isset($_GET['pagina'])
                                      ? $_GET['pagina']
                                      : 1),
                             null,true)));
     $_div->addSubTag($_pag);
     $this->getBotaoIncluir($_div,$_tipos);
     $_html .= $_div->toHTML();
     return utf8_encode($_html);
}
```

A Figura 1.16 mostra um exemplo de personalização das opções disponibilizadas.

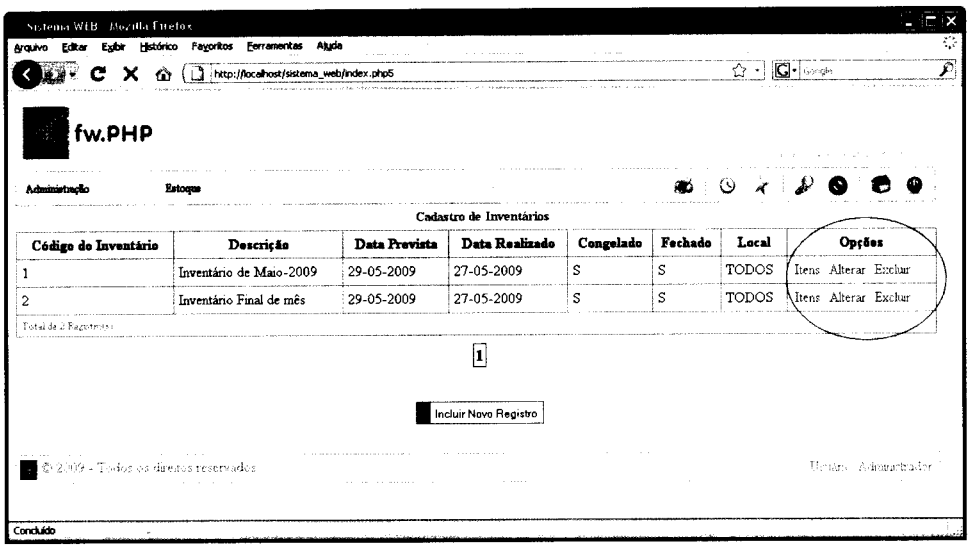


Figura 1.16

Geração de relatórios

O framework disponibiliza uma classe para a geração de relatórios, principalmente para as páginas de manutenção de cadastro, porém um sistema sempre exige relatórios de apoio, sejam operacionais ou gerenciais, os quais não são plenamente suportados pelo framework.

Para implementar o suporte a relatórios especiais, devemos alterar os seguintes pontos do framework:

1. Implementar na classe **base** métodos para suporte a relatórios especiais:

   - geraFormularioFiltroRelatorio

     Método que exibe o formulário para seleção dos filtros do relatório. Ele monta o formulário e envia o comando adequado para execução do relatório, ou seja, no atributo ONSUBMIT do relatório ele envia objRelatEspecial.runRelatorio.

   - imprimirRelatorioEspecial

     Esse método é chamado pelo relatório especial, sendo responsável por instanciar a classe **relatorio** e executar o método *imprimirEspecial* dessa classe.

- processaQuebra

  Utilizado pela classe do relatório para o processamento dos registros que estão sendo impressos. Por meio desse método a classe do relatório pode controlar a exibição de totais e outros controles úteis. A classe **relatorio** executa esse método antes de imprimir os registros selecionados.

- posRelatorio

  A classe **relatorio** executa esse método no final da impressão, quando todos os registros selecionados tiverem sido impressos e antes de imprimir o rodapé do relatório.

2. Criar o método *imprimirespecial* na classe **relatorio**:

Esse método executa o processamento do relatório especial. É um pouco diferente do método de impressão de cadastro, uma vez que implementa rotinas para controle de quebras do relatório, além de permitir que a classe do relatório especial defina o método de busca e os filtros que serão aplicados.

3. Criar o método runRelatorio na classe javascript relatorio:

O método javascript runRelatorio da classe relatorio abre uma nova janela no browser e executa através de uma chamada AJAX o método *imprimirRelatorioEspecial* da classe do relatório que está sendo executado.

A seguir temos as alterações nas classes:

base

```
/**
 * Gera o filtro para Relatórios especiais
 * Que não fazem parte de um cadastro
 *
 * @return string
 */
public function geraFormularioFiltroRelatorio() {
     $this->preFiltro();
     $_filtro = $this->getFiltro();
     $_form = new formulario("FILTRO_{$this->_nome_tabela}",
                                "Javascript:void(0);");
     $_form->deleteAtributo(4);
     $_form->addAtributo(new atributo("ONSUBMIT",
                           "objRelatEspecial.runRelatorio(" .
                           "retornaFiltro(this,false));return false;"));
     $_tab = $this->getTabelaFiltro();
     $_tab->getLastSubTag()->addSubTag(new tag(new tipotag("TD"),
          Array(new atributo("COLSPAN",4),
                new atributo('STYLE','text-align:right;padding-right:20px;'),
                new atributo("ID","BTNOK"))));
     $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
          new formInputSubmit("OK"," Gerar relatório "));
     $_form->addSubTag($_tab);
     echo utf8_encode($this->getInfoPagina()->toHTML() . $_form->toHTML());
}
```

```
/**
 * Gera a impressão de um relatório especial
 *
 */
public function imprimirRelatorioEspecial() {
     $_rel = new relatorio($this);
     echo $_rel->imprimirEspecial();
}

/**
 * Gerenciamento do relatório
 * Processamento de quebras
 * Totais e outros
 *
 * @return mixed   null ou instancia da classe tag
 */
public function processaQuebra($_fim=false) {
     return null;
}


/**
 * Processamento final do relatório
 *
 */
public function posRelatorio() {
     return null;
}
```

relatorio

```
/**
 * Método para impressões especiais
 * utilizado para gerar relatórios no sistema
 *
 * @return string
 */
public function imprimirEspecial() {
     $this->addCabecalho();
     $_camposrel    = $this->_classe->filtrarCampos("relatorio");
     $_strSQL = $this->_classe->montaSELECT($_camposrel);
     $_tab = new tag($this->_tipos->getTipo('TABLE'),
               Array(new atributo("BORDER",0),
                    new atributo("CELLPADDING",5),
                    new atributo("CELLSPACING",3),
                    new atributo("STYLE","border-collapse:collapse;"),
                    new atributo("WIDTH","100%")));
     $_tr  = new tag($this->_tipos->getTipo('TR'));
     foreach($_camposrel as $_campo) {
          $_tr->addSubTag(new tag($this->_tipos->getTipo('TH'),
               Array(new atributo("STYLE",
                                  "border-bottom:2px solid #a0a0a0;")),
               $_campo->getTitulo()));
     }
     $_tab->addSubTag($_tr);
     if($this->_classe->getConn()->executaSQL($_strSQL)!==false&&
        $this->_classe->getConn()->getNumRows()>0) {
          while($this->_classe->proximo()) {
               if(($_quebra=$this->_classe->processaQuebra())!==null) {
                    if(is_array($_quebra)) {
                         foreach($_quebra as $_linha) {
                              $_tab->addSubtag($_linha);
```

```
                    }
                } else {
                    $_tab->addSubtag($_quebra);
                }
            }
            $this->_classe->processaRelatorio();
            $_tab->addSubtag(new tag($this->_tipos->getTipo('TR')));
            foreach($_camposrel as $_campo) {
                $_tab->getLastSubTag()->addSubTag(
                    new tag($this->_tipos->getTipo('TD'),
                        Array($_campo->getAtributosExtras()),
                        $_campo->toPrint()));
            }
        }
        // Ultima Quebra
        if(($_quebra=$this->_classe->processaQuebra(true))!==null) {
            if(is_array($_quebra)) {
                foreach($_quebra as $_linha) {
                    $_tab->addSubtag($_linha);
                }
            } else {
                $_tab->addSubtag($_quebra);
            }
        }
        // processamento do final do relatório, totais por exemplo
        if(($_footer=$this->_classe->posRelatorio())!==null) {
            foreach($_footer as $_res) {
                $_tab->addSubtag(new tag($this->_tipos->getTipo('TR')));
                foreach($_res[0] as $_k=>$_campo) {
                    $_att = Array($_campo->getAtributosExtras());
                    if($_res[1][$_k]!==null) {
                        $_att[1] = $_res[1][$_k];
                    }
                    $_tab->getLastSubTag()->addSubTag(
                        new tag($this->_tipos->getTipo('TD'),
                            $_att,$_campo->toPrint()));
                }
            }
        }
        $_trf = new tag($this->_tipos->getTipo('TR'));
        $_trf->addSubTag(new tag($this->_tipos->getTipo("TD"),
            Array(new atributo("COLSPAN",sizeof($_camposrel)),
                new atributo("STYLE","color:#a0a0a0;font-size:11px;" .
                                     "border-top:1px solid #a0a0a0;")),
            "Total de {$this->_classe->getConn()->getNumRows()} " .
            "Registro(s)"));
        $_tab->addSubTag($_trf);
    } else {
        $_trf = new tag($this->_tipos->getTipo('TR'));
        $_trf->addSubTag(new tag($this->_tipos->getTipo("TD"),
            Array(new atributo("COLSPAN",sizeof($_camposrel)),
                new atributo("STYLE","text-align:center")),
            " Nenhum Registro "));
        $_tab->addSubTag($_trf);
    }
    $this->getLastSubTag()->addSubTag($_tab);
    $this->addbotaoimpressao();
    return utf8_encode($this->toHTML(false));
}
```

javascript relatorio

```
relatorio.prototype.runRelatorio=
     function(params) {
          var ax=new AJAX();
          ax.url='executa_busca_ajax.php5';
          ax.metodo='POST';
          ax.params='metodo=ImprimirRelatorioEspecial'+params+
               '&titulo='+document.getElementById('TITULO').innerHTML+
               '&'+ObjInfoPagina.retornaParametros();
          ax.obj = this;
          ax.processaresultado=
               function(html) {
                    var w = window.open('','relatorioespecial',
                                   'height=600,width=1010,left=100,top=100,' +
                                   'resizable=yes,scrollbars=yes,' +
                                   'toolbar=no,status=no');
                    w.document.write(html);
                    w.document.close();
                    if(window.focus) {
                         w.focus();
                    }
                    ax.obj.objme.reset();
               };
          ax.conectar();
     };
```

A Figura 1.17 mostra o exemplo de um relatório construído com essa nova ferramenta do framework.

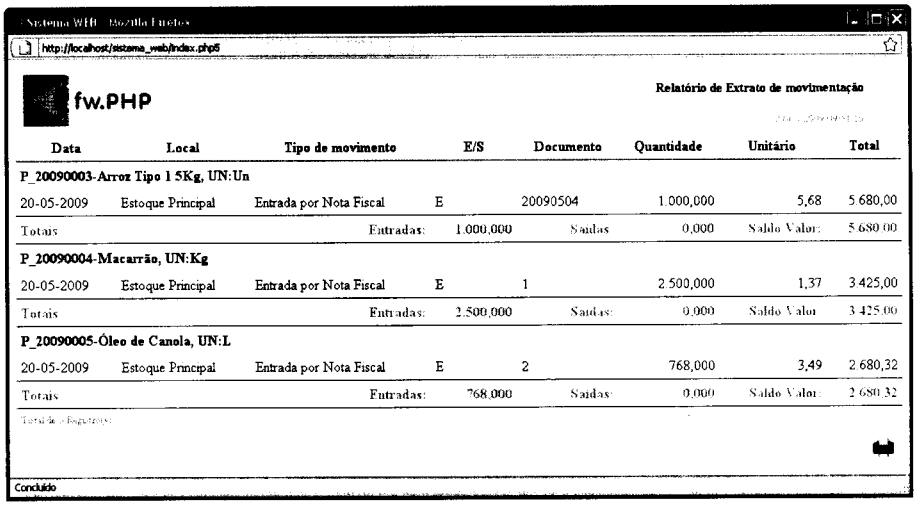


Figura 1.17

Mostrar e esconder botões

Como condição-padrão o framework exibe em todas as páginas um grupo predefinido de botões, não importa a página que está sendo executada (na verdade, os

botões são montados apenas no momento em que a autenticação do usuário é feita), porém há situações em que precisamos esconder alguns desses botões. Tome como exemplo o módulo de relatórios, em que não é necessária a exibição dos botões de filtro nem mesmo de impressão do relatório-padrão.

Para implementar esta facilidade, devemos criar na classe **base** dois métodos, um para esconder os botões e outro para mostrá-los novamente. É preciso, ainda, a criação de um método javascript que execute essas ações.

base

```
/**
 * Exibe os botões que podem ter sido escondidos
 *
 * @return tag
 */
protected function habilitaBotoes() {
     $_comandos  = "mostraescondebotao(['' Relatório ',"" .
                    "' Filtro de registros'],false);";
     return new tag(new tipotag("SCRIPT"),null,$_comandos);
}

/**
 * Esconde botões inuteis para o módulo relatório
 *
 * @return tag
 */
protected function inibeBotoes() {
     $_comandos  = "mostraescondebotao(['' Relatório ',"" .
                    "' Filtro de registros'],true);";
     return new tag(new tipotag("SCRIPT"),null,$_comandos);
}
```

javascript

```
function mostraescondebotao(b,f) {
     var img = document.images;
     for(var i=0;i<img.length;i++) {
          if(b.indexOf(img[i].src)!=-1||b.indexOf(img[i].name)!=-1) {
               img[i].parentNode.style.display = (f===true ? 'none' : 'inline');
          }
     }
};
```

Para utilizar a nova funcionalidade precisamos alterar dois métodos da classe **base**. O método *processaAcao()* deve sempre enviar o comando para exibir os botões. Já o método *geraFormularioFiltroRelatorio()* deve enviar o comando para esconder os botões.

```
public function processaAcao() {
     ...
     echo $_html . $this->listar() .
          utf8_encode($_sct->toHTML() .
          $this->habilitaBotoes()->toHTML());
}

public function geraFormularioFiltroRelatorio() {
```

```
        ...
        echo utf8_encode(
                $this->getInfoPagina()->toHTML() .
                $this->inibeBotoes()->toHTML() .
                $_form->toHTML()
            );
}
```

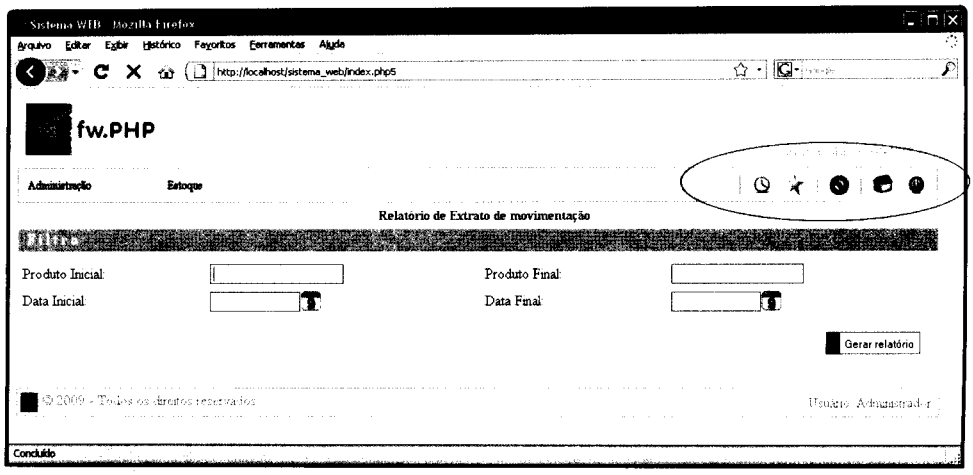


Figura 1.18

Exibição de mensagem personalizada

O framework fw.PHP tem por padrão a exibição de mensagens de alerta por meio do comando *alert()* do javascript. Na maioria dos casos esse procedimento é suficiente, mas em alguns momentos precisamos de um processo mais sofisticado de mensagens, como uma tela personalizável, na qual possamos inserir textos, comandos HTML, botões e comandos javascript. É importante também que a caixa de mensagens apareça como modal, não permitindo o acesso à página antes que o usuário opte por uma das ações disponíveis na caixa de mensagens.

Para implementar a caixa de mensagens, vamos utilizar um pequeno grupo de funções javascripts criado por **Michael Leigeber** (http://www.leigeber.com). A utilização é bem simples e o resultado é excelente. Para exibir a caixa de mensagens precisamos apenas executar a função *showDialog()*, enviando o título da caixa de mensagens, seu conteúdo e o tipo, que pode ser:

- **Error:** mensagem de erro
- **Warning:** mensagem de aviso
- **Success:** mensagem de sucesso
- **Prompt:** mensagem com algum questionamento

Cada tipo de mensagem tem um grafismo próprio, de forma que o usuário identifique rapidamente o objetivo da caixa de mensagens. A Figura 1.19 mostra um exemplo do tipo Success.

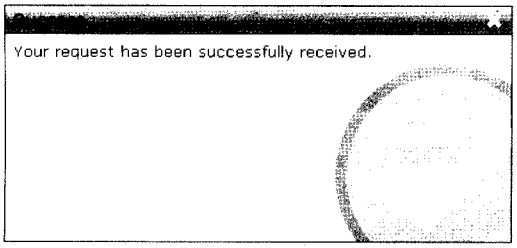


Figura 1.19

A Figura 1.20 ilustra uma caixa de mensagens mais sofisticada, com textos e botões para que o usuário selecione a opção mais indicada.

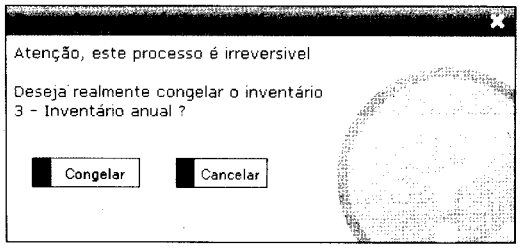


Figura 1.20

Capítulo 2

Definição do Sistema

O objetivo de um sistema de controle de estoque é prover meios para que os tomadores de decisão da empresa tenham condições de avaliar o giro de mercadorias, sua situação atual, sua movimentação passada e várias outras informações que possibilitam uma visão clara dos estoques e permitem decidir que providências devem ser tomadas, seja a reposição do estoque dos produtos, programação ou reprogramação de compras, estabelecimento de promoções para aumentar o giro, retirada de produtos do mix da empresa e outras atitudes compatíveis com a situação atual.

Logo, é importante ter os dados atualizados e confiáveis, pois uma boa decisão baseada em dados equivocados geralmente se torna uma decisão errada.

Neste capítulo definimos a estrutura do sistema de controle de estoque, as entidades relacionadas e os processos que serão disponibilizados aos usuários. O controle de acesso, as permissões, a auditoria e os menus já estão definidos no framework, não sendo necessária a implementação neste livro (o capítulo um mostra como instalar, configurar e utilizar o framework). Veremos apenas como utilizá-los para o correto desenvolvimento do sistema.

Definição dos módulos

Um sistema de controle de estoque necessita de módulos para cadastramento, movimentação de produtos (entradas e saídas), inventário de estoque e um módulo para relatórios de gerenciamento.

O módulo de cadastros fornece meios para que o usuário defina os produtos existentes no sistema, bem como o cadastro de tipos de movimento para que seja possível gerar um histórico de movimentação adequado. O cadastro de produtos exige, ainda, alguns cadastros auxiliares para ajudar na organização dos produtos. Um dos cadastros é o de departamentos o qual permite a organização dos produtos por locais.

O módulo de movimentação é o lugar onde os usuários informam as entradas e saídas dos produtos, criando desta forma o histórico, além de fazer com que os produtos virtuais estejam sempre em sintonia com os produtos físicos no quesito estoque atual.

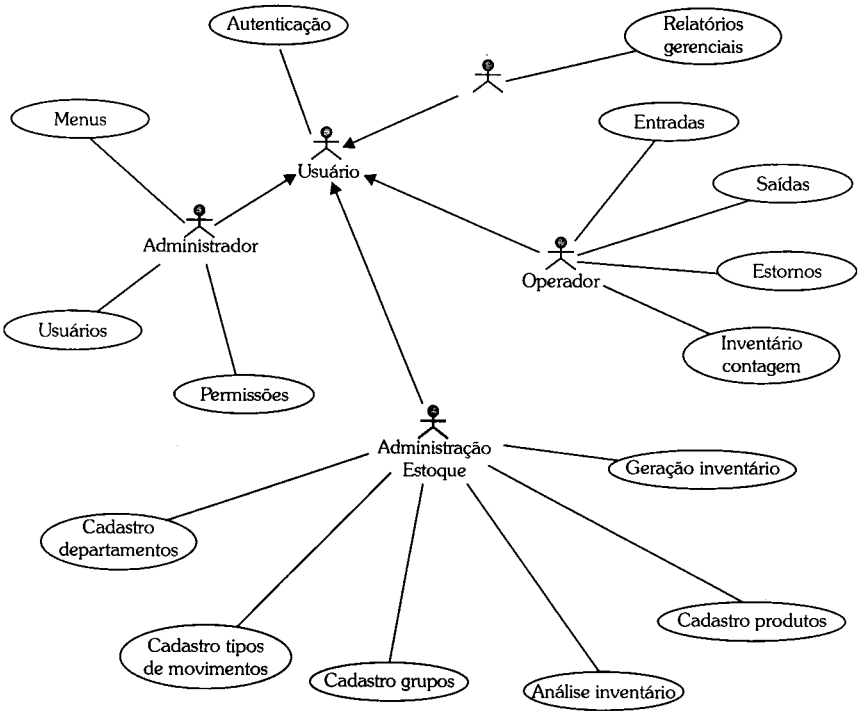


Figura 2.1

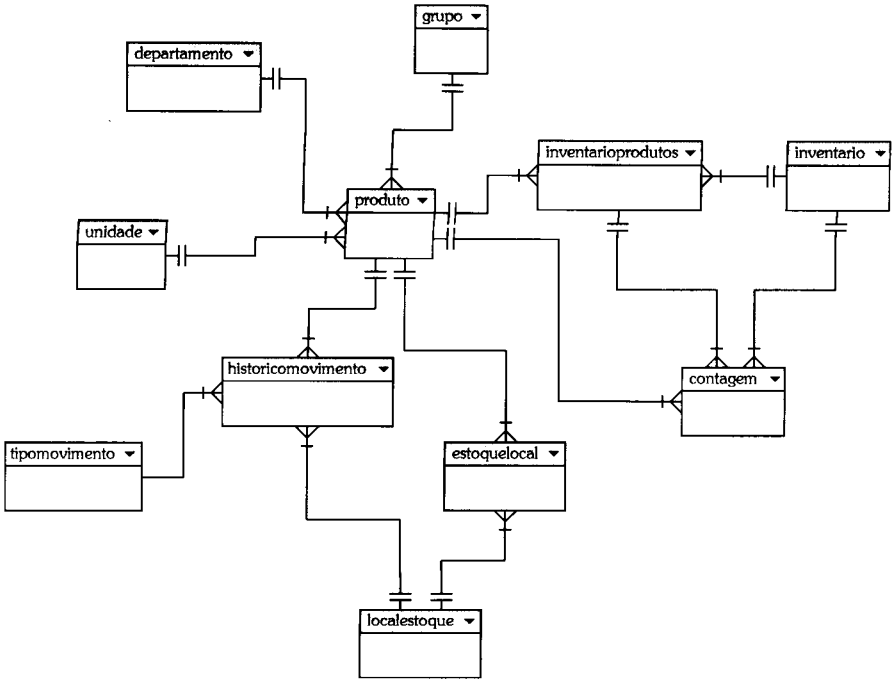


Figura 2.2

O módulo de inventário disponibiliza as rotinas necessárias para a geração do inventário, bem como sua apuração e ajuste do estoque.

O último módulo apresenta relatórios operacionais e gerenciais para melhor controle e entendimento dos estoques existentes.

Cadastros

Qualquer sistema, por mais simples que seja, necessita de uma ou mais tabelas de cadastro. No sistema que vamos desenvolver neste livro precisamos de várias tabelas de cadastro para que ele funcione corretamente. Os cadastrados são responsáveis pela organização e padronização do sistema, permitindo a correta estruturação dos dados.

Precisamos criar as seguintes entidades no sistema de controle de estoque.

- Departamento
- Grupo
- Unidade
- Produto
- Local de estoque
- Tipo de movimento

O cadastro de departamentos define os setores, sessões e departamentos existentes na empresa. Os produtos são distribuídos pelos departamentos, permitindo a correta organização e distribuição.

O cadastro de grupos é utilizado para que os produtos sejam separados em organizações diferentes das determinadas pelos departamentos, levando uma nova visão aos gestores (por exemplo, podemos colocar refrigerante, cerveja e carne no grupo churrasco, enquanto fisicamente esses produtos estão em departamentos diferentes).

O cadastro de unidades permite ao usuário definir as unidades de medida que estarão disponíveis no sistema (Quilo, Litro, Caixa etc.).

É no cadastro de produtos que definimos a estrutura fundamental do sistema de controle de estoques, pois tudo gira em torno desta entidade. Nesse cadastro definimos os atributos fundamentais para a gestão dos saldos de estoque e das demais informações necessárias para a correta tomada de decisão.

O cadastro de locais de estoque permite a separação do estoque dos produtos em diversos locais físicos. Podemos ter, por exemplo, o estoque central, o estoque da loja e o estoque da gôndola.

Já o cadastro de tipos de movimento permite a separação dos lançamentos no histórico de movimentação, trazendo melhor organização a todo o processo.

Requisitos

No processo de cadastramento das entidades necessárias ao sistema de controle de estoque, os seguintes requisitos estão definidos:

1. Um departamento não pode estar relacionado a ele mesmo.
2. Deve existir pelo menos um departamento cadastrado para que seja possível cadastrar um produto.
3. Deve existir pelo menos um grupo cadastrado para o cadastramento de um produto.
4. Deve existir pelo menos uma unidade cadastrada para que seja possível cadastrar um produto.

Departamento

A entidade departamento contém as informações necessárias para definir departamentos, sessões, setores e qualquer outra organização física ou lógica dos produtos. Precisamos de poucos atributos para definir esta entidade.

- Código
- Descrição
- Departamento relacionado
- Tipo do departamento
- Usuário
- Data alteração

O atributo departamento relacionado define a estrutura em árvore da entidade, ou seja, é possível ter múltiplos níveis de departamentos (por exemplo, Bebida > refrigerante > Cola), quantos forem necessários para definir de forma correta a estrutura física adotada pela empresa.

O atributo tipo de departamento define se o departamento é um almoxarifado, um armazém interno ou um estoque para vendas. Com isso, em outros sistemas, como, por exemplo, o sistema de vendas, é possível limitar a busca de produtos somente aos departamentos relacionados a essa atividade.

Os atributos Usuário e Data Alteração definem quem fez a última alteração e quando essa operação foi realizada.

Atributo	Tipo	Tamanho	Nulo	Chave
depto_codigo	Caractere	10	Não	Primária
depto_desc	Caractere	60	Não	-
depto_pai	Caractere	10	Sim	-
depto_tipo	Inteiro	-	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

O script de criação da tabela no postgreSQL está descrito a seguir.

```
CREATE TABLE departamento(
  depto_codigo character varying(10) NOT NULL,
  depto_desc character varying(60) NOT NULL,
  depto_pai character varying(10) DEFAULT ''::character varying,
  depto_tipo integer DEFAULT 1,
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT departamento_pkey PRIMARY KEY (depto_codigo),
  CONSTRAINT usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Grupo

A entidade Grupo é ainda mais simples que a entidade departamento. É preciso apenas definir um código e sua descrição. Esta entidade será utilizada basicamente para obtermos uma classificação secundária dos produtos.

Atributo	Tipo	Tamanho	Nulo	Chave
grupo_codigo	Caractere	10	Não	Primária
grupo_desc	Caractere	60	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

O comando SQL para criação da tabela é apresentado em seguida:

```
CREATE TABLE grupo(
  grupo_codigo character varying(10) NOT NULL,
  grupo_desc character varying(60) NOT NULL,
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT grupo_pkey PRIMARY KEY (grupo_codigo),
  CONSTRAINT usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Unidade

O cadastro de unidade é extremamente simples, pois contém apenas o símbolo da unidade, que por padrão tem até quatro caracteres, e a descrição da unidade de medida.

Atributo	Tipo	Tamanho	Nulo	Chave
unidade_codigo	Caractere	4	Não	Primária
unidade_desc	Caractere	20	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

Veja o comando SQL para criação da tabela no postgreSQL.

```
CREATE TABLE unidade (
  unidade_codigo character(4) NOT NULL,
  unidade_desc character varying(20) NOT NULL,
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT unidade_pkey PRIMARY KEY (unidade_codigo),
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Produto

A principal entidade descreve os atributos dos produtos que serão gerenciados no controle de estoque. Precisamos escolher atributos que definam corretamente o código do produto, sua descrição, preço, unidade de medida, departamento ao qual pertence e vários outros atributos que ajudam a especificar corretamente os produtos.

- Código
- Descrição
- Preço atual
- Custo atual
- Custo médio
- Departamento
- Grupo
- Unidade
- Peso por unidade
- Estoque atual
- Estoque mínimo
- Estoque máximo
- Acumulado de entradas quantidade
- Acumulado de entradas valor
- Acumulado de saídas

Os atributos custo atual e custo médio são gerados na entrada de estoque. O custo atual reflete o custo da última entrada e o custo médio é a divisão do valor total de entradas pela quantidade acumulada de entradas.

Departamento e grupo vêm das respectivas tabelas, e servem para que o produto possua duas classificações, uma relacionada ao mundo físico e a segunda para uma visualização diferente dos produtos.

O atributo unidade define a unidade de medida do produto, por exemplo, quilo, litro, caixa etc.

Já peso por unidade serve para termos uma medida comum a todos os produtos.

Os atributos estoque mínimo e máximo são utilizados para cálculos do estoque, possibilitando mecanismos de controle para evitar um estoque muito baixo ou muito alto.

Atributo	Tipo	Tamanho	Nulo	Chave
produto_codigo	Caractere	20	Não	Primária
produto_desc	Caractere	60	Não	-
produto_preco	Numérico	12,3	Não	-
produto_custoatual	Numérico	12,3	Não	-
produto_customedio	Numérico	12,3	Não	-
depto_codigo	Caractere	10	Não	Estrangeira
grupo_codigo	Caractere	10	Não	Estrangeira
unidade_codigo	Caractere	4	Não	Estrangeira
produto_peso	Numérico	10,5	-	-
produto_estoque	Numérico	10,3	Não	-
produto_est_minimo	Numérico	10,3	Não	-
produto_est_maximo	Numérico	10,3	Não	-
produto_ac_ent_qde	Numérico	18,3	Não	-
produto_ac_ent_vlr	Numérico	18,3	Não	-
produto_ac_saidas	Numérico	18,3	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

Agora que temos todos os atributos da tabela de produtos declarados podemos gerar o comando SQL para a criação da tabela no banco de dados.

```
CREATE TABLE produto (
  produto_codigo character varying(20) NOT NULL,
  produto_desc character varying(60) NOT NULL,
  produto_preco numeric(12,3),
  produto_custoatual numeric(12,3),
  produto_customedio numeric(12,3),
  depto_codigo character varying(10) NOT NULL,
  grupo_codigo character varying(10) NOT NULL,
  unidade_codigo character(4) NOT NULL,
  produto_peso numeric(10,5),
  produto_estoque numeric(10,3),
  produto_est_minimo numeric(10,3),
```

```
  produto_est_maximo numeric(10,3),
  produto_ac_ent_qde numeric(18,3),
  produto_ac_ent_vlr numeric(18,3),
  produto_ac_saidas numeric(18,3),
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT produto_pkey PRIMARY KEY (produto_codigo),
  CONSTRAINT fk_depto FOREIGN KEY (depto_codigo)
      REFERENCES departamento (depto_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_grupo FOREIGN KEY (grupo_codigo)
      REFERENCES grupo (grupo_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_unidade FOREIGN KEY (unidade_codigo)
      REFERENCES unidade (unidade_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Local de estoque

A entidade que descrevemos agora contém as informações sobre os locais de estocagem disponíveis na empresa. Os locais podem ser lojas, centros de estocagem, gôndolas e outros que sejam necessários para a correta mensuração dos estoques. Os atributos necessários para descrever esta entidade estão listados a seguir.

- Código do local
- Descrição do local
- Tipo do local (loja, gôndola, centro de estocagem)

Para que o sistema funcione corretamente é necessário que pelo menos um local de estoque esteja cadastrado.

O tipo de local é apenas informativo e pode ser utilizado nos relatórios gerenciais.

Atributo	Tipo	Tamanho	Nulo	Chave
local_codigo	Caractere	10	Não	Primária
local_desc	Caractere	30	Não	-
local_tipo	Char	1	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

O comando SQL para criação da tabela de locais de estoque está descrito em seguida:

```
CREATE TABLE localestoque (
  local_codigo character varying(10) NOT NULL,
  local_desc character varying(30) NOT NULL,
  local_tipo character(1) DEFAULT 'L'::bpchar,
```

```
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT localestoque_pkey PRIMARY KEY (local_codigo),
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Tipo de movimento

A tabela de tipos de movimentos é utilizada nos lançamentos de movimentação dos produtos, tanto de entradas quanto de saídas, auxiliando o usuário a classificar corretamente os lançamentos realizados. A tabela precisa de poucos atributos, somente o necessário para identificar e classificar os tipos de movimento.

- Código do tipo de movimento
- Descrição
- Entrada ou saída
- Estorno
- Afeta o custo médio
- Afeta o custo atual

O atributo "Entrada ou saída" define se o tipo de movimento é utilizado nas operações de entradas de mercadoria ou nas operações de saída.

Os atributos "Afeta o custo médio" e "Afeta o custo atual" indicam se o movimento é utilizado para o recálculo dos custos médio e atual do produto. Somente os tipos de movimento de entrada podem participar do cálculo.

O atributo Estorno define se o tipo de movimento é um estorno de movimentação, ou seja, se é um tipo de movimento para desfazer uma operação do sistema. O estorno deve ser sempre contrário ao movimento efetuado, ou seja, um estorno de saída deve ser uma entrada e o estorno de uma entrada deve ser uma saída.

Para que os cálculos dos custos médio e atual sejam executados, o sistema deve levar em conta os movimentos de entrada que não sejam um estorno e os movimentos de saída que sejam estorno de uma entrada. A tabela a seguir mostra esta relação:

	Estorno	
	Sim	**Não**
Entrada	**Não**	**Sim**
Saída	**Sim**	**Não**

Desta forma temos apenas duas situações para cálculo do custo médio e duas para o cálculo do custo atual.

Atributo	Tipo	Tamanho	Nulo	Chave
tipomov_codigo	Caractere	10	Não	Primária
tipomov_desc	Caractere	30	Não	-
tipomov_tipo	Caractere	1	Não	-
tipomov_estorno	Caractere	1	Não	-
tipomov_customedio	Caractere	1	Não	-
tipomov_custoatual	Caractere	1	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

Transformando os atributos anteriores em um comando SQL para criação da tabela, teremos o código listado em seguida:

```
CREATE TABLE tipomovimento (
  tipomov_codigo character varying(10) NOT NULL,
  tipomov_desc character varying(30) NOT NULL,
  tipomov_tipo character(1) NOT NULL DEFAULT 'E'::bpchar,
  tipomov_estorno character(1) NOT NULL DEFAULT 'N'::bpchar,
  tipomov_customedio character(1) NOT NULL DEFAULT 'S'::bpchar,
  tipomov_custoatual character(1) NOT NULL DEFAULT 'S'::bpchar,
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT tipomovimento_pkey PRIMARY KEY (tipomov_codigo),
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Movimentação

O módulo de movimentação possibilita as funcionalidades necessárias para o ajuste de estoque, sejam lançamentos de entradas ou lançamentos de saídas de estoque. O módulo é composto basicamente de três funcionalidades. Uma para que seja possível realizar as entradas de estoque, outra para as saídas e uma última funcionalidade para o estorno tanto de entradas quanto de saídas.

- Entradas de estoque
- Saídas de estoque
- Estornos de entradas e saídas

Esse módulo necessita de duas novas entidades, além das entidades já definidas no módulo de cadastro.

- Histórico de lançamentos
- Estoque por local

A primeira entidade, Histórico, contém a movimentação dos produtos, detalhando cada movimento, seja de entrada ou saída.

A outra entidade, Estoque por local, armazena os estoques dos produtos por local cadastrado, apresentando um resumo de entradas e saídas de cada local e produto.

Requisitos

Os seguintes requisitos são considerados nos lançamentos de entradas, saídas e estornos de estoque:

1. Deve existir pelo menos um produto cadastrado.
2. Deve existir pelo menos um local de estoque cadastrado.
3. Devem existir pelo menos dois tipos de lançamentos cadastrados, um de entrada e outro de saída.
4. Um produto não pode ter seu saldo de estoque menor que zero.
5. Um local não pode ter seu saldo menor que zero.
6. A entrada de estoque é considerada para cálculo dos custos médio e atual, exceto nos seguintes casos:
   - ➲ Se a entrada for um estorno de saída;
   - ➲ Se o tipo de movimento utilizado indicar o contrário.
7. A saída de estoque somente é considerada para cálculo dos custos médio e atual quando o movimento for um estorno de entrada e o tipo de movimento indicar que os custos devem ser recalculados. Nesta situação os valores considerados têm o sinal negativo tanto para quantidade quanto para valor. Desta forma os efeitos da entrada efetuada são revertidos.

Histórico de lançamentos

A tabela de histórico tem os dados necessários para a correta identificação e rastreamento dos movimentos realizados no sistema de controle de estoques. Precisamos de atributos que identifiquem o produto, o local, a data, o tipo de movimento, a quantidade movimentada, o valor unitário do movimento, o valor total da operação, o documento que originou o movimento, o histórico da movimentação, bem como se este é um movimento de estorno ou não e caso seja, qual o código do movimento associado.

- ➲ Código do produto
- ➲ Código do local

- Data do movimento
- Seqüencial do lançamento
- Tipo de movimento
- Entrada ou saída
- Quantidade movimentada
- Valor unitário
- Valor total
- Documento
- Histórico
- Estorno
- Seqüencial do estorno
- Estornado

O atributo seqüencial do lançamento é um valor único na tabela de histórico, sendo utilizado como referência em vários processos do sistema, como, por exemplo, o estorno de um lançamento no histórico.

Apesar de termos o código do tipo de movimento relacionado, vamos gravar também se o movimento é uma entrada ou uma saída para que a busca dos dados seja mais rápida.

O atributo valor unitário é informado nos lançamentos de entrada, já nos lançamentos de saída utilizamos o custo médio atual do produto. Nos lançamentos de estorno utilizamos o valor do movimento original.

O atributo estorno indica se este é um movimento de estorno ou não.

O seqüencial do estorno indica o seqüencial da operação original que está sendo estornada.

O atributo estornado indica se o movimento já foi estornado ou não. Um movimento estornado não pode ser estornado novamente.

Atributo	Tipo	Tamanho	Nulo	Chave
produto_codigo	Caractere	20	Não	Primária
local_codigo	Caractere	10	Não	Primária
historico_data	Data	10	Não	Primária
historico_sequencia	Inteiro		Não	Primária
tipomov_codigo	Caractere	10	Não	-
tipomov_tipo	Caractere	1	Não	-

Atributo	Tipo	Tamanho	Nulo	Chave
historico_documento	Caractere	20	Sim	-
historico_historico	Texto	2000	Sim	-
historico_quantidade	Numérico	10,3	Não	-
historico_valor_unit	Numérico	12,3	Não	-
historico_valor_total	Numérico	12,3	Não	-
historico_estorno	Caractere	1	Não	-
historico_estorno_seq	Inteiro		Sim	-
historico_estornado	Caractere	1	Sim	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

O comando SQL descrito em seguida exemplifica como criar a tabela historicomovimento, pois no framework já existe uma tabela com o nome historico.

```
CREATE TABLE historicomovimento (
  produto_codigo character varying(20) NOT NULL,
  local_codigo character varying(10) NOT NULL,
  historico_data date NOT NULL,
  historico_sequencia bigint NOT NULL,
  tipomov_codigo character varying(10) NOT NULL,
  tipomov_tipo character(1) NOT NULL DEFAULT 'E'::bpchar,
  historico_documento character varying(20),
  historico_historico text,
  historico_quantidade numeric(10,3),
  historico_valor_unit numeric(12,3),
  historico_valor_total numeric(12,3),
  historico_estorno character(1) DEFAULT 'N'::bpchar,
  historico_estorno_seq bigint,
  historico_estornado character(1) DEFAULT 'N'::bpchar,
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT historicomovimento_pkey
          PRIMARY KEY (produto_codigo, local_codigo,
                       historico_data, historico_sequencia),
  CONSTRAINT fk_local FOREIGN KEY (local_codigo)
      REFERENCES localestoque (local_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_produto FOREIGN KEY (produto_codigo)
      REFERENCES produto (produto_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_tipo FOREIGN KEY (tipomov_codigo)
      REFERENCES tipomovimento (tipomov_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Estoque por local

Uma vez que temos o estoque dividido em um ou mais locais de estocagem (pelo menos um local), precisamos criar uma tabela que contenha o estoque consolidado de cada produto por local de estocagem. A tabela de Estoque por local tem o código do produto, o código do local, o estoque atual do produto no local, o acumulado de entradas e saídas, o valor acumulado de entradas, o custo médio do local e o custo atual do local.

- Código do produto
- Código do local
- Estoque atual
- Custo atual
- Custo médio atual
- Acumulado de entradas quantidade
- Acumulado de saídas
- Acumulado de entradas valor

Os dados de custo atual, custo médio, acumulados de entradas e saídas são informações meramente gerenciais, não influenciando nos cálculos do sistema.

Atributo	Tipo	Tamanho	Nulo	Chave
produto_codigo	Caractere	20	Não	Primária
local_codigo	Caractere	10	Não	Primária
produto_estoque	Numérico	10,3	Não	-
produto_custoatual	Numérico	12,3	Não	-
produto_customedio	Numérico	12,3	Não	-
produto_ac_ent_qde	Numérico	18,3	Não	-
produto_ac_ent_vlr	Numérico	18,3	Não	-
produto_ac_saidas	Numérico	18,3	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

Os dados da tabela são gerados a partir da primeira movimentação do produto no local. Veja o código SQL necessário para criar a tabela no postgreSQL:

```
CREATE TABLE estoquelocal (
  produto_codigo character varying(20) NOT NULL,
  local_codigo character varying(10) NOT NULL,
  produto_estoque numeric(10,3),
  produto_custoatual numeric(12,3),
  produto_customedio numeric(12,3),
```

```
  produto_ac_ent_qde numeric(18,3),
  produto_ac_ent_vlr numeric(18,3),
  produto_ac_saidas numeric(18,3),
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT estoquelocal_pkey PRIMARY KEY (produto_codigo, local_codigo),
  CONSTRAINT fk_local FOREIGN KEY (local_codigo)
      REFERENCES localestoque (local_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_produto FOREIGN KEY (produto_codigo)
      REFERENCES produto (produto_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Inventário

O módulo de inventário é utilizado para apuração da precisão do estoque, pois é através do inventário que comparamos o estoque físico com o estoque virtual, corrigindo as diferenças existentes. O módulo de inventário deve disponibilizar ao usuário ferramentas para geração do inventário, definição dos produtos que farão parte do inventário, emissão da ficha de inventários e várias outras que completam a lista de ferramentas necessárias.

- Criação do inventário
- Geração da lista de produtos
- Congelamento do saldo de estoque
- Emissão de fichas de inventário
- Lançamento das contagens
- Comparação de contagens
- Análise do inventário

O módulo exige três novas entidades, além das entidades definidas até agora.

- Inventário
- Lista de produtos do inventário
- *Resultados das contagens*

A primeira entidade contém os dados básicos do inventário, a segunda a lista de produtos que farão parte do inventário e a terceira e última entidade tem os valores apurados nas contagens.

Requisitos

O módulo de inventário define os seguintes requisitos básicos:

1. Deve existir pelo menos um produto cadastrado.
2. Deve existir pelo menos um local de estoque cadastrado.
3. Devem existir pelo menos dois tipos de movimento cadastrados, um para entradas e outro para saídas de estoque.
4. Um produto será considerado inventariado quando:
   - A primeira contagem for igual ao estoque atual;
   - Quando duas contagens sucessivas forem iguais.
5. O inventário é considerado concluído quando não houver necessidade de uma nova contagem de nenhum produto da lista, conforme item 4.
6. As diferenças entre estoque e inventário são ajustadas conforme os tipos de movimento informados pelo usuário.
7. O congelamento dos estoques indica o início do inventário. Nesta situação as seguintes ações são tomadas pelo sistema:
   - Os estoques atuais são gravados na lista de produtos. O estoque pode ser consolidado ou por local conforme definido no cadastro do inventário.
   - Um inventário congelado não pode receber novos itens na lista de produtos.

Inventário

Nessa tabela descrevemos o inventário, informando seu código, descrição, data prevista para realização, se o inventário está congelado ou não, a contagem atual do inventário e um último atributo para definir se o inventário será consolidado ou por local de estoque.

- Código do inventário
- Descrição
- Local de estoque
- Data prevista
- Data real
- Congelado
- Número da contagem atual
- Consolidado

O atributo local de estoque determina se o inventário será realizado para todos os locais existentes no sistema (tabela localestoque) ou somente para um local específico. No processo de cadastramento do inventário o usuário deve selecionar a opção adequada para o inventário, sendo o padrão 'TODOS', o que indica que todos os locais serão inventariados.

Os atributos data prevista e data real tendem a conter o mesmo valor. O primeiro informa a previsão de realização e o segundo atributo a data real de congelamento dos estoques.

O atributo congelado informa se o inventário já foi congelado ou não. Um inventário congelado não pode ter seu conteúdo alterado (saldos e produtos participantes).

O atributo número da contagem atual informa a contagem do inventário. Ao ser criado outro inventário, esse atributo é iniciado com um.

O atributo consolidado define o comportamento do sistema na geração das fichas de contagem, bem como na apuração do estoque. Caso o usuário opte pelo inventário consolidado, é gerada apenas uma ficha por produto, da mesma forma será solicitada nas contagens a informação do estoque total do produto. Caso o usuário opte pelo inventário não consolidado, o sistema gera uma ficha para cada local de estoque cadastrado e solicita que o usuário informe as contagens por local de estoque.

Atributo	Tipo	Tamanho	Nulo	Chave
inventario_codigo	Caractere	10	Não	Primária
inventario_desc	Caractere	30	Não	-
local_codigo	Caractere	10	Não	
inventario_data_prevista	Data	-	Não	-
inventario_data_real	Data	-	Não	-
inventario_congelado	Caractere	1	Não	-
inventario_contagem	Inteiro	-	Não	-
inventario_consolidado	Caractere	1	Não	
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

O código SQL mostrado a seguir cria a tabela **inventario** no banco de dados.

```
CREATE TABLE inventario (
  inventario_codigo character varying(10) NOT NULL,
  local_codigo character varying(10) DEFAULT 'TODOS'::character varying,
  inventario_desc character varying(30) NOT NULL,
  inventario_data_prevista date NOT NULL,
  inventario_data_real date,
  inventario_congelado character(1) DEFAULT 'N'::bpchar,
  inventario_contagem integer DEFAULT 1,
  inventario_consolidado character(1) DEFAULT 'N'::bpchar,
  usuario_login character varying(20),
```

```
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT inventario_pkey PRIMARY KEY (inventario_codigo),
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Lista de produtos

A lista de produtos traz a relação de produtos que participarão de um determinado inventário. Basicamente o que precisamos é do código do inventário, do código do produto e do estoque base de comparação.

- Código do inventário
- Código do produto
- Código do local
- Estoque na data base
- Custo médio na data base
- Estoque inventariado

O atributo local de estoque tem os locais cadastrados no sistema. Caso a opção seja pelo estoque consolidado, teremos esse atributo com o valor zero.

Os atributos estoque e custo médio na data base são gerados no momento do congelamento do inventário.

O atributo estoque inventariado é gerado na confirmação do inventário através da funcionalidade análise do inventário.

Atributo	Tipo	Tamanho	Nulo	Chave
inventario_codigo	Caractere	10	Não	Primária
produto_codigo	Caractere	20	Não	Primária
local_codigo	Caractere	10	Não	Primária
produto_estoque	Numérico	10,3	Não	-
produto_customedio	Numérico	12,3	Não	-
inventario_estoque	Numérico	10,3	Não	-
usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

Para criar a tabela inventarioproduto devemos executar no postgreSQL o seguinte código SQL:

```
CREATE TABLE inventarioproduto (
  inventario_codigo character varying(10) NOT NULL,
  produto_codigo character varying(20) NOT NULL,
```

```
    local_codigo character varying(10) NOT NULL,
    produto_customedio numeric(12,3),
    produto_estoque numeric(10,3),
    inventario_estoque numeric(10,3),
    usuario_login character varying(20),
    data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
    CONSTRAINT pk_inventarioproduto
        PRIMARY KEY (inventario_codigo, produto_codigo, local_codigo),
    CONSTRAINT fk_inventario FOREIGN KEY (inventario_codigo)
        REFERENCES inventario (inventario_codigo) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_produto FOREIGN KEY (produto_codigo)
        REFERENCES produto (produto_codigo) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION,
    CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
        REFERENCES usuario (usuario_login) MATCH SIMPLE
        ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Note que não temos uma chave estrangeira para código do local, pois nem sempre o local existirá (quando o usuário optar pelo inventário consolidado, teremos o código do local zerado).

Contagens

No processo de execução de um inventário podemos ter uma, duas, três ou mais contagens do estoque, pois tudo depende da exatidão dos dados e do estoque virtual existente. Para controlar as contagens precisamos de uma tabela que registre para cada inventário e produto os estoques apurados em cada contagem. Os valores encontrados nas contagens são utilizados na hora de definir se o inventário é válido ou se é necessária uma nova contagem para alguns produtos.

- Código do inventário
- Código do produto
- Código do local
- Número da contagem
- Estoque inventariado

O atributo código do local será igual a zero quando o inventário for consolidado.

O atributo estoque inventariado contém o valor informado nas fichas de inventário para cada contagem.

Atributo	Tipo	Tamanho	Nulo	Chave
inventario_codigo	Caractere	10	Não	Primária
produto_codigo	Caractere	20	Não	Primária
local_codigo	Caractere	10	Não	Primária
contagem_numero	Inteiro	-	Não	Primária

Atributo	Tipo	Tamanho	Nulo	Chave
contagem_estoque	Numérico	10,3	Não	-
Usuario_login	Caractere	20	Não	Estrangeira
data_ultima_alteracao	Data/Hora	-	Não	-

A criação da tabela de contagens é realizada pela execução do comando SQL listado em seguida:

```
CREATE TABLE contagem (
  inventario_codigo character varying(10) NOT NULL,
  produto_codigo character varying(20) NOT NULL,
  local_codigo character varying(10) NOT NULL,
  contagem_numero integer DEFAULT 1,
  contagem_estoque numeric(10,3),
  usuario_login character varying(20),
  data_ultima_alteracao timestamp without time zone NOT NULL DEFAULT now(),
  CONSTRAINT contagem_pkey
      PRIMARY KEY (inventario_codigo, produto_codigo,
                   local_codigo, contagem_numero),
  CONSTRAINT fk_inventario FOREIGN KEY (inventario_codigo)
      REFERENCES inventario (inventario_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_produto FOREIGN KEY (produto_codigo)
      REFERENCES produto (produto_codigo) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION,
  CONSTRAINT fk_usuario FOREIGN KEY (usuario_login)
      REFERENCES usuario (usuario_login) MATCH SIMPLE
      ON UPDATE NO ACTION ON DELETE NO ACTION
)
```

Relatórios

Esse módulo disponibiliza uma série de relatórios para a correta tomada de decisão pelos gerenciadores do sistema. A lista apresentada em seguida mostra os relatórios disponibilizados nesse módulo.

- Saldo de produtos
- Estoque por departamento
- Estoque por grupo
- Estoque por local
- Estoque por produto e local
- Produtos com estoque acima do máximo
- Produtos com estoque abaixo do mínimo
- Extrato de movimentação
- Razão de estoque
- Curva ABC de estoque

Além destes relatórios, os cadastros possuem um módulo de impressão disponibilizado pelo framework.

A Figura 2.3 mostra o Modelo Entidade Relacionamento (MER) do sistema de controle de estoque. Não estão listadas as tabelas pertencentes ao framework.

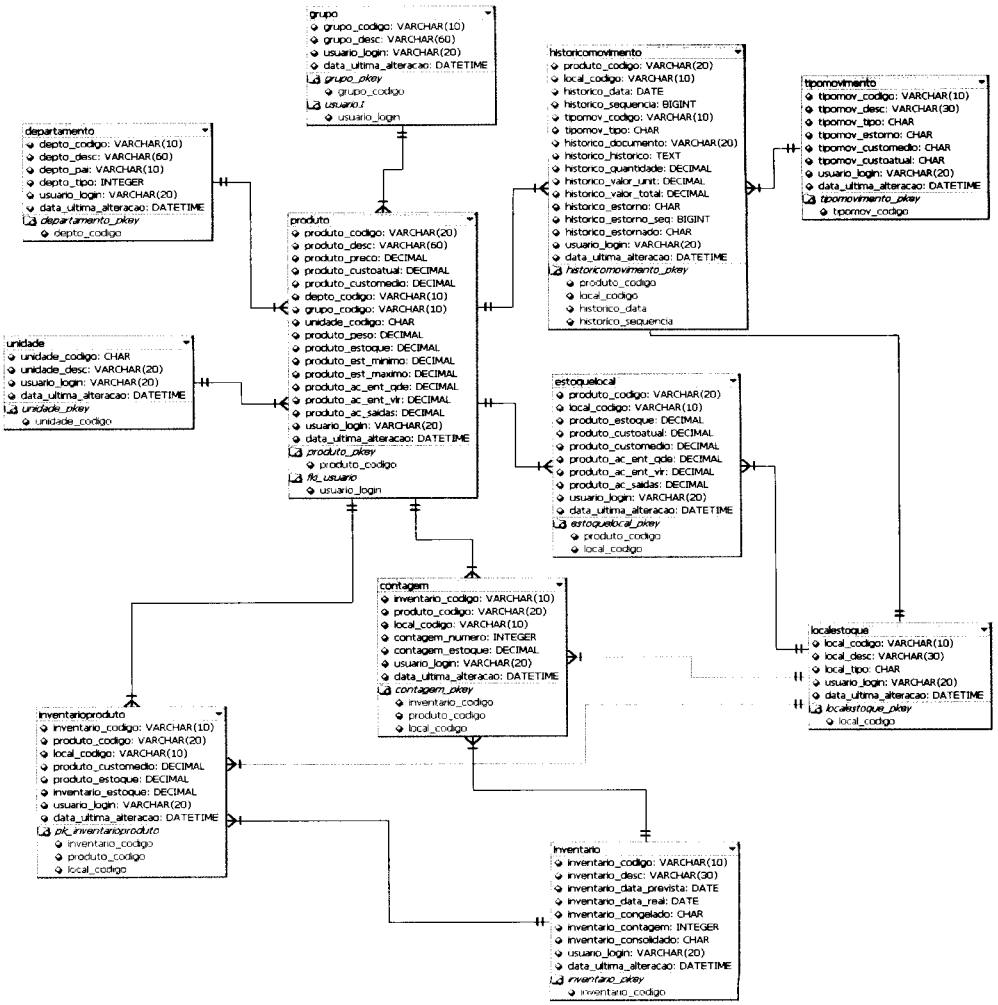


Figura 2.3

Anotações

Capítulo 3

Cadastros Auxiliares

Os cadastros auxiliares compreendem as entidades departamento, grupo, unidade e tipo de movimento. Esses cadastros são, em sua maioria, bem simples, com poucos requisitos.

O framework **fw.PHP** fornece o suporte necessário para que a criação das classes e dos programas de gerenciamento dos cadastros seja bem simples e rápida, minimizando o trabalho do desenvolvedor.

Tudo que precisamos é definir as classes que serão instanciadas, criar os programas de gerenciamento e incluir o programa no menu do sistema. Para incluir os menus, é necessário, antes de qualquer coisa, alterar o arquivo *configmenu.inc* do framework, para que o sistema de controle de estoque esteja visível na árvore de programas.

Lista 1: configmenu.inc

```
<?php
/**
 * Configurador de Sistemas existentes no framework
 * Não incluir o próprio Framework
 * O formato deve ser
 * Array('Sistema/Módulo'=>Diretório, ...)
 * Exemplo:
 *   Array(
 *             'Estoque/Cadastros'=>'estoque/cadastros/',
 *             'Estoque/Relatórios'=>'estoque/relatorios/',
 *             'Estoque/Processos'=>'estoque/processos/'
 *       );
 * A chave 'Sistema/Módulo' será exibida na inclusão de menus
 */
return    Array('Estoque'=>'estoque/');
?>
```

Departamento

A classe para gerenciamento dos departamentos é razoavelmente simples. Precisamos apenas declarar os campos conforme os tipos declarados no banco de dados e ajustar alguns campos para que seja possível maior flexibilidade do sistema. Os ajustes estão relacionados a seguir.

- Código do departamento

  Incluir um botão para busca do próximo código disponível; essa funcionalidade é útil apenas para código numéricos.

- Departamento relacionado

  Executar a busca dinâmica dos departamentos, conforme o conteúdo digitado pelo usuário.

- Tipo de departamento

  O campo deve ter uma lista fixa com os três valores inicialmente definidos, ou seja, Almoxarifado, Armazém interno e Estoque de vendas.

A inclusão do botão para busca do próximo código é disponibilizada pelo framework por meio da definição do atributo comportamento form do campo para 'proximo'. É necessário que na classe seja declarado o método *montaSQLProximoCodigo()* para a correta definição do comando SQL de busca do maior código disponível na tabela.

Para inserir a busca dinâmica dos departamentos relacionados, basta alterar o atributo comportamento form do campo para 'ajax'. Com isso o framework insere a rotina necessária para a busca dos dados no servidor web, dentro da classe **departamento**, conforme os critérios definidos. Na classe é preciso definir o método *montaSQLAjax()* com o comando SQL necessário para a busca dinâmica dos dados. Esse método define em quais campos da tabela a busca deve ser realizada, e o framework cuida da execução do comando e da formatação da resposta que será enviada ao browser.

A definição de uma lista fixa de valores para o atributo tipo de departamento é conseguida pela alteração do atributo comportamento form do campo relacionado para 'select' junto com a definição da lista de valores dentro do padrão exigido pelo framework. A lista deve ser gerada dentro do atributo valor fixo do campo.

Por último precisamos sobrescrever os métodos *incluir()* e *alterar()* da classe para que o método *setUsuarioeData()* seja executado antes da efetivação da operação de inclusão e alteração.

Com tudo isso em mente, podemos construir a classe **departamento**, a qual está listada a seguir, lembrando que o arquivo que vai conter a classe deve ser chamado de *classe_departamento.inc* e estar no diretório *estoque/classes/*.

Lista 2: classe_departamento.inc

```
<?php
/**
 * Classe Departamento
 */
class departamento extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
```

```
        $this->_nome_tabela = 'departamento';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("DEPTO_CODIGO","Código do Departamento",10,
                        null,null,true,false,true,1,null,null,true,$_conn));
        $this->addCampo(new string("DEPTO_DESC","Descrição",60,
                        null,null,true,true,true,3,null,null,false,$_conn));
        $this->addCampo(new string("DEPTO_PAI","Departamento relacionado(pai)",
                        10,null,null,true,true,true,null,null,null,false,$_conn));
        $this->addCampo(new inteiro("DEPTO_TIPO","Tipo",1,
                        null,null,true,true,true,null,null,null));
        $this->addCampo(new string("USUARIO_LOGIN","Usuário",20,
                        null,null,false,false,false,null,null,null,false,$_conn));
        $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",14,
                        null,null,false,false,false,null,null,null,false,$_conn));
        $this->getCampo("DEPTO_CODIGO")->setComportamento_form('proximo');
        $this->getCampo("DEPTO_PAI")->setComportamento_form('ajax');
        $this->getCampo("DEPTO_PAI")->setCallBack(
                        "function(v,qstr) {return '&'+qstr+'&depto='" .
                        "+document.getElementById('DEPTO_CODIGO').value;}");
        $this->getCampo("DEPTO_TIPO")->setComportamento_form('select');
        $this->getCampo("DEPTO_TIPO")->setValor_fixo(
                        Array(
                            Array('valor'=>1,'label'=>'Almoxarifado'),
                            Array('valor'=>2,'label'=>'Armazém interno'),
                            Array('valor'=>3,'label'=>'Estoque vendas')
                        )
                    );
    }

    /**
     * Verifica se o Departamento Pai informado Existe no banco de dados
     *
     * @return boolean
     */
    protected function verificaDeptoPai() {
        if($_POST['DEPTO_PAI']!=''&&$_POST['DEPTO_PAI']!==null) {
            $_deptoClone = new departamento($this->_conn);
            $_res = $_deptoClone->Buscar(
                                    Array("DEPTO_CODIGO"=>$_POST['DEPTO_PAI'])
                                   );
            if($_res===false||$_res<=0) {
                $this->_msgextra = "O Departamento Relacionado " .
                                   "{$_POST['DEPTO_PAI']} não existe...";
                return false;
            }
        }
        return true;
    }

    /**
     * Retorna o comando SQL necessário para a busca dinâmica
     * de departamentos, sendo que verificaremos os campos DEPTO_CODIGO
     * e DEPTO_DESC
     *
     * @return string
     */
    public function montaSQLAjax() {
        return $this->montaSELECT(
            "DEPTO_CODIGO AS CODIGO,DEPTO_DESC AS DESCRICAO",
            "UPPER(DEPTO_CODIGO)!='" . strtoupper($_POST['depto']) . "' AND " .
            "(UPPER(DEPTO_CODIGO) LIKE '%" . strtoupper($_POST['valor']) .
            "%' OR UPPER(DEPTO_DESC) LIKE '%" . strtoupper($_POST['valor']) .
            "%')");
    }
```

```
    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(DEPTO_CODIGO) AS CODIGO FROM {$this->_nome_tabela}";
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function incluir() {
        if($this->verificaDeptoPai()!==false) {
            $this->setUsuarioeData();
            return parent::incluir();
        } else {
            return false;
        }   }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function Alterar() {
        if($this->verificaDeptoPai()!==false) {
            $this->setUsuarioeData();
            return parent::alterar();
        } else {
            return false;
        }
    }
}
?>
```

Note que, na busca dinâmica de dados, ajustamos a função callback do campo para que o departamento atual seja enviado ao servidor, evitando que ele apareça na lista retornada ao browser.

Tanto o método *incluir()* quanto o método *alterar()* chamam o método protegido *verificaDeptoPai()*, que busca o departamento relacionado (pai) na tabela e, caso não exista, retorna falso (false), interrompendo a execução da operação em curso. Esse método é necessário, uma vez que o campo usado para informar o departamento relacionado possui comportamento AJAX, ou seja, é um campo de livre digitação, o que obriga a implementar esse controle extra, pois não temos certeza se o valor informado está correto ou não.

No método *montaSQLAjax()* criamos apelidos para os dois campos presentes no resultado. Isso se deve ao fato de que o framework, no momento da montagem do resultado, busca os campos com nome CODIGO e DESCRICAO. Veja o método *buscaDadosAjax()* da classe **base**, listado a seguir, para melhor compreensão do processo.

```
public function buscaDadosAjax() {
     if(($_strSQL=$this->montaSQLAjax())===null) {
          return Array(0=>Array("valor"=>0,"label"=>"Não Implementado"));
     } else {
          $_opcoes = Array();
          if($this->_conn->executaSQL("{$_strSQL} LIMIT 26")!==false&&
             $this->_conn->getNumRows()>0) {
               while(($_dados=$this->_conn->proximo())!==false) {
                    $_opcoes[] = Array('valor'=>$_dados['CODIGO'],
                                       'label'=>$_dados['DESCRICAO']);
               }
          }
          $this->retornaListAjax($_opcoes);
     }
}
```

Por último precisamos criar o programa a ser incluído no menu do sistema e que será responsável por enviar ao framework os parâmetros necessários para que a classe seja instanciada corretamente. Esse programa é bem simples, pois a maior parte do trabalho é feita pelo framework. Precisamos apenas de três linhas de código, sendo uma para declarar o caminho da classe e o seu nome, outra para o título da aplicação (neste caso definimos o título como 'Departamentos'), e finalmente a última linha com a inclusão do arquivo *instanciaclasse.php5* do framework que é responsável por todo o processamento da classe.

Esse arquivo pode ficar no diretório raiz do sistema , que no caso é *etoque*, ou em um diretório auxiliar mais estratificado, como, por exemplo, *estoque/cadatros* (neste caso é necessário alterar o arquivo *configmenu.inc* e incluir o novo diretório; caso contrário, o programa não é listado nas opções de menu). Chamaremos o programa de *man_departamento.php5*.

Lista 3: man_departamento.php5

```
<?php
/**
 * Cadastro de Departamentos
 *
 */
$_classe = Array("arquivo"=>"estoque/classes/classe_departamento.inc",
                 "nome"=>"departamento");
$_FTitulo = "Departamentos";
return include_once("../instanciaclasse.php5");
?>
```

Para executar o programa, inclua a opção Departamentos no menu do sistema, dentro do menu Estoque e do submenu Cadastros.

Figura 3.1

A Figura 3.2 mostra a tela inicial do cadastro de departamentos. Já a Figura 3.3 exibe o formulário de cadastramento de departamento. Note a presença do botão de busca do próximo código, da busca dinâmica do departamento relacionado e da lista de tipos permitidos.

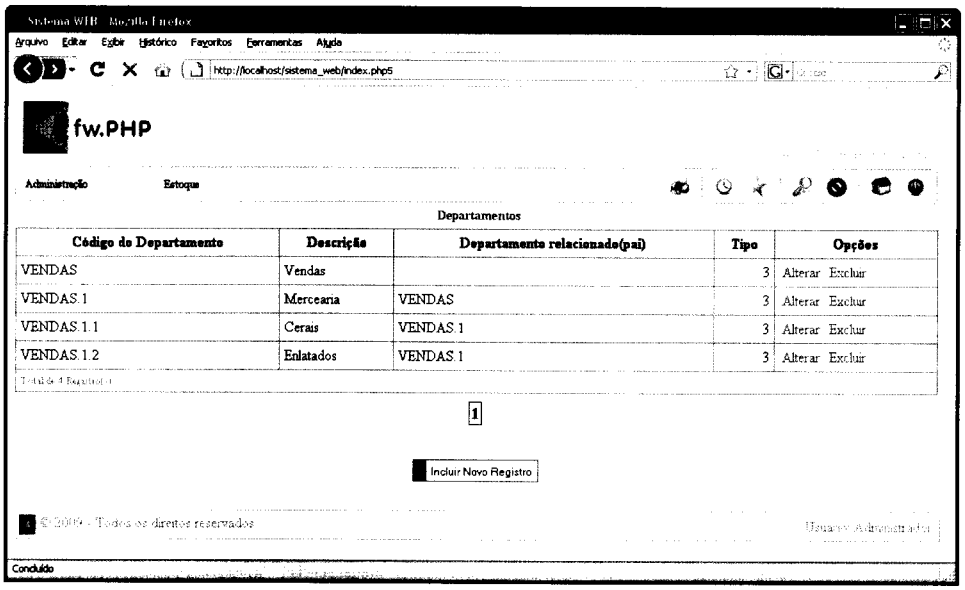


Figura 3.2

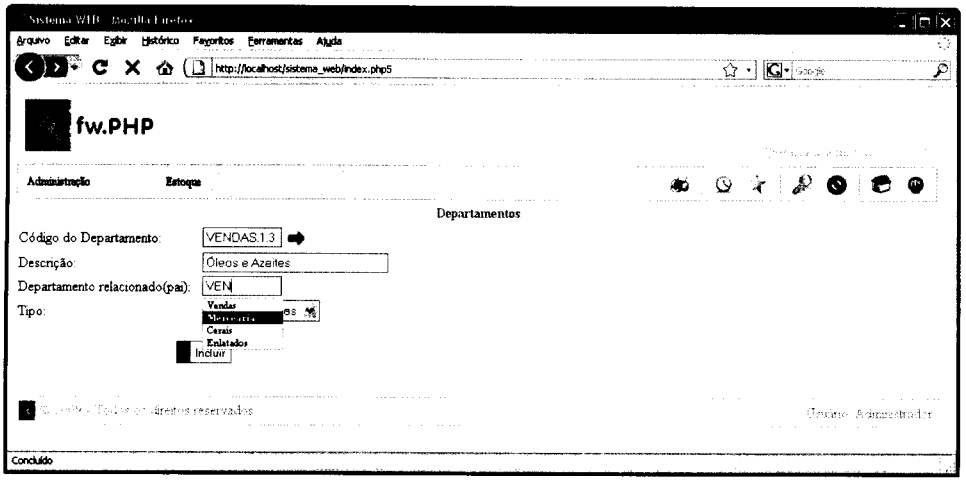


Figura 3.3

Grupo

O cadastro de grupos é bem simples. Precisamos apenas criar a classe em sua estrutura básica, com apenas os dois campos principais do grupo e os campos de usuário e data da última alteração. A classe não precisa de nenhum método extra-especial, como ocorreu com a classe **departamento**. Assim como a classe anterior, esta também deve ser gravada no diretório *estoque/classes/* no arquivo *classe_grupo.inc*.

Lista 4: classe_grupo.inc

```
<?php
/**
 * Classe Grupo
 */
class grupo extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'grupo';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("GRUPO_CODIGO","Código do Grupo",10,
                         null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("GRUPO_DESC","Descrição",60,
                         null,null,true,true,true,3,null,null,false,$_conn));
          $this->addCampo(new string("USUARIO_LOGIN","Usuário",20,
                         null,null,false,false,false,null,null,null,false,$_conn));
          $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",14,
                         null,null,false,false,false,null,null,null,false,$_conn));
          $this->getCampo("GRUPO_CODIGO")->setComportamento_form('proximo');
     }

     /**
      * Retorna o comando SQL necessário para a busca dinâmica
      *
      * @return string
      */
     public function montaSQLAjax() {
          return $this->montaSELECT(
                         "GRUPO_CODIGO AS CODIGO,GRUPO_DESC AS DESCRICAO",
                         "(UPPER(GRUPOO_CODIGO) LIKE '%" .
                         strtoupper($_POST['valor']) . "%' OR " .
                         "UPPER(GRUPO_DESC) LIKE '%" .
                         strtoupper($_POST['valor'])  . "%')");
     }

     /**
      * Retorna o comando SQL necessário para
      * a busca do próximo código
      *
      * @return string
      */
     public function montaSQLProximoCodigo() {
          return "SELECT MAX(GRUPO_CODIGO) AS CODIGO FROM {$this->_nome_tabela}";
     }

     /**
      * Ajustar os campos Usuário e data da
      * última alteração antes de efetivar a transação
      *
      * @return mixed
```

```
     */
    public function incluir() {
        $this->setUsuarioeData();
        return parent::incluir();
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function Alterar() {
        $this->setUsuarioeData();
        return parent::alterar();
    }
}
?>
```

Não há nada de especial no desenvolvimento dessa classe. Apenas ajustamos o atributo "comportamento form" do campo GRUPO_CODIGO para 'proximo', assim o botão para buscar o próximo código disponível é exibido, sobrescrevemos os métodos *incluir()* e *alterar()* e então desenvolvemos, para uso posterior, o método *montaSQLAjax()*.

O programa de manutenção também é muito simples, assim como foi o programa de manutenção de departamentos. Chamaremos o programa de *man_grupo.php5*.

Lista 5: man_grupo.php5

```
<?php
/**
 * Cadastro de Grupos
 *
 */
$_classe = Array("arquivo"=>"estoque/classes/classe_grupo.inc","nome"=>"grupo");
$_FTitulo = "Grupos";
return include_once("../instanciaclasse.php5");
?>
```

Antes de executar o programa devemos incluí-lo na lista de programas disponíveis no menu do sistema. Vamos colocá-lo junto com o programa de manutenção de departamentos dentro do menu Cadastros que é um submenu de Estoque.

Figura 3.4

As Figuras 3.5 e 3.6 mostram a tela inicial de cadastro de grupos e o formulário para cadastramento.

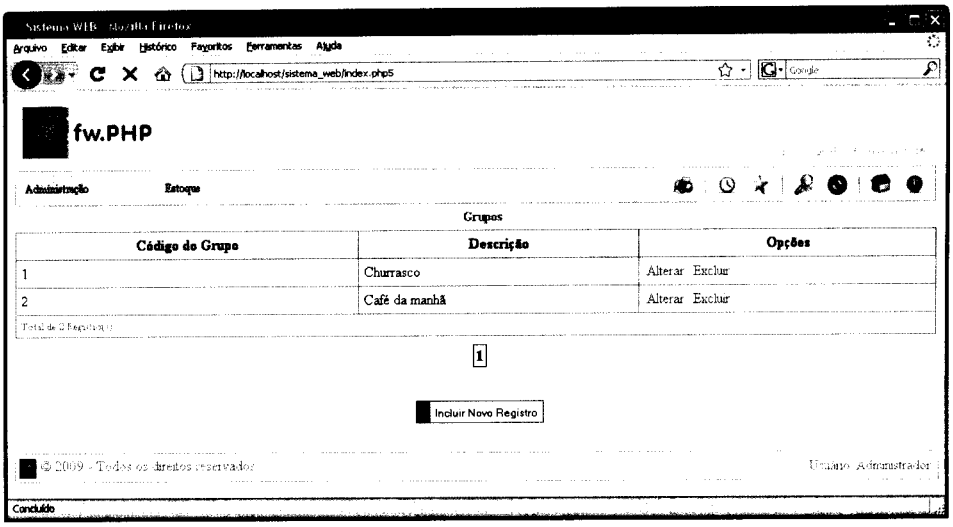


Figura 3.5

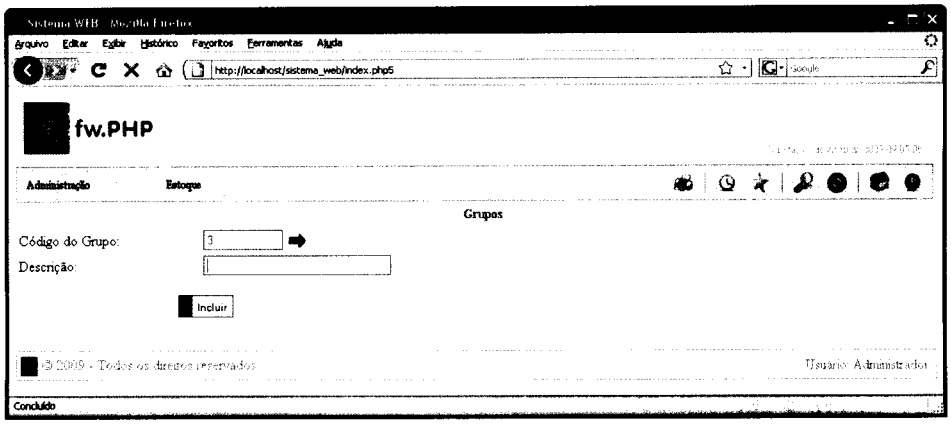


Figura 3.6

Unidade

Da mesma forma que o cadastro de grupos, o cadastro de unidade de medida é muito simples, exigindo pouco esforço de programação. Precisamos apenas definir a classe com seus campos, não havendo a necessidade de nenhum tratamento extra.

A tendência é que a lista de unidade de medida seja pequena; desta forma não precisamos implementar a busca dinâmica (AJAX) de unidades. Em vez disso a classe disponibiliza um método para retornar todo o seu conteúdo em uma lista, no padrão exigido pelo framework, para ser inserida como opções de seleção em outra classe do sistema.

O arquivo que contém a classe **unidade** deve se chamar *classe_unidade.inc* e, assim como os demais arquivos de classes, ser gravado no diretório *estoque/classes*.

Lista 6: classe_unidade.inc

```
<?php
/**
 * Classe Unidades de medida
 */

class unidade extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'unidade';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("UNIDADE_CODIGO","Código da Unidade",4,
                         null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("UNIDADE_DESC","Descrição",20,
                         null,null,true,true,true,2,null,null,false,$_conn));
          $this->addCampo(new string("USUARIO_LOGIN","Usuário",20,
                         null,null,false,false,false,null,null,null,false,$_conn));
          $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",14,
                         null,null,false,false,false,null,null,null,false,$_conn));
     }

     /**
      * Retorna a lista de unidades cadastradas no sistema
      *
      * @return Array
      */
     public function retornaLista() {
          $_un = Array();
          if($this->_conn->executaSQL($this->montaSELECT())!==false) {
               while(($_dados=$this->_conn->proximo())!==false) {
                    $_un[] = Array(
                                   'valor'=>$_dados['UNIDADE_CODIGO'],
                                   'label'=>$_dados['UNIDADE_DESC']
                              );
               }
          }
          return $_un;
     }


     /**
      * Ajustar os campos Usuário e data da
      * última alteração antes de efetivar a transação
      *
      * @return mixed
      */
     public function incluir() {
          $this->setUsuarioeData();
          return parent::incluir();
     }

     /**
      * Ajustar os campos Usuário e data da
      * última alteração antes de efetivar a transação
      *
      * @return mixed
      */
     public function Alterar() {
```

```
            $this->setUsuarioeData();
            return parent::alterar();
        }
}
?>
```

O método *retornaLista()* deve ser executado dentro das classes que tenham relacionamento com a classe **unidade**, por exemplo, a classe **produto**, e que precisem fornecer ao usuário uma lista fechada das unidades de medida disponíveis.

Um exemplo da sua utilização é listado a seguir:

```
$_un = new unidade($this->_conn);
$this->getCampo("UNIDADE_CODIGO")->SetComportamento_form("select");
$this->getCampo("UNIDADE_CODIGO")->setValor_fixo($_un->retornaLista());
```

Seguindo o que estudamos até agora, o programa de manutenção do cadastro de unidades é igualmente simples de ser construído. Vamos chamá-lo de *man_unidade.php5* e, assim como os demais programas, gravaremos o arquivo no diretório *estoque/*.

Lista 7: man_unidade.php5

```
<?php
/**
 * Cadastro de Unidades de medida
 *
 */
$_classe = Array("arquivo"=>"estoque/classes/classe_unidade.inc",
                 "nome"=>"unidade");
$_FTitulo = "Unidades de medida";
return include_once("../instanciaclasse.php5");
?>
```

Vamos incluir o programa, assim como os itens anteriores, no menu Cadastros do sistema de Estoque.

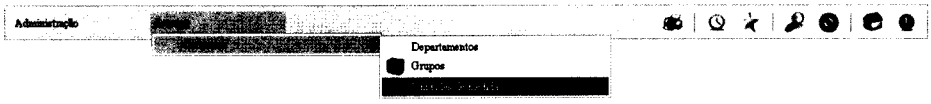


Figura 3.7

As Figuras 3.8 e 3.9 mostram a página inicial do cadastro de unidades e a página com o formulário de manutenção do cadastro.

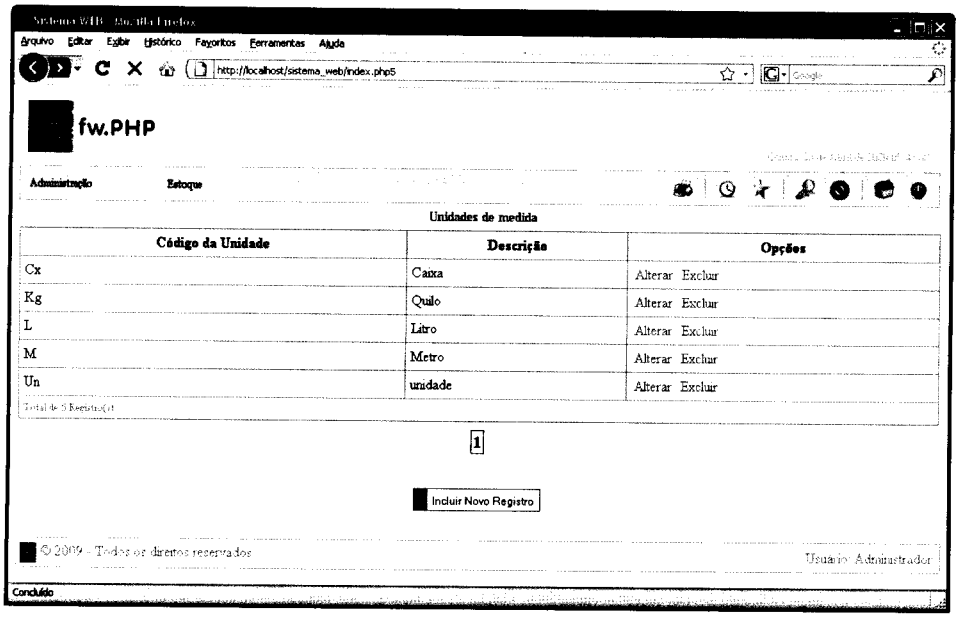


Figura 3.8

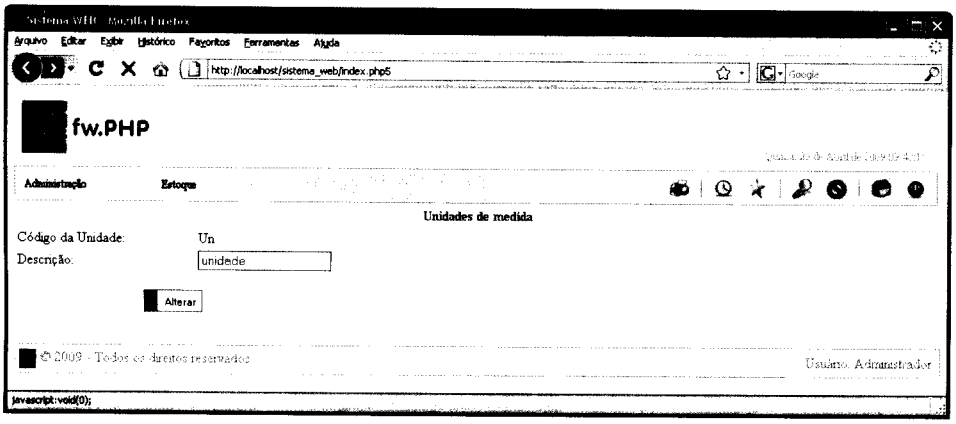


Figura 3.9

Tipo de movimento

O cadastro de tipos de movimento é um pouco mais complicado que os outros exibidos até agora, pois precisamos monitorar algumas situações durante o processo.

Conforme mostrado no capítulo 2, temos alguns requisitos para o cadastro do tipo de movimento:

1. O campo "Afeta custo médio" é habilitado apenas para o tipo de movimento de entrada que não seja um estorno e para o tipo de movimento de saída que seja um estorno.
2. O campo "Afeta custo atual" é habilitado apenas para o tipo de movimento de entrada que não seja um estorno e para o tipo de movimento de saída que seja um estorno.

Estes requisitos exigem que seja implementado um controle da opção selecionada pelo usuário, o qual será inserido no formulário pelo método *processaCampoFormulario()*. Conforme descrito no capítulo 1, esse método recebe três parâmetros. O primeiro identifica o nome do campo, o segundo é a instância da classe **tag** que contém a definição do campo e o último contém o nome do método que efetuou a chamada. No caso vamos fazer a implementação dos controles nos campos TIPOMOV_TIPO e TIPOMOV_ESTORNO e somente quando o método que originou a chamada for o *getFormulario*. Será inserida nesses dois campos uma rotina para habilitar ou desabilitar os campos TIPOMOV_CUSTOMEDIO e TIPOMOV_CUSTOATUAL, conforme a opção selecionada pelo usuário. A decisão de habilitar ou não os campos de custo será construída em javascript e inserida nos dois campos do formulário dentro do evento Onclick de cada campo. O código javascript necessário para esse processamento é mostrado a seguir:

```
var tm = document.getElementsByName('TIPOMOV_TIPO');
var est= document.getElementById('TIPOMOV_ESTORNO');
var tipo = (tm[0].checked===true ? tm[0].value : tm[1].value);
var ativar = false;
if((tipo==='E'&&est.checked===false)||(tipo==='S'&&est.checked===true)) {
     ativar = true;
}
document.getElementById('TIPOMOV_CUSTOMEDIO').checked = ativar;
document.getElementById('TIPOMOV_CUSTOMEDIO').disabled = !ativar;
document.getElementById('TIPOMOV_CUSTOATUAL').checked = ativar;
document.getElementById('TIPOMOV_CUSTOATUAL').disabled = !ativar;
```

O campo TIPOMOV_TIPO deve ter o atributo "comportamento form" ajustado para o tipo 'radio' com duas opções 'Entrada' ou 'Saída'.

Já os campos TIPOMOV_ESTORNO, TIPOMOV_CUSTOMEDIO e TIPOMOV_CUSTOATUAL terão o atributo "comportamento form" ajustado para o tipo 'checkbox', e o valor dos campos, quando estiverem marcados, será 'S'. É necessário, ainda, tratar os dados enviados pelo formulário de manutenção, uma vez que os campos do formulário do tipo 'checkbox' não são enviados para o servidor web caso não estejam marcados. A rotina de tratamento verifica essa situação e atribui o valor 'N' nestes casos. Aproveitamos esse ajuste para verificar se os valores estão em conformidade com os requisitos estabelecidos para o cadastro.

```
/**
 * Ajusta os valores dos campos que são CheckBox
 *
 */
```

```
protected function AjustaValores() {
     if($this->getCampo('TIPOMOV_ESTORNO')->getValor()!='S') {
          $this->getCampo('TIPOMOV_ESTORNO')->setValor('N');
     }
     $_macar = ($this->getCampo('TIPOMOV_TIPO')->getValor()=='E'&&
                    $this->getCampo('TIPOMOV_ESTORNO')->GetValor()=='S')||
               ($this->getCampo('TIPOMOV_TIPO')->getValor()=='S'&&
                    $this->getCampo('TIPOMOV_ESTORNO')->GetValor()=='N')
                         ? false : true;
     if($this->getCampo('TIPOMOV_CUSTOMEDIO')->getValor()!='S'||!$_marcar) {
          $this->getCampo('TIPOMOV_CUSTOMEDIO')->setValor('N');
     }
     if($this->getCampo('TIPOMOV_CUSTOATUAL')->getValor()!='S'||!$_marcar) {
          $this->getCampo('TIPOMOV_CUSTOATUAL')->setValor('N');
     }
}
```

Esse ajuste é incluído tanto no método *incluir()* quanto no método *alterar()* da classe, antes de a operação ser efetivada.

Essa classe terá, ainda, um método para retornar de forma seletiva a lista de tipos de movimentos cadastrados. Esse método necessita de um parâmetro de entrada que define o critério de filtragem. São três critérios, a saber:

'E' = Entradas

'S' = Saídas

'EST' = Estorno

Conforme o critério desejado uma lista diferente de tipos de movimentos é retornada.

```
/**
 * Retorna a lista de unidades cadastradas no sistema
 * Conforme o parâmetro enviado
 * 'E' = Entradas
 * 'S' = Saídas
 * 'EST' = Estorno
 * @return Array
 */
public function retornaLista($_tipo='E') {
     $_lst = Array();
     $_where = ($_tipo=='EST'
                    ? "TIPOMOV_ESTORNO='S'"
                    : "TIPOMOV_TIPO='{$_tipo}' AND TIPOMOV_ESTORNO='N'"
                );
     if($this->_conn->executaSQL($this->montaSELECT('*',$_where))!==false) {
          while(($_dados=$this->_conn->proximo())!==false) {
               $_lst[] = Array('valor'=>$_dados['TIPOMOV_CODIGO'],
                               'label'=>$_dados['TIPOMOV_DESC']);
          }
     }
     return$_lst;
}
```

A seguir está a listagem completa da classe **tipomovimento** que deve ser salva no arquivo *classe_tipomovimento.inc* no diretório *estoque/classes*.

Lista 8: classe_tipomovimento.inc

```
<?php
/**
 * Classe Tipo de Movimento
 */

class tipomovimento extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'tipomovimento';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("TIPOMOV_CODIGO",
                         "Código do Tipo de movimento",10,null,null,
                         true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("TIPOMOV_DESC","Descrição",30,
                         null,null,true,true,true,5,null,null,false,$_conn));
          $this->addCampo(new string("TIPOMOV_TIPO","Entrada/Saída",1,
                         null,null,true,true,true,1,null,null,false,$_conn));
          $this->addCampo(new string("TIPOMOV_ESTORNO","Estorno",1,
                         null,null,true,true,true,1,null,null,false,$_conn));
          $this->addCampo(new string("TIPOMOV_CUSTOMEDIO","Afeta Custo médio",1,
                         null,null,true,true,true,1,null,null,false,$_conn));
          $this->addCampo(new string("TIPOMOV_CUSTOATUAL","Afeta Custo atual",1,
                         null,null,true,true,true,1,null,null,false,$_conn));
          $this->addCampo(new string("USUARIO_LOGIN","Usuário",20,
                         null,null,false,false,false,null,null,null,false,$_conn));
          $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",14,
                         null,null,false,false,false,null,null,null,false,$_conn));
          $this->getCampo("TIPOMOV_CODIGO")->setComportamento_form('proximo');
          $this->getCampo("TIPOMOV_TIPO")->setComportamento_form('radio');
          $this->getCampo("TIPOMOV_TIPO")->setValor_fixo(
                         Array(
                              Array('label'=>'Entrada','valor'=>'E','marcar'=>true),
                              Array('label'=>'Saída','valor'=>'S')
                         )
                    );
          $this->getCampo("TIPOMOV_ESTORNO")->setComportamento_form('checkbox');
          $this->getCampo("TIPOMOV_ESTORNO")->setValor_fixo('S');
          $this->getCampo("TIPOMOV_CUSTOMEDIO")->setComportamento_form('checkbox');
          $this->getCampo("TIPOMOV_CUSTOMEDIO")->setValor_fixo('S');
          $this->getCampo("TIPOMOV_CUSTOATUAL")->setComportamento_form('checkbox');
          $this->getCampo("TIPOMOV_CUSTOATUAL")->setValor_fixo('S');
          if($_GET['ACAO']=='INC') {
               $this->getCampo("TIPOMOV_CUSTOMEDIO")->setMarcar(true);
               $this->getCampo("TIPOMOV_CUSTOATUAL")->setMarcar(true);
          }
     }

     /**
      * Ajusta os valores dos campos que são CheckBox
      *
      */
     protected function AjustaValores() {
          if($this->getCampo('TIPOMOV_ESTORNO')->getValor()!='S') {
               $this->getCampo('TIPOMOV_ESTORNO')->setValor('N');
          }
          $_macar =($this->getCampo('TIPOMOV_TIPO')->getValor()=='E'&&
                         $this->getCampo('TIPOMOV_ESTORNO')->GetValor()=='S')||
                    ($this->getCampo('TIPOMOV_TIPO')->getValor()=='S'&&
                         $this->getCampo('TIPOMOV_ESTORNO')->GetValor()=='N')
                         ? false : true;
          if($this->getCampo('TIPOMOV_CUSTOMEDIO')->getValor()!='S'||!$_marcar) {
               $this->getCampo('TIPOMOV_CUSTOMEDIO')->setValor('N');
```

```
        }
        if($this->getCampo('TIPOMOV_CUSTOATUAL')->getValor()!='S'||!$_marcar) {
            $this->getCampo('TIPOMOV_CUSTOATUAL')->setValor('N');
        }
    }

    /**
     * Processa os campos para o formulário de cadastro
     * Somente os campos TIPOMOV_TIPO e TIPOMOV_ESTORNO são processados
     *
     * @param string $_nome
     * @param tag  $_campo
     * @param string $_metodo
     */
    protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
        if(stripos($_metodo,"getformulario")!==false) {
            switch($_nome) {
            case 'TIPOMOV_TIPO':
            case 'TIPOMOV_ESTORNO':
                $_ev = new Eventos();
                $_ev->setOnclick(
                    "var tm = document.getElementsByName('TIPOMOV_TIPO');" .
                    "var est= document.getElementById('TIPOMOV_ESTORNO');" .
                    "var tipo = (tm[0].checked===true " .
                    "? tm[0].value : tm[1].value);" .
                    "var ativar = false;" .
                    "if((tipo==='E'&&est.checked===false)||" .
                    "(tipo==='S'&&est.checked===true)) {" .
                    "    ativar = true;" .
                    "};" .
                    "var cm=document.getElementById('TIPOMOV_CUSTOMEDIO');" .
                    "cm.checked = ativar;" .
                    "cm.disabled = !ativar;" .
                    "var ca=document.getElementById('TIPOMOV_CUSTOATUAL');" .
                    "ca.checked = ativar;" .
                    "ca.disabled = !ativar;"
                );
                $_campo->SetEventos($_ev);
                break;
            }
        }
    }

    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(TIPOMOV_CODIGO) AS CODIGO FROM {$this->_nome_tabela}";
    }

    /**
     * Retorna a lista de unidades cadastradas no sistema
     * Conforme o parâmetro enviado
     * 'E' = Entradas
     * 'S' = Saídas
     * 'EST' = Estorno
     * @return Array
     */
    public function retornaLista($_tipo='E') {
        $_lst = Array();
        $_where = ($_tipo=='EST'
                    ? "TIPOMOV_ESTORNO='S'"
```

```
                    : "TIPOMOV_TIPO='{$_tipo}' AND TIPOMOV_ESTORNO='N'"
                );
        if($this->_conn->executaSQL($this->montaSELECT('*',$_where))!==false) {
            while(($_dados=$this->_conn->proximo())!==false) {
                $_lst[] = Array('valor'=>$_dados['TIPOMOV_CODIGO'],
                                'label'=>$_dados['TIPOMOV_DESC']);
            }
        }
        return $_lst;
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function incluir() {
        $this->setUsuarioeData();
        $this->AjustaValores();
        return parent::incluir();
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function Alterar() {
        $this->setUsuarioeData();
        $this->AjustaValores();
        return parent::alterar();
    }
}
?>
```

O programa de manutenção de tipos de movimentos está descrito na lista a seguir.

Lista 9: man_tipomovimento.php5

```
<?php
/**
 * Cadastro de Tipos de movimentos
 *
 */
$_classe = Array("arquivo"=>"estoque/classes/classe_tipomovimento.inc",
                 "nome"=>"tipomovimento");
$_FTitulo = "Tipos de Movimentos";
return include_once("../instanciaclasse.php5");
?>
```

Vamos incluir o programa no menu Cadastros dentro de Estoque. A Figura 3.10 mostra como fica a estrutura do menu.

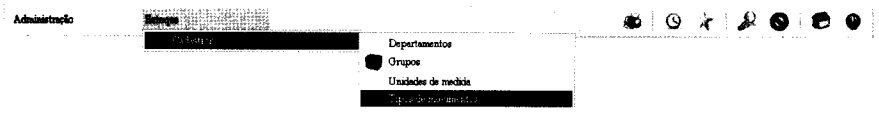


Figura 3.10

As Figuras 3.11 e 3.12 exibem a página inicial da manutenção de tipos de movimentos e a página com o formulário para inclusão de um novo tipo de movimento.

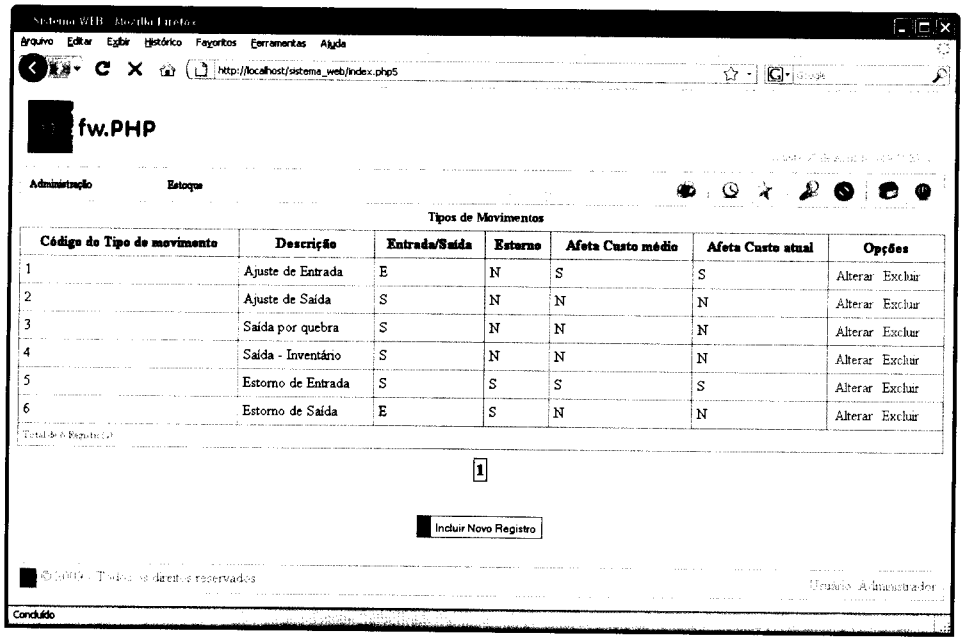


Figura 3.11

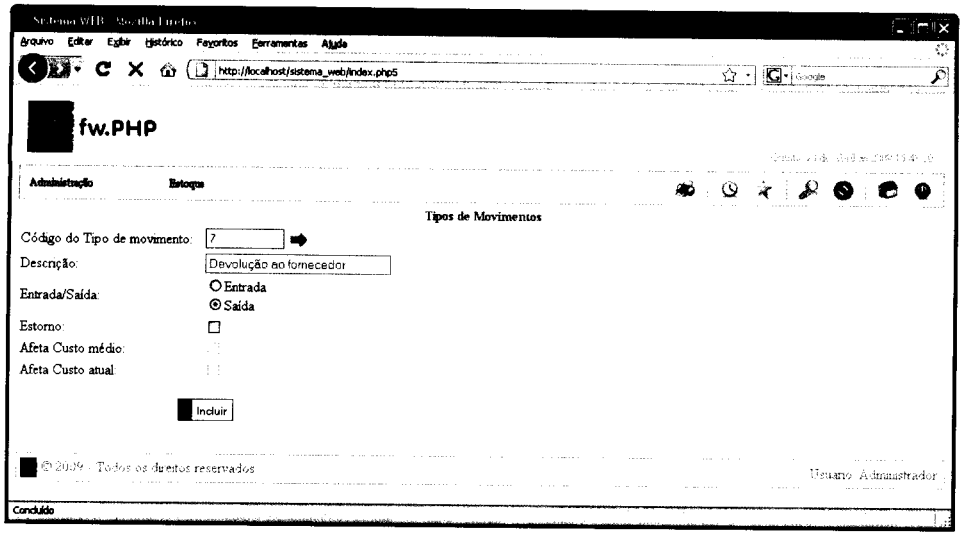


Figura 3.12

Capítulo 4

Cadastro de Produtos

A principal entidade em um sistema de controle de estoques é, sem sombra de dúvida, a que descreve os produtos, pois tudo no sistema gira em torno deles. Este capítulo descreve em detalhes essa entidade, cria a classe relacionada e desenvolve as regras de negócio relacionadas a esse cadastro.

Essa entidade foi formalmente definida no capítulo 2, em que descrevemos os atributos necessários ao cadastro de produtos.

- Código
- Descrição
- Preço atual
- Custo atual
- Custo médio
- Departamento
- Grupo
- Unidade
- Peso por unidade
- Estoque atual
- Estoque mínimo
- Estoque máximo
- Acumulado de entradas quantidade
- Acumulado de entradas valor
- Acumulado de saídas

Ainda no capítulo 2, tivemos os requisitos definidos para o cadastro.

1. Deve existir pelo menos um departamento cadastrado para que seja possível cadastrar um produto.

2. Deve existir pelo menos um grupo cadastrado para o cadastramento de um produto.
3. Deve existir pelo menos uma unidade cadastrada para que seja possível cadastrar um produto.
4. Os atributos custo atual e custo médio são gerados na entrada de estoque. O custo atual reflete o custo da última entrada e o custo médio é a divisão do valor total de entradas pela quantidade acumulada de entradas.
5. Departamento e grupo vêm das respectivas tabelas e servem para que o produto possua duas classificações, uma relacionada ao mundo físico e a segunda para uma visualização diferente dos produtos.
6. O atributo unidade define a unidade de medida do produto, por exemplo, quilo, litro, caixa etc.
7. Peso por unidade serve para termos uma medida comum a todos os produtos.
8. Os atributos estoque mínimo e máximo são utilizados para cálculos do estoque, possibilitando mecanismos de controle para evitar estoque muito baixo ou muito alto.

A lista de atributos, juntamente com os requisitos definidos para a entidade, permite que a classe de produtos seja especificada. Alguns atributos da entidade possuem comportamentos diferenciados e precisam ser implementados na classe respectiva, além disso algumas regras de negócio devem estar implementadas na classe. A seguir temos uma lista do que deve ser implementado na classe **produto**.

- Código do produto

  Inserir o botão para geração do próximo código de produto disponível; essa funcionalidade só tem uso em cadastros numéricos.

- Departamento

  É exibida no formulário de cadastro de produtos, tanto em inclusão quanto em alteração, a lista de departamentos existentes.

- Grupo

  É exibida a lista de grupos cadastrados no processo de inclusão ou alteração de produtos.

- Unidade

  O framework mostra no processo de cadastramento de produtos a lista das unidades de medida disponíveis.

- Estoque máximo

  O estoque máximo não pode ser menor do que o estoque mínimo cadastrado para o produto.

A inclusão do botão para busca do próximo código é disponibilizada pelo framework por meio da definição do atributo comportamento form do campo para 'proximo'. É necessário que na classe seja declarado o método *montaSQLProximoCodigo()* para a correta definição do comando SQL de busca do maior código disponível na tabela.

A definição de uma lista fixa de valores para os atributos departamento, grupo e unidade de medida é conseguida pela alteração do atributo comportamento form dos campos relacionados para 'select' junto com a definição da lista de valores dentro do padrão exigido pelo framework. A lista deve ser gerada dentro do atributo valor fixo dos campos relacionados.

O atributo estoque máximo da classe deve estar ligado ao atributo estoque mínimo, sendo este último o referencial de valor mínimo para o primeiro.

Precisamos, ainda, sobrescrever os métodos *incluir()* e *alterar()* da classe para que o método *setUsuarioeData()* seja executado antes da efetivação da operação de inclusão e alteração.

Os requisitos de produto dizem, ainda, que é necessário verificar, antes de disponibilizar o formulário de cadastramento, se as entidades departamento, grupo e unidade de medida possuem pelo menos um registro cada uma; caso contrário, isto é, se qualquer uma delas estiver vazia, o cadastro do produto não deve ser mostrado ao usuário.

Essa verificação deve ser feita por meio do método *processaAcao()* que deve ser sobrescrito da classe **base**, tratando somente o momento da geração do formulário de inclusão de produtos, uma vez que para alteração e exclusão não precisamos validar essa entidade, já que as tabelas estão fisicamente ligadas, impedindo que uma das entidades esteja vazia (as chaves estrangeiras criadas na tabela de produtos impedem que os departamentos, grupos e unidade de medida relacionados sejam excluídos caso haja um ou mais produtos ligados a essas entidades).

Classe produto

A classe **produto** reflete os requisitos e comentários descritos anteriormente, implementando os métodos necessários e descrevendo os campos da tabela. A princípio a classe deve conter o mínimo necessário para que o cadastramento e a manutenção dos produtos sejam executados pelos usuários, mas à medida que desenvolvermos o sistema de controle de estoque, vamos implementar métodos extras na classe (ainda neste capítulo temos alguns métodos extras).

Para construir a classe **produto** existem as seguintes definições:

1. O campo PRODUTO_CODIGO é uma instância da classe **string**, é exibido na lista inicial, marcado como chave primária da tabela e tem tamanho mínimo de um caractere. Além disso, o atributo comportamento form para esse campo é

ajustado para 'proximo', fazendo com que seja exibido no formulário, na tela de inclusão de novo produto, o botão para a busca do próximo código disponível, lembrando que essa funcionalidade somente é válida caso os códigos no cadastro sejam numéricos.

2. O campo PRODUTO_DESC é uma instância da classe **string**, é exibido na lista inicial e define-se como cinco o número mínimo de caracteres aceitos para esse campo.

3. O campo DEPTO_CODIGO é uma instância da classe **string** e exibido na lista inicial. O atributo comportamento form desse campo será alterado para 'select' e a classe deve gerar a lista dos departamentos disponíveis.

4. O campo GRUPO_CODIGO é uma instância da classe **string** e exibido na lista inicial. O atributo comportamento form desse campo será alterado para 'select' e a classe deve gerar a lista dos grupos disponíveis.

5. O campo UNIDADE_CODIGO é uma instância da classe **string** e exibido na lista inicial. O atributo comportamento form desse campo será alterado para 'select' e a classe deve gerar a lista das unidades disponíveis.

6. O campo PRODUTO_PESO é uma instância da classe **float** (derivada da classe **numero**), e devemos informar o valor cinco no parâmetro número de casas decimais do construtor da classe, uma vez que este é o número de casas decimais definido no banco de dados. Esse campo não é exibido na lista de produtos da página inicial do cadastro.

7. O campo PRODUTO_PRECO indica o preço atual cadastrado para o produto, é uma instância da classe **dinheiro** (derivada da classe **numero**) e exibido na lista de produtos. Não será exigido um valor mínimo para esse campo.

8. O campo PRODUTO_ESTOQUE contém o saldo de estoque atual do produto, é uma instância da classe **float** (derivada da classe **numero**) e é exibido na lista de produtos. Não será exigido um valor mínimo para esse campo, que será atualizado por meio dos processos de movimentação de produtos, tanto entradas quanto saídas, e no inventário.

9. O campo PRODUTO_CUSTOATUAL informa o custo atual do produto, geralmente associado ao valor da última entrada informada no sistema, é uma instância da classe **dinheiro** (derivada da classe **numero**) e não é exibido na lista de produtos. Não será exigido um valor mínimo para esse campo, que terá seu valor atualizado no processo de entrada de produtos.

10. O campo PRODUTO_EST_MINIMO define a quantidade mínima de estoque para o produto, é uma instância da classe **float** (derivada da classe **numero**) e não é exibido na lista de produtos. Não será exigido um valor mínimo para esse campo.

11. O campo PRODUTO_EST_MAXIMO define a quantidade máxima de estoque para o produto, é uma instância da classe **float** (derivada da classe **numero**) e não é exibido na lista de produtos. Será exigido um valor mínimo para esse campo, o qual está relacionado ao valor que foi informado para o estoque mínimo (PRODUTO_EST_MINIMO), uma vez que o estoque máximo não pode ser menor que o estoque mínimo.

12. O campo PRODUTO_AC_ENT_QDE define a quantidade acumulada de entradas do produto, é uma instância da classe **float** (derivada da classe **numero**) e não é exibido na lista de produtos. Não será exigido um valor mínimo para esse campo. Esse campo não aparece em nenhum formulário de cadastramento, sendo utilizado como acumulador interno do sistema, tendo seu valor alimentado no processo de entrada de estoques, e é utilizado para cálculo do custo médio do produto.

13. O campo PRODUTO_AC_ENT_VLR define o valor acumulado de entradas do produto, é uma instância da classe **float** (derivada da classe **numero**) e não é exibido na lista de produtos. Não será exigido um valor mínimo para esse campo. Esse campo não aparece em nenhum formulário de cadastramento, sendo utilizado como acumulador interno do sistema, tendo seu valor alimentado no processo de entrada de estoques, e é utilizado para cálculo do custo médio do produto.

14. O campo PRODUTO_AC_SAIDAS define a quantidade acumulada de saídas do produto, é uma instância da classe **float** (derivada da classe **numero**) e não é exibido na lista de produtos. Não será exigido um valor mínimo para esse campo. Esse campo não aparece em nenhum formulário de cadastramento, sendo utilizado como acumulador interno do sistema, tendo seu valor alimentado no processo de saídas de estoques.

A geração da lista de valores aceitos para os campos DEPTO_CODIGO, GRUPO_CODIGO e UNIDADE_CODIGO pode ser feita em dois lugares da classe. O primeiro e mais evidente é na criação da classe, isto é, dentro do método de construção chamado *__construct()*, pois é nesse momento que definimos os campos da classe e seus comportamentos. O problema dessa opção é que temos a geração das listas sempre que a classe **produto** for instanciada, apesar de utilizarmos esses valores apenas nos formulários de cadastramento do produto, o que gera um processamento inútil em muitas situações.

Para contornar este problema em potencial, vamos criar as listas somente no momento em que realmente forem necessárias, ou seja, antes de gerar o formulário de cadastramento. Para alcançar este objetivo, é necessário sobrescrever o método *getFormulario()*, executando o processamento necessário antes de gerar o formulário de cadastramento. Vamos aproveitar esta sobrescrita do método para realizar as validações prévias definidas nas regras de negócio do produto.

1. Deve existir pelo menos um departamento cadastrado para que seja possível cadastrar um produto.
2. Deve existir pelo menos um grupo cadastrado para o cadastramento de um produto.
3. Deve existir pelo menos uma unidade cadastrada para que seja possível cadastrar um produto.

Estas regras são verificadas apenas no processo de inclusão de produtos, uma vez que não faz sentido verificar no processo de alteração do produto, pois a tabela de produto foi definida com chaves estrangeiras, relacionando o produto com as tabelas de departamento, grupo e unidade de medida, o que torna impossível a exclusão de todos os registros de qualquer uma destas três tabelas (pois existe pelo menos um registro em cada uma delas ligado aos produtos cadastrados).

Assim como fizemos anteriormente, devemos sobrescrever os métodos de inclusão e alteração de dados, ou seja, os métodos *incluir()* e *alterar()*, para gerar os valores corretos para os campos USUARIO_LOGIN e DATA_ULTIMA_ALTERACAO.

Devemos alterar o método *montaSQLProximoCodigo()* para geração do comando SQL adequado para a busca do próximo valor disponível do código do produto, ou seja, do campo PRODUTO_CODIGO.

A classe **produto**, assim como todas as classes definidas até agora para o sistema de controle de estoques, deve ser gerada no arquivo *classe_produto.inc* que deve ser gravado no diretório *estoque/classes*.

Lista 1: classe_produto.inc

```
<?php
/**
 * Classe produto
 */

class produto extends base {
    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'produto';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("PRODUTO_CODIGO","Código do Produto",
                    20,null,null,true,false,true,1,null,null,true,$_conn));
        $this->addCampo(new string("PRODUTO_DESC","Descrição",
                    60,null,null,true,true,true,5,null,null,false,$_conn));
        $this->addCampo(new string("DEPTO_CODIGO","Código do Departamento",
                    10,null,null,true,true,true,1,null,null,false,$_conn));
        $this->addCampo(new string("GRUPO_CODIGO","Código do Grupo",
                    10,null,null,true,true,true,1,null,null,false,$_conn));
        $this->addCampo(new string("UNIDADE_CODIGO","Código da Unidade",
                    4,null,null,true,true,false,1,null,null,false,$_conn));
        $this->addCampo(new float("PRODUTO_PESO","Peso do Produto",
                    10,null,null,true,true,true,0.001,null,null,false,5));
        $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque Atual",
                    12,null,null,false,false,true,0.001,null,null,false));
        $this->addCampo(new dinheiro("PRODUTO_PRECO","Preço Atual",
```

```
                    12,null,null,true,true,true,0,null,null,false));
        $this->addCampo(new dinheiro("PRODUTO_CUSTOATUAL","Custo Atual",
                    12,null,null,true,true,true,0,null,null,false));
        $this->addCampo(new dinheiro("PRODUTO_CUSTOMEDIO","Custo médio",
                    12,null,null,false,false,false,0,null,null,false));
        $this->addCampo(new float("PRODUTO_EST_MINIMO",
                    "Estoque Minimo do Produto",
                    10,null,null,true,true,false,0,null,null,false));
        $this->addCampo(new float("PRODUTO_EST_MAXIMO",
                    "Estoque Máximo do Produto",
                    10,null,null,true,true,false,'[PRODUTO_EST_MINIMO]',
                    null,null,false));
        $this->addCampo(new float("PRODUTO_AC_ENT_QDE",
                    "Acumulado de Entradas (Quantidade)",
                    18,null,null,false,false,false,1,null,null,false));
        $this->addCampo(new float("PRODUTO_AC_ENT_VLR",
                    "Acumulado de Entradas (Valor)",
                    18,null,null,false,false,false,null,null,null,false));
        $this->addCampo(new float("PRODUTO_AC_SAIDAS",
                    "Acumulado de Saídas (Quantidade)",
                    18,null,null,false,false,false,null,null,null,false));
        $this->addCampo(new string("USUARIO_LOGIN","Usuário",
                20,null,null,false,false,false,null,null,null,false,$_conn));
        $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",
                14,null,null,false,false,false,null,null,null,false,$_conn));
        $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('proximo');
    }

    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(PRODUTO_CODIGO) AS CODIGO FROM {$this->_nome_tabela}";
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function incluir() {
        $this->setUsuarioeData();
        return parent::incluir();
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function Alterar() {
        $this->setUsuarioeData();
        return parent::alterar();
    }

    public function getFormulario($_funcao='INC') {
        $_depto = new departamento($this->_conn);
        $_depto->Buscar();
        if($_funcao=='INC') {
```

```
            if($_depto->getConn()->getNumrows()<1) {
                die(
                    utf8_encode("<span style='color:red;font-size:20px;" .
                        "text-align:center;'>É necessário ter " .
                        "pelo menos 1 departamento cadastrado</span>")
                );
            }
        }
        $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
        while($_depto->proximo()!==false) {
            $_ldep[] =
                Array(
                    "valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                    "label"=>$_depto->getCampo('DEPTO_DESC')->getValor()
                );
        }
        $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
        $_grupo = new grupo($this->_conn);
        $_grupo->Buscar();
        if($_funcao=='INC') {
            if($_grupo->getConn()->getNumrows()<1) {
                die(
                    utf8_encode("<span style='color:red;font-size:20px;" .
                        "text-align:center;'>É necessário ter pelo menos " .
                        "1 grupo cadastrado</span>")
                );
            }
        }
        $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
        while($_grupo->proximo()!==false) {
            $_lgrp[] =
                Array(
                    "valor"=>$_grupo->getCampo('GRUPO_CODIGO')->getValor(),
                    "label"=>$_grupo->getCampo('GRUPO_DESC')->getValor()
                );
        }
        $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
        $_un = new unidade($this->_conn);
        $_un->Buscar();
        if($_funcao=='INC') {
            if($_un->getConn()->getNumrows()<1) {
                die(
                    utf8_encode("<span style='color:red;font-size:20px;" .
                        "text-align:center;'>É necessário ter pelo menos " .
                        "1 unidade cadastrada</span>"));
            }
        }
        $this->getCampo("UNIDADE_CODIGO")->setComportamento_form('select');
        while($_un->proximo()!==false) {
            $_lun[] =
                Array(
                    "valor"=>$_un->getCampo('UNIDADE_CODIGO')->getValor(),
                    "label"=>$_un->getCampo('UNIDADE_DESC')->getValor());
        }
        $this->getCampo("UNIDADE_CODIGO")->setValor_fixo($_lun);
        $this->_num_colunas_form = 2;
        return parent::getFormulario($_funcao);
    }
}
?>
```

Note que definimos o número de colunas do formulário de cadastramento ($this->_num_colunas_form) com o valor dois, assim temos uma melhor distribuição dos campos no formulário, evitando que seja gerada uma página muito comprida. Esse comportamento não é original do framework, tendo sido implementado no capítulo 1 deste livro.

Esta é apenas a versão inicial da classe **produto**. É necessária a implementação de vários outros métodos para que o sistema fique completo.

O programa de manutenção de produtos, que chamaremos de *man_produto.php5*, é tão simples quanto qualquer um dos demais programas implementados até agora.

Lista 2: man_produto.php5

```
<?php
/**
 * Cadastro de Produtos
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_produto.inc",
              "nome"=>"produto");
$_FTitulo = "Produtos";
return include_once("../instanciaclasse.php5");
?>
```

As Figuras 4.1 e 4.2 mostram, respectivamente, a página inicial do programa de cadastro de produtos e a página para inclusão de um novo produto.

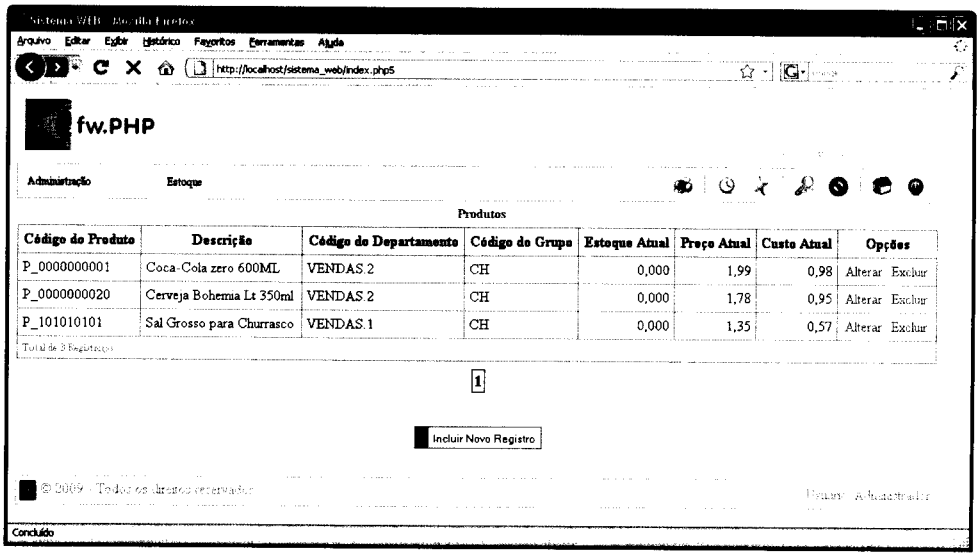


Figura 4.1

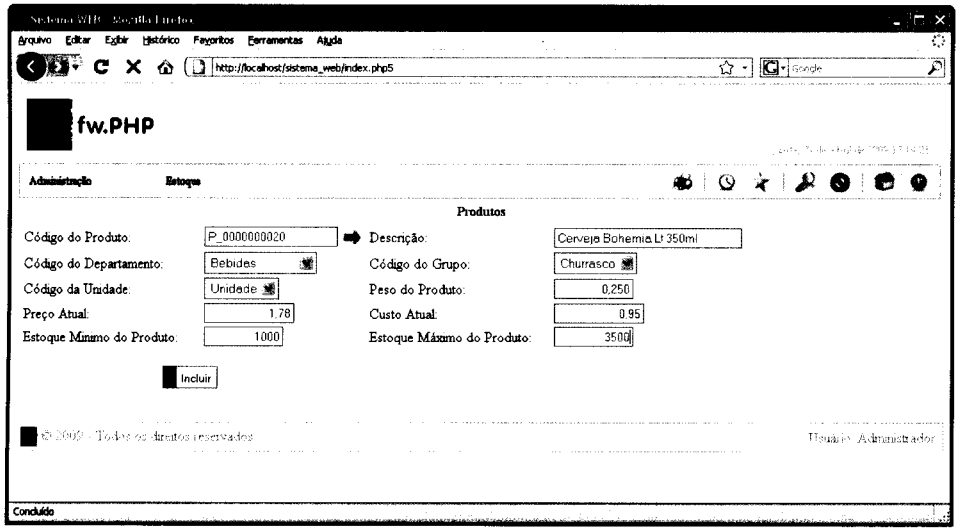


Figura 4.2

Métodos adicionais

Além dos métodos básicos mostrados anteriormente, vamos desenvolver métodos auxiliares para uso futuro no sistema de controle de estoque. Na classe **produto** precisamos ter métodos que disponibilizem funções para cálculo do custo médio do produto, ajustem os acumulados de entrada e saída, alterem o saldo atual de estoque e outros métodos auxiliares, mas fundamentais para o desenvolvimento do sistema.

A lista a seguir mostra os métodos que vamos desenvolver, porém não é a lista definitiva de métodos da classe, pois no decorrer do livro teremos a necessidade de criar outros métodos para completar o sistema.

calculaCustoMedio

Esse método realiza a alteração do atributo custo médio do produto (PRODUTO_CUSTOMEDIO), ajustando-o conforme os valores dos atributos que compõem o cálculo do custo médio. Utilizamos a fórmula:

Custo médio = acumulado de entradas valor / acumulado entradas quantidade

Esta fórmula somente é valida se o acumulado de entradas quantidade for diferente de zero, uma vez que não existe divisão por zero.

A implementação desse método em PHP está listada em seguida:

```
public function calculaCustoMedio() {
     $this->getCampo("PRODUTO_CUSTOMEDIO")->setValor(
          ($this->getCampo("PRODUTO_AC_ENT_QDE")->getValor())<>0
               ?    ($this->getCampo("PRODUTO_AC_ENT_VLR")->getValor()) /
                    ($this->getCampo("PRODUTO_AC_ENT_QDE")->getValor())
               :    0
          );
}
```

acumulaEntradas

Recebe como parâmetros a quantidade e o valor do movimento que foi inserido no sistema. De posse destes altera os atributos acumulado de entradas quantidade (PRODUTO_AC_ENT_QDE) e acumulado de entradas valor (PRODUTO_AC_ENT_VLR).

```
public function acumulaEntradas($_qde,$_vlr) {
     $this->getCampo("PRODUTO_AC_ENT_QDE")->setValor(
               $this->getCampo("PRODUTO_AC_ENT_QDE")->getValor()+$_qde);
     $this->getCampo("PRODUTO_AC_ENT_VLR")->setValor(
               $this->getCampo("PRODUTO_AC_ENT_VLR")->getValor()+$_vlr);
}
```

acumulaSaidas

Assim como o método anterior, esse método é executado no processo de movimentação de estoque, sendo chamado quando uma saída é executada. Ele recebe apenas um parâmetro, que informa a quantidade que deve ser acumulada, então soma a quantidade informada ao valor existente no atributo acumulado de saídas (PRODUTO_AC_SAIDAS).

```
public function acumulaSaidas($_qde) {
     $this->getCampo("PRODUTO_AC_SAIDAS")->setValor(
          $this->getCampo("PRODUTO_AC_SAIDAS")->getValor()+$_qde);
}
```

acertaSaldoAtual

Esse método mantém o saldo de estoque do produto (PRODUTO_ESTOQUE) sempre atualizado. Apenas a quantidade movimentada é necessária como parâmetro de entrada do método. O detalhe importante é que a quantidade deve vir corretamente sinalizada, ou seja, se for uma entrada, a quantidade deve ser positiva, e caso seja uma saída, deve ser negativa.

```
protected function acertaSaldoAtual($_qde) {
     $this->getCampo("PRODUTO_ESTOQUE")->setValor(
               $this->getCampo("PRODUTO_ESTOQUE")->getValor() + $_qde);
}
```

acertaCustoAtual

Toda vez que uma entrada for executada e o tipo de movimento associado estiver marcado para afetar o custo atual (TIPOMOV_CUSTOATUAL) o método de nome *acertaCustAtual* deve ser executado, recebendo como parâmetro de entrada o novo custo do produto. De posse do novo custo o método altera o valor do parâmetro custo atual do produto (PRODUTO_CUSTOATUAL).

```
protected function acertaCustoAtual($_vlr) {
    $this->getCampo("PRODUTO_CUSTOATUAL")->setValor($_vlr);
}
```

acertaEstoque

Este é o método-chave para o processo de ajuste de estoque e é chamado sempre que um movimento qualquer for executado (entrada, saída, estorno, inventário). Ele recebe três parâmetros. O primeiro informa a quantidade movimentada, o segundo informa o valor da movimentação (somente para entradas ou estornos) e o último é o tipo de movimento que foi utilizado na geração do movimento. É por meio dessas três informações que o método realiza os ajustes necessários, alterando o saldo de estoque, ajustando os acumulados de entrada e saída e ainda os custos médio e atual.

Um detalhe interessante é que o lançamento de estorno exige que a quantidade e o valor tenham o sinal trocado para negativo, uma vez que, nestes casos, precisamos subtrair os valores dos acumulados, pois estamos desfazendo uma transação.

```
public function acertaEstoque($_qde,$_vlr,tipomovimento $_tipo) {
    $this->acertaSaldoAtual(
                            $_tipo->getCampo("TIPOMOV_TIPO")->getValor()=='E'
                                ? $_qde
                                : $_qde*-1
                            );
    if($_tipo->getCampo("TIPOMOV_ESTORNO")->getValor()=='S') {
        $_qde *= -1;
        $_vlr *= -1;
    }
    if($_tipo->getCampo("TIPOMOV_CUSTOMEDIO")->getValor()=='S') {
        $this->acumulaEntradas($_qde,$_vlr);
        $this->calculaCustoMedio();
    } elseif($_tipo->getCampo("TIPOMOV_TIPO")->getValor()=='S') {
        $this->acumulaSaidas($_qde);
    }
    if($_tipo->getCampo("TIPOMOV_CUSTOATUAL")->getValor()=='S'&&$_vlr>0) {
        $this->acertaCustoAtual($_vlr);
    }
}
```

produtoAbaixoMinimo

Em algumas situações, como em relatórios ou em algum processo de reposição de estoque, precisamos saber quais produtos estão com estoque perigosamente baixo, ou seja, com seu estoque abaixo do mínimo estabelecido. O método *produtoAbaixoMinimo()* retorna o comando SQL necessário para obtermos a lista de produtos nesta situação. Seu único parâmetro de entrada é a ordem em que os registros devem ser gerados, sendo o padrão o código do departamento (DEPTO_CODIGO).

```
public function produtosAbaixoMinimo($_ordem="DEPTO_CODIGO") {
     return $this->montaSELECT(   "*",
                                  "PRODUTO_ESTOQUE<PRODUTO_EST_MINIMO",
                                  $_ordem
                              );
}
```

produtoAcimaMaximo

O método *produtoAcimaMaximo()*, assim como o método anterior, é responsável por gerar um comando SQL para busca selecionada de produtos, só que em vez de comparar o estoque atual do produto com o estoque mínimo, esse método compara o estoque atual com o estoque máximo, retornando o SQL necessário para gerar a lista de produtos com estoque acima do máximo. O método recebe como parâmetro de entrada, assim como o método anterior, a ordem desejada para a lista gerada, e o valor-padrão também é o código do departamento.

```
public function produtosAcimaMaximo($_ordem="DEPTO_CODIGO") {
     return $this->montaSELECT(   "*",
                                  "PRODUTO_ESTOQUE>PRODUTO_EST_MAXIMO",
                                  $_ordem
                              );
}
```

Estes são os métodos auxiliares que deixamos, neste momento, disponíveis na classe **produto**. À medida que novos métodos forem exigidos, eles serão adicionados.

A listagem seguinte mostra o formato final (neste momento) da classe.

Lista 3: classe_produto.inc

```
<?php
/**
 * Classe produto
 */
class produto extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'produto';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("PRODUTO_CODIGO","Código do Produto",
                    20,null,null,true,false,true,1,null,null,true,$_conn));
```

```
        $this->addCampo(new string("PRODUTO_DESC","Descrição",
                60,null,null,true,true,true,5,null,null,false,$_conn));
        $this->addCampo(new string("DEPTO_CODIGO","Código do Departamento",
                10,null,null,true,true,true,1,null,null,false,$_conn));
        $this->addCampo(new string("GRUPO_CODIGO","Código do Grupo",
                10,null,null,true,true,true,1,null,null,false,$_conn));
        $this->addCampo(new string("UNIDADE_CODIGO","Código da Unidade",
                4,null,null,true,true,false,1,null,null,false,$_conn));
        $this->addCampo(new float("PRODUTO_PESO","Peso do Produto",
                10,null,null,true,true,false,0.001,null,null,false,5));
        $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque Atual",
                12,null,null,false,false,true,0.001,null,null,false));
        $this->addCampo(new dinheiro("PRODUTO_PRECO","Preço Atual",
                12,null,null,true,true,true,0,null,null,false));
        $this->addCampo(new dinheiro("PRODUTO_CUSTOATUAL","Custo Atual",
                12,null,null,true,true,true,0,null,null,false));
        $this->addCampo(new dinheiro("PRODUTO_CUSTOMEDIO","Custo médio",
                12,null,null,false,false,false,0,null,null,false));
        $this->addCampo(new float("PRODUTO_EST_MINIMO",
                "Estoque Minimo do Produto",
                10,null,null,true,true,false,0,null,null,false));
        $this->addCampo(new float("PRODUTO_EST_MAXIMO",
                "Estoque Máximo do Produto",
                10,null,null,true,true,false,'[PRODUTO_EST_MINIMO]',
                null,null,false));
        $this->addCampo(new float("PRODUTO_AC_ENT_QDE",
                "Acumulado de Entradas (Quantidade)",
                18,null,null,false,false,false,1,null,null,false));
        $this->addCampo(new float("PRODUTO_AC_ENT_VLR",
                "Acumulado de Entradas (Valor)",
                18,null,null,false,false,false,null,null,null,false));
        $this->addCampo(new float("PRODUTO_AC_SAIDAS",
                "Acumulado de Saídas (Quantidade)",
                18,null,null,false,false,false,null,null,null,false));
        $this->addCampo(new string("USUARIO_LOGIN","Usuário",
                20,null,null,false,false,false,null,null,null,false,$_conn));
        $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",
                14,null,null,false,false,false,null,null,null,false,$_conn));
        $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('proximo');
    }

    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(PRODUTO_CODIGO) AS CODIGO FROM {$this->_nome_tabela}";
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function incluir() {
        $this->setUsuarioeData();
        return parent::incluir();
    }

    /**
     * Ajustar os campos Usuário e data da
```

```
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function Alterar() {
        $this->setUsuarioeData();
        return parent::alterar();
    }

    public function getFormulario($_funcao='INC') {
        $_depto = new departamento($this->_conn);
        $_depto->Buscar();
        if($_funcao=='INC') {
            if($_depto->getConn()->getNumrows()<1) {
                die(utf8_encode("<span style='color:red;font-size:20px;" .
                        "text-align:center;'>É necessário ter pelo menos " .
                        "1 departamento cadastrado</span>"));
            }
        }
        $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
        while($_depto->proximo()!==false) {
            $_ldep[] =
                Array(
                    "valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                    "label"=>$_depto->getCampo('DEPTO_DESC')->getValor()
                );
        }
        $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
        $_grupo = new grupo($this->_conn);
        $_grupo->Buscar();
        if($_funcao=='INC') {
            if($_grupo->getConn()->getNumrows()<1) {
                die(utf8_encode("<span style='color:red;font-size:20px;" .
                        "text-align:center;'>É necessário ter pelo menos " .
                        "1 grupo cadastrado</span>"));
            }
        }
        $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
        while($_grupo->proximo()!==false) {
            $_lgrp[] =
                Array(
                    "valor"=>$_grupo->getCampo('GRUPO_CODIGO')->getValor(),
                    "label"=>$_grupo->getCampo('GRUPO_DESC')->getValor()
                );
        }
        $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
        $_un = new unidade($this->_conn);
        $_un->Buscar();
        if($_funcao=='INC') {
            if($_un->getConn()->getNumrows()<1) {
                die(utf8_encode("<span style='color:red;font-size:20px;" .
                        "text-align:center;'>É necessário ter pelo menos " .
                        "1 unidade cadastrada</span>"));
            }
        }
        $this->getCampo("UNIDADE_CODIGO")->setComportamento_form('select');
        while($_un->proximo()!==false) {
            $_lun[] =
                Array(
                    "valor"=>$_un->getCampo('UNIDADE_CODIGO')->getValor(),
                    "label"=>$_un->getCampo('UNIDADE_DESC')->getValor()
                );
        }
        $this->getCampo("UNIDADE_CODIGO")->setValor_fixo($_lun);
```

```
        $this->_num_colunas_form = 2;
        return parent::getFormulario($_funcao);
    }

    /**
     * Cálculo do Custo médio
     * Fórmula
     *   CM = (Acumulado_entradas_valor)/(Acumulado_entradas_qde)
     *
     */
    public function calculaCustoMedio() {
        $this->getCampo("PRODUTO_CUSTOMEDIO")->setValor(
            $this->getCampo("PRODUTO_AC_ENT_QDE")->getValor()<>0
                    ?   ($this->getCampo("PRODUTO_AC_ENT_VLR")->getValor())
                        / ($this->getCampo("PRODUTO_AC_ENT_QDE")->getValor())
                    :   0
        );
    }

    /**
     * Gera os acumulados de Entrada
     *
     * @param float $_qde
     * @param float $_vlr
     */
    public function acumulaEntradas($_qde,$_vlr) {
        $this->getCampo("PRODUTO_AC_ENT_QDE")->setValor(
                $this->getCampo("PRODUTO_AC_ENT_QDE")->getValor()+$_qde);
        $this->getCampo("PRODUTO_AC_ENT_VLR")->setValor(
                $this->getCampo("PRODUTO_AC_ENT_VLR")->getValor()+$_vlr);
    }

    /**
     * Gera o acumulado de Saídas
     *
     * @param float $_qde
     */
    public function acumulaSaidas($_qde) {
        $this->getCampo("PRODUTO_AC_SAIDAS")->setValor(
                $this->getCampo("PRODUTO_AC_SAIDAS")->getValor()+$_qde);
    }

    /**
     * Acerta o saldo atual do produto
     *
     * @param float $_qde
     */
    protected function acertaSaldoAtual($_qde) {
        $this->getCampo("PRODUTO_ESTOQUE")->setValor(
                $this->getCampo("PRODUTO_ESTOQUE")->getValor() + $_qde);
    }

    /**
     * Atualiza o Custo Atual do produto
     *
     * @param float $_vlr
     */
    protected function acertaCustoAtual($_vlr) {
        $this->getCampo("PRODUTO_CUSTOATUAL")->setValor($_vlr);
    }

    /**
     * Realiza os acertos necessários no produto
     *
```

```
     * @param float $_qde
     * @param float $_vlr
     * @param tipomovimento $_tipo
     */
    public function acertaEstoque($_qde,$_vlr,tipomovimento $_tipo) {
        $this->acertaSaldoAtual(
                $_tipo->getCampo("TIPOMOV_TIPO")->getValor()=='E'
                    ? $_qde
                    : $_qde*-1
        );
        if($_tipo->getCampo("TIPOMOV_ESTORNO")->getValor()=='S') {
            $_qde *= -1;
            $_vlr *= -1;
        }
        if($_tipo->getCampo("TIPOMOV_CUSTOMEDIO")->getValor()=='S') {
            $this->acumulaEntradas($_qde,$_vlr);
            $this->calculaCustoMedio();
        } elseif($_tipo->getCampo("TIPOMOV_TIPO")->getValor()=='S') {
            $this->acumulaSaidas($_qde);
        }
        if($_tipo->getCampo("TIPOMOV_CUSTOATUAL")->getValor()=='S'&&$_vlr>0) {
            $this->acertaCustoAtual($_vlr);
        }
    }

    /**
     * Retorna o SQL para busca dos produtos com estoque abaixo do mínimo
     *
     * @param string $_ordem
     * @return string
     */
    public function produtosAbaixoMinimo($_ordem="DEPTO_CODIGO") {
        return $this->montaSELECT(  "*",
                                    "PRODUTO_ESTOQUE<PRODUTO_EST_MINIMO",
                                    $_ordem
                                );
    }

    /**
     * Retorna o SQL para busca dos produtos com estoque acima do máximo
     *
     * @param string $_ordem
     * @return string
     */
    public function produtosAcimaMaximo($_ordem="DEPTO_CODIGO") {
        return $this->montaSELECT(  "*",
                                    "PRODUTO_ESTOQUE>PRODUTO_EST_MAXIMO",
                                    $_ordem
                                );
    }
}
?>
```

Anotações

Capítulo 5

Entrada de Estoque

Com os cadastros construídos podemos iniciar o desenvolvimento das rotinas de movimentação de estoque. Basicamente temos as rotinas de entrada de estoque e a saída de estoque (juntamente com as variantes de estorno de entrada e saída). A entrada de estoque, objeto deste capítulo, pode originar-se de várias operações, por exemplo:

- Entrada de nota fiscal de compra;
- Transferência de outro local;
- Devolução de vendas;
- Ajuste manual.

Cada uma destas operações define como o sistema ajusta os dados dos produtos. Para cada uma das operações devemos ter um tipo de movimento cadastrado e devidamente configurado no que diz respeito ao custo médio e ao custo atual do produto.

Para desenvolver os módulos relacionados à movimentação de estoque dos produtos, devemos definir várias classes auxiliares, além da classe principal de movimentação. Além disso, devemos realizar atividades preparatórias para que os processos estejam totalmente funcionais no momento de sua utilização. A seguir temos uma lista do que é necessário fazer para implementar a movimentação e o processo de entrada de estoque.

- Definição da classe de estoque por local
- Definição da classe de movimentação de estoque (histórico de movimentos)
- Definição dos comandos javascript associados à movimentação
- Definição da classe de entradas
- Criação dos tipos de movimentos relacionados ao processo de entradas
- Criação de um local de estoque padrão
- Definição do programa de entrada de estoque
- Geração do menu para movimentação

A classe de estoque por local será definida neste momento, pois somente agora ela é necessária. A classe contém as definições do estoque por produto e por local, sendo vinculada tanto à tabela de produtos quanto à tabela de locais.

A classe de movimentação de estoque é feita neste capítulo e utilizada no restante do livro, sendo uma das principais classes do sistema, uma vez que é responsável pelo gerenciamento de toda a movimentação.

A classe de entradas de estoques é uma especialização da classe de movimentação, contendo definição específica do processo de entradas. Assim como existe a classe de entradas, temos necessidade, nos próximos capítulos, de definir classes especializadas também para saídas e estornos.

Para que o sistema funcione corretamente, é necessário que exista pelo menos um local de estoque cadastrado e um tipo de movimento de entrada cadastrado (da mesma forma é exigido à frente que tipos de movimentos de saída e estorno também estejam cadastrados).

A seguir detalhamos cada um dos itens da lista de tarefas para a construção do módulo de movimentação de estoque e, em particular, neste capítulo, o processo de entrada de estoques.

Definição da classe estoquelocal

Qualquer movimentação de estoque no sistema implica na geração ou atualização de valores do estoque do local selecionado. Na entrada e somente na entrada de estoque temos o processo de inicialização do estoque do local, quando criamos o estoque do local, uma vez que o estoque do produto ou do local não podem ser negativos.

A classe **estoquelocal** é bem parecida com a classe **produto**, só que mais simples, pois não temos vários campos nessa classe e não há necessidade de gerar formulários, uma vez que toda a atualização é feita internamente, por meio das classes de movimentação.

Para construir a classe **estoquelocal** temos as seguintes definições:

1. O campo PRODUTO_CODIGO é uma instância da classe **string**, tendo seu valor determinado nas classes de movimentação de estoque. Além disso, ele faz parte da chave primária da tabela.
2. O campo LOCAL_CODIGO é uma instância da classe **string**, tendo seu valor determinado nas classes de movimentação de estoque. Além disso, ele faz parte da chave primária da tabela.
3. O campo PRODUTO_ESTOQUE contém o saldo de estoque atual do produto e local (um produto pode ter vários locais, desta forma o somatório dos estoques de todos os locais deve ser igual ao estoque atual do produto) e é uma instância da

classe **float** (derivada da classe **numero**). Esse campo é atualizado por meio dos processos de movimentação de produtos, tanto entradas quanto saídas, e no inventário.

4. O campo PRODUTO_CUSTOATUAL informa o custo atual do produto e local (que pode ser diferente do custo atual do produto), geralmente associado ao valor da última entrada informada no sistema, e é uma instância da classe **dinheiro** (derivada da classe **numero**). Esse campo tem seu valor atualizado no processo de entrada de produtos.

5. O campo PRODUTO_CUSTOMEDIO contém o custo médio do produto no local. Esse custo é calculado a cada entrada (desde que o tipo de movimento esteja marcado para afetar o custo médio). O campo é uma instância da classe **dinheiro**.

6. O campo PRODUTO_AC_ENT_QDE define a quantidade acumulada de entradas do produto em um determinado local e é uma instância da classe **float** (derivada da classe **numero**). Esse campo é utilizado como um acumulador interno do sistema, tendo seu valor alimentado no processo de entrada de estoques, e é utilizado para cálculo do custo médio do produto.

7. O campo PRODUTO_AC_ENT_VLR define o valor acumulado de entradas do produto em um determinado local e é uma instância da classe **float** (derivada da classe **numero**). Ele é utilizado como um acumulador interno do sistema, tendo seu valor alimentado no processo de entrada de estoques, e é utilizado para cálculo do custo médio do produto.

8. O campo PRODUTO_AC_SAIDAS define a quantidade acumulada de saídas do produto em um determinado local e é uma instância da classe **float** (derivada da classe **numero**). Esse campo é utilizado como um acumulador interno do sistema, tendo seu valor alimentado no processo de saídas de estoques.

Uma vez que toda a operação de inclusão e alteração dos registros vinculados a essa classe é feita por rotinas internas do sistema, não precisamos nos preocupar com o controle de produtos e locais de estoque, se eles estão cadastrados ou não, pois se pressupõe que essas validações já foram executadas pelos processos chamadores.

Assim com a classe de produtos, necessitamos de métodos auxiliares para cálculo dos custos médio e atual, acumulados de entradas em quantidade e valor e acumulados de saídas.

Além desses métodos, é necessária a criação de um método de inicialização do estoque para um determinado local e produto. Esse método deve ser criado com o objetivo de inserir um novo registro na tabela, representando o processo de criação do estoque do produto em um determinado local. Somente o código do produto e o código do local são necessários; os demais campos da classe serão zerados antes de ser disparado o processo de criação do registro.

Atendendo ao padrão já definido, vamos criar a classe **estoquelocal** dentro do arquivo *classe_estoquelocal.inc*, o qual deve ser gravado no diretório *estoque/classes/*.

Lista 1: classe_estoquelocal.inc

```
<?php
/**
 * Classe Estoque por Local
 */
class estoquelocal extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'estoquelocal';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("PRODUTO_CODIGO","Código do Produto",
                         20,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("LOCAL_CODIGO","Código do Local",
                         10,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque Atual",
                         12,null,null,true,false,true,0.001,null,null,false));
          $this->addCampo(new dinheiro("PRODUTO_CUSTOATUAL","Custo Atual",
                         12,null,null,true,true,true,0,null,null,false));
          $this->addCampo(new dinheiro("PRODUTO_CUSTOMEDIO","Custo médio",
                         12,null,null,true,false,false,0,null,null,false));
          $this->addCampo(new float("PRODUTO_AC_ENT_QDE",
                    "Acumulado de Entradas (Quantidade)",
                    18,null,null,true,false,false,1,null,null,false));
          $this->addCampo(new float("PRODUTO_AC_ENT_VLR",
                    "Acumulado de Entradas (Valor)",
                    18,null,null,true,false,false,null,null,null,false));
          $this->addCampo(new float("PRODUTO_AC_SAIDAS",
                    "Acumulado de Saídas (Quantidade)",
                    18,null,null,true,false,false,null,null,null,false));
          $this->addCampo(new string("USUARIO_LOGIN","Usuário",
                    20,null,null,false,false,false,null,null,null,false,$_conn));
          $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",
                    14,null,null,false,false,false,null,null,null,false,$_conn));
     }

     /**
      * Ajustar os campos Usuário e data da
      * última alteração antes de efetivar a transação
      *
      * @return mixed
      */
     public function incluir() {
          $this->setUsuarioeData();
          return parent::incluir();
     }

     /**
      * Gera o registro inicial do Estoque por produto e local
      *
      * @param string $_prd
      * @param string $_local
      * @return mixed
      */
     public function inicializaEstoqueLocal($_prd,$_local) {
          $this->getCampo("PRODUTO_CODIGO")->setValor($_prd);
          $this->getCampo("LOCAL_CODIGO")->setValor($_local);
          foreach($this->filtrarCampos("incluir") as $_k=>$_campo) {
               if($_k!="PRODUTO_CODIGO"&&$_k!="LOCAL_CODIGO") {
```

```
                $_campo->setValor("0,000");
            }
        }
        return $this->incluir();
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function Alterar() {
        $this->setUsuarioeData();
        return parent::alterar();
    }

    /**
     * Calculo do Custo médio
     * Fórmula
     *   CM = (Acumulado_entradas_valor)/(Acumulado_entradas_qde)
     *
     */
    public function calculaCustoMedio() {
        $this->getCampo("PRODUTO_CUSTOMEDIO")->setValor(
                ($this->getCampo("PRODUTO_AC_ENT_QDE")->getValor())<>0
                    ? ($this->getCampo("PRODUTO_AC_ENT_VLR")->getValor()) /
                      ($this->getCampo("PRODUTO_AC_ENT_QDE")->getValor())
                    : 0
        );
    }

    /**
     * Gera os acumulados de Entrada
     *
     * @param float $_qde
     * @param float $_vlr
     */
    public function acumulaEntradas($_qde,$_vlr) {
        $this->getCampo("PRODUTO_AC_ENT_QDE")->setValor(
                $this->getCampo("PRODUTO_AC_ENT_QDE")->getValor()+$_qde);
        $this->getCampo("PRODUTO_AC_ENT_VLR")->setValor(
                $this->getCampo("PRODUTO_AC_ENT_VLR")->getValor()+$_vlr);
    }

    /**
     * Gera o acumulado de Saídas
     *
     * @param float $_qde
     */
    public function acumulaSaidas($_qde) {
        $this->getCampo("PRODUTO_AC_SAIDAS")->setValor(
                $this->getCampo("PRODUTO_AC_SAIDAS")->getValor()+$_qde);
    }

    /**
     * Acerta o saldo atual do produto
     *
     * @param float $_qde
     */
    protected function acertaSaldoAtual($_qde) {
        $this->getCampo("PRODUTO_ESTOQUE")->setValor(
                $this->getCampo("PRODUTO_ESTOQUE")->getValor() + $_qde);
    }
```

```
    /**
     * Atualiza o Custo Atual do produto
     *
     * @param float $_vlr
     */
    protected function acertaCustoAtual($_vlr) {
        $this->getCampo("PRODUTO_CUSTOATUAL")->setValor($_vlr);
    }

    /**
     * Realiza os acertos necessários no produto
     *
     * @param float $_qde
     * @param float $_vlr
     * @param tipomovimento $_tipo
     */
    public function acertaEstoque($_qde,$_vlr,tipomovimento $_tipo) {
        $this->acertaSaldoAtual(
            $_tipo->getCampo("TIPOMOV_TIPO")->getValor()=='E'
                ? $_qde
                : $_qde*-1);
        if($_tipo->getCampo("TIPOMOV_ESTORNO")->getValor()=='S') {
            $_qde *= -1;
            $_vlr *= -1;
        }
        if($_tipo->getCampo("TIPOMOV_CUSTOMEDIO")->getValor()=='S') {
            $this->acumulaEntradas($_qde,$_vlr);
            $this->calculaCustoMedio();
        } elseif($_tipo->getCampo("TIPOMOV_TIPO")->getValor()=='S') {
            $this->acumulaSaidas($_qde);
        }
        if($_tipo->getCampo("TIPOMOV_CUSTOATUAL")->getValor()=='S'&&$_vlr>0) {
            $this->acertaCustoAtual($_vlr/$_qde);
        }
        $_strSQL = "UPDATE {$this->_nome_tabela} SET " .
                   " PRODUTO_ESTOQUE=" .
                        $this->getCampo("PRODUTO_ESTOQUE")->getValor() .
                   ",PRODUTO_CUSTOMEDIO=" .
                        $this->getCampo("PRODUTO_CUSTOMEDIO")->getValor() .
                   ",PRODUTO_CUSTOATUAL=" .
                        $this->getCampo("PRODUTO_CUSTOATUAL")->getValor();
        if($_tipo->getCampo("TIPOMOV_TIPO")->getValor()=='E') {
            $_strSQL .= ",PRODUTO_AC_ENT_QDE=" .
                        $this->getCampo("PRODUTO_AC_ENT_QDE")->getValor() .
                      ",PRODUTO_AC_ENT_VLR=" .
                        $this->getCampo("PRODUTO_AC_ENT_VLR")->getValor();
        } else {
            $_strSQL .= ",PRODUTO_AC_SAIDAS=" .
                        $this->getCampo("PRODUTO_AC_SAIDAS")->getValor();
        }
        $_strSQL .= " WHERE PRODUTO_CODIGO=" .
                        $this->getCampo("PRODUTO_CODIGO")->toBD() .
                    " AND LOCAL_CODIGO=" .
                        $this->getCampo("LOCAL_CODIGO")->toBD();
        return ($_res=$this->_conn->executaSQL($_strSQL))===false
                    ? false
                    : $this->_conn->getNumRows();
    }
}
?>
```

Definição da classe historicomovimento

Os processos de movimentação de estoque, seja entrada, saída, estorno ou ajustes de inventário, possuem algo em comum; todos devem necessariamente acessar a tabela de histórico de movimentação de produtos. É essa entidade que centraliza todas as informações detalhadas da movimentação.

Precisamos disponibilizar métodos genéricos para o gerenciamento da movimentação, os quais são utilizados algumas vezes em sua versão original e em outras estendidos ou totalmente substituídos, dependendo da necessidade de cada classe.

A classe **historicomovimento** deve conter os campos necessários para descrever o histórico de movimentação e as regras para seu correto processamento. A lista seguinte descreve os campos da classe, bem como suas características:

1. PRODUTO_CODIGO

   Instância da classe **string**, especifica o código do produto que está sendo movimentado, sendo preenchido no formulário de movimentação de estoque. Além disso, esse campo faz parte da chave primária da tabela e terá o atributo comportamento form determinado como 'ajax', disponibilizando ao usuário, desta forma, um meio prático de busca do produto desejado (evitando o carregamento de uma lista muito grande de produtos).

2. LOCAL_CODIGO

   Instância da classe **string**, define o código do local que está sendo movimentado, sendo preenchido no formulário de movimentação de estoque. Além disso, esse campo faz parte da chave primária da tabela e será montado através de uma lista de locais, pois pressupõe-se que existam poucos locais (uma centena, no máximo). Assim temos o atributo comportamento form desse campo ajustado para 'select'. A lista de locais é preenchida no momento de geração do formulário.

3. HISTORICO_DATA

   Instância da classe **data**, indica a data da movimentação. O valor é definido pelo usuário no momento da inclusão do movimento, e nenhum movimento pode ser em data futura, ou seja, a data deve ser sempre menor ou igual ao dia atual. Esse campo faz parte da chave primária da tabela de movimentação.

4. HISTORICO_SEQUENCIA

   Instância da classe **inteiro** (que é uma especialização da classe **numero**), define a seqüência do movimento dentro da tabela de histórico da movimentação; seu conteúdo é gerado dinamicamente no momento anterior à inclusão do movimento. Esse campo não é exibido no formulário de inclusão e faz parte da chave primária da tabela.

5. TIPOMOV_CODIGO

   Instância da classe **string**, determina o código do tipo de movimento associado ao movimento atual, sendo seu conteúdo informado no formulário de inclusão de movimentos. O sistema disponibiliza uma lista fixa de tipos de movimentos permitidos e a lista deve ser sensível ao contexto, ou seja, somente podem ser gerados tipos de movimentos compatíveis com a operação em curso, o que obriga a especialização do método de busca em cada classe que foi herdada da classe **historicomovimento**.

6. TIPOMOV_TIPO

   Instância da classe **string**, define se o movimento é uma entrada ou uma saída. Esse campo tem seu valor especificado pelo tipo de movimento associado ao movimento. Criamos esse campo na tabela para tornar mais eficientes as buscas realizadas pelo sistema; caso contrário, seria preciso relacionar a tabela de tipo de movimentos sempre que fôssemos buscar um histórico de movimentação de determinado tipo (por exemplo, de entradas). No momento da geração do movimento, quando o método buscar o tipo de movimento, esse campo é automaticamente preenchido.

7. HISTORICO_QUANTIDADE

   Instância da classe **float** (que é uma especialização da classe **numero**), define a quantidade que está sendo movimentada. Seu valor deve ser sempre maior que zero, e pode ter seu valor máximo limitado ao estoque do produto e do local, conforme o tipo de movimento que é efetuado (tipicamente, quando fazemos uma saída e não desejamos um estoque negativo).

8. HISTORICO_DOCUMENTO

   Instância da classe **string**, indica o número ou qualquer outro valor que define o documento que originou a movimentação.

9. HISTORICO_VALOR_UNIT

   Instância da classe **dinheiro** (especialização da classe **numero**), define o valor unitário do movimento executado, e seu conteúdo pode ser informado no formulário quando for um movimento de entrada, ou ser definido dinamicamente (por exemplo, o custo médio do produto) quando for um movimento de saída.

10. HISTORICO_VALOR_TOTAL

    Instância da classe **dinheiro** (especialização da classe **numero**), define o valor total do movimento executado, e seu conteúdo pode ser informado no formulário quando for um movimento de entrada, ou ser definido dinamicamente (por exemplo, o custo médio do produto) quando for um movimento de saída.

11. HISTORICO_HISTORICO

    Instância da classe **string**, determina o histórico do movimento, podendo conter até 2.000 caracteres, e serve para descrever de forma detalhada o movimento. O atributo comportamento form desse campo deve ser ajustado para 'textarea' para que seja disponibilizada ao usuário uma forma mais prática para digitação dos dados.

12. HISTORICO_ESTORNO

    Instância da classe **string**, define se este é um movimento de estorno. Caso seja um movimento de estorno, o campo seguinte, HISTORICO_ESTORNO_SEQ, deve estar obrigatoriamente preenchido.

13. HISTORICO_ESTORNO_SEQ

    Instância da classe **inteiro** (uma especialização da classe **numero**), estabelece o seqüencial do movimento que foi estornado.

14. HISTORICO_ESTORNADO

    Instância da classe **string**, define se o movimento atual foi estornado ou não. Caso o movimento tenha sido estornado, ele não pode ser relacionado na lista de movimentos passíveis de estorno (o que ocasionaria uma discrepância no sistema).

Os campos HISTORICO_QUANTIDADE, HISTORICO_VALOR_UNIT e HISTORICO_VALOR_TOTAL estão diretamente relacionados, e sempre devemos ter a seguinte relação matemática:

```
HISTORICO_QUANTIDADE * HISTORICO_VALOR_UNIT = HISTORICO_VALOR_TOTAL
```

Além dos campos listados anteriormente e também dos campos-padrão USUARIO_LOGIN e DATA_ULTIMA_ALTERACAO, temos na classe alguns campos auxiliares utilizados para a exibição de informações ao usuário, tais como a descrição do produto, o estoque atual do produto, o custo atual do produto e o estoque do local selecionado. A lista seguinte mostra os campos extras da classe:

1. PRODUTO_DESC

   Instância da classe **string**, exibe a descrição do produto selecionado.

2. PRODUTO_ESTOQUE

   Instância da classe **float** (especialização da classe **numero**), mostra o estoque atual do produto.

3. PRODUTO_CUSTOMEDIO

   Instância da classe **dinheiro**, é utilizada para exibir o custo médio do produto. Esse custo pode ser utilizado para valorar o movimento em algumas situações, como, por exemplo, a saída de estoque.

4. LOCAL_ESTOQUE

Instância da classe **float**, exibe o estoque do produto no local selecionado, sendo utilizado na movimentação de saída de estoque.

A construção da classe exige a criação de outros métodos e também a alteração de alguns métodos existentes.

buscaProximaSequencia

Esse método é responsável pela busca da próxima seqüência disponível no arquivo de movimentação, lembrando que a seqüência é única.

```
/**
 * Busca a próxima seqüência disponível retornando 1 caso nenhuma exista
 *
 * @return integer
 */
protected function buscaProximaSequencia() {
     if($this->_conn->executaSQL(
               $this->montaSELECT("COALESCE(MAX(HISTORICO_SEQUENCIA),0)+1
               AS SEQ"))!==false) {
          $_dados = $this->_conn->proximo();
          return (int) $_dados["SEQ"];
     }
     return 1;
}
```

buscaTiposMovimentos

Esse método realiza a busca dos tipos de movimentos e deve ser estendido em cada classe dependente da classe **historicomovimento**, em que será indicado o tipo de movimento que deve ser retornado, pois o padrão nesse método é retornar todos os tipos de movimentos cadastrados.

```
/**
 * Preenche a lista de valores para Tipo de movimento
 *
 * @param string $_where
 */
protected function buscaTiposMovimentos($_where=null) {
     $_tm = new tipomovimento($this->_conn);
     $_tm->getConn()->executaSQL($_tm->montaSELECT("*",$_where));
     if($_tm->getConn()->getNumrows()<1) {
          die(utf8_encode("<span style='color:red;font-size:20px;"
                          "text-align:center;'>É necessário ter pelo menos " .
                          "1 Tipo de movimento cadastrado</span>"));
     }
     $_lc[] = Array("valor"=>"-","label"=>"selecione o Tipo de movimento");
     while($_tm->proximo()!==false) {
          $_lc[] = Array(
                         "valor"=>$_tm->getCampo('TIPOMOV_CODIGO')->getValor(),
                         "label"=>$_tm->getCampo('TIPOMOV_DESC')->getValor());
     }
     $this->getCampo("TIPOMOV_CODIGO")->setValor_fixo($_lc);
}
```

processaCampoFormulario

Nesse método vamos tratar os campos do formulário, conforme as necessidades de cada um. Alguns campos são definidos com o atributo 'DISABLED', informando que o campo não fica disponível para digitação, pelo menos no momento de carregamento do formulário. Outros campos recebem o atributo 'ONCHANGE' para que seja possível a execução de comandos adicionais sempre que o campo for alterado pelo usuário.

```
/**
 * Ajusta o comportamento dos campos do formulário
 *
 * @param string $_nome
 * @param tag $_campo
 * @param string $_metodo
 */
protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
     if(stripos($_metodo,'getformulario')!==false) {
          switch($_nome) {
               case 'PRODUTO_CODIGO':
                    $_campo->addAtributo(new atributo("ONCHANGE",
                                         "objBP.inicializa();"));
                    break;
               case 'PRODUTO_DESC':
               case 'PRODUTO_ESTOQUE':
               case 'LOCAL_ESTOQUE':
               case 'HISTORICO_DOCUMENTO':
               case 'HISTORICO_HISTORICO':
               case 'HISTORICO_DATA':
               case 'PRODUTO_CUSTOMEDIO':
                    $_campo->addAtributo(new atributo("DISABLED"));
                    break;
               case 'LOCAL_CODIGO':
                    $_campo->addAtributo(new atributo("ONCHANGE",
                                         "objBP.buscaLocal();"));
                    break;
               case 'TIPOMOV_CODIGO':
                    $_campo->addAtributo(new atributo("ONCHANGE",
                                         "objBP.validaTipomov();"));
                    break;
               case 'HISTORICO_QUANTIDADE':
               case 'HISTORICO_VALOR_UNIT':
               case 'HISTORICO_VALOR_TOTAL':
                    $_campo->addAtributo(new atributo("ONCHANGE",
                                         "recalculoValores(this)"));
                    $_campo->addAtributo(new atributo("DISABLED"));
                    break;
          }
     }
}
```

Os campos que são apenas informativos não são habilitados, já outros campos são habilitados ou não conforme a necessidade de cada operação de movimentação.

getOrdem

Nesse método determinamos a ordem de classificação da lista de movimentos retornados. Como a seqüência é única, utilizamos esse campo como forma de ordenação dos registros (em ordem decrescente).

```
/**
 * Define uma nova ordem na lista retornada
 *
 * @return string
 */
public function getOrdem() {
     return "HISTORICO_SEQUENCIA DESC";
}
```

incluir

O método incluir da classe **base** é estendido para que seja possível tratar situações específicas da classe de movimentação. As seguintes atividades precisam ser implementadas no método incluir, antes e depois da execução do mesmo método na classe **base**.

Antes de executar parent::incluir()

- Definir o valor do campo HISTORICO_SEQUENCIA pela chamada ao método buscaProximaSequencia() definido anteriormente.
- Incluir o campo HISTORICO_SEQUENCIA na lista de campos gerados na inclusão do movimento.
- Iniciar uma transação para o controle do processo todo, uma vez que teremos diversas interações com o banco de dados, alterando dados em várias tabelas.

Depois de parent::incluir()

- Instanciar as classes produto e tipomovimento.
- Buscar os registros correspondentes ao produto e tipo de movimento definido no movimento.
- Ajustar o estoque e demais dados do produto, tais como estoque atual, acumulados de entrada ou saídas conforme a situação, custo atual, custo médio e outros valores conforme o tipo de movimento.
- Buscar o registro correspondente ao produto no local de estoque.
- Caso o registro de produto e local de estoque não exista, gerar o seu registro de inicialização.

- Ajustar o estoque e demais dados relacionados ao produto e o local de estoque.
- Confirmar a transação iniciada anteriormente.

Caso qualquer um dos passos descritos falhe, o sistema desfaz a transação pelo comando ROLLBACK da classe **bancodados**.

getFormulario

Na geração do formulário para inclusão de movimentos é necessário um tratamento prévio, gerando algumas informações complementares necessárias ao controle do movimento. Precisamos gerar a lista de locais de estoque e também a lista dos tipos de movimentos relacionados ao movimento. A lista de locais será gerada dentro do próprio método e a lista de tipos de movimentos, no método *buscaTiposMovimentos* definido anteriormente, o qual gera apenas os tipos de movimentos relacionados ao processo de movimentação que está sendo executado. Devemos, ainda, incluir na geração do formulário o comando javascript necessário para o carregamento do arquivo com os comandos javascript necessários.

```
public function getFormulario($_funcao='INC') {
     $_extra = "";
     // Buscar Locais
     $_local = new localestoque($this->_conn);
     $_local->Buscar();
     if($_local->getConn()->getNumrows()<1) {
          die(utf8_encode("<span style='color:red;font-size:20px;" .
                          "text-align:center;'>É necessário ter pelo menos " .
                          "1 Local cadastrado</span>"));
     }
     $_lc[] = Array("valor"=>"-","label"=>"selecione o Local");
     while($_local->proximo()!==false) {
          $_lc[] = Array(
                    "valor"=>$_local->getCampo('LOCAL_CODIGO')->getValor(),
                    "label"=>$_local->getCampo('LOCAL_DESC')->getValor());
     }
     $this->getCampo("LOCAL_CODIGO")->setValor_fixo($_lc);
     // Tipos de movimentos
     $this->buscaTiposMovimentos();
     // Carregar javascript especifico
     $_sc = new tag(new tipotag("SCRIPT"),null,
                          "LoadJS('estoque/javascript/movimento.js');");
     $_extra = $_sc->toHTML();
     $this->_num_colunas_form = 2;
     return $_extra . parent::getFormulario($_funcao);
}
```

O arquivo com os códigos javascript é descrito no próximo tópico.

Finalmente a classe **historicomovimento** deve então ser gravada no arquivo *classe_historicomovimento.inc* dentro do diretório *estoque/classes/*.

Lista 2: classe_historicomovimento.inc

```
/**
 * Classe historicomovimento
 */

class historicomovimento extends base {
    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'historicomovimento';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("PRODUTO_CODIGO","Produto",
                    20,null,null,true,false,true,1,null,null,true,$_conn));
        $this->addCampo(new string("PRODUTO_DESC","Descrição",
                    60,null,null,true,false,false,1,null,null,false,$_conn));
        $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque Atual",
                    12,null,null,true,false,false,null,null,null,false));
        $this->addCampo(new float("PRODUTO_CUSTOMEDIO","Custo Médio",
                    12,null,null,true,false,false,null,null,null,false));
        $this->addCampo(new string("LOCAL_CODIGO","Local",
                    10,null,null,true,false,true,1,null,null,true,$_conn));
        $this->addCampo(new float("LOCAL_ESTOQUE","Estoque do Local",
                    12,null,null,true,false,false,null,null,null,false));
        $this->addCampo(new inteiro("HISTORICO_SEQUENCIA","Sequência",
                    15,null,null,false,false,true,1,null,null,true));
        $this->addCampo(new string("TIPOMOV_CODIGO","Tipo de movimento",
                    10,null,null,true,false,false,1,null,null,false,$_conn));
        $this->addCampo(new data("HISTORICO_DATA","Data",
                    10,null,null,true,true,true,null,'NOW',null,true,$_conn));
        $this->addCampo(new string("TIPOMOV_TIPO","Entrada/Saída",
                    1,null,null,false,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("HISTORICO_DOCUMENTO","Documento",
                    20,null,null,true,true,true,1,null,null,false,$_conn));
        $this->addCampo(new float("HISTORICO_QUANTIDADE","Quantidade",
                    10,null,null,true,false,true,0.001,null,null,false));
        $this->addCampo(new dinheiro("HISTORICO_VALOR_UNIT","Valor unitário",
                    12,null,null,true,true,true,0,null,null,false));
        $this->addCampo(new dinheiro("HISTORICO_VALOR_TOTAL","Total",
                12,null,null,true,true,true,0,null,null,false));
        $this->addCampo(new string("HISTORICO_HISTORICO","Histórico",
                2000,null,null,true,false,false,1,null,null,false,$_conn));
        $this->addCampo(new string("HISTORICO_ESTORNO","Estorno?",
                1,null,null,false,false,true,1,null,null,false,$_conn));
        $this->addCampo(new inteiro("HISTORICO_ESTORNO_SEQ",
                "Sequência Estornada",
                15,null,null,false,false,false,1,null,null,false));
        $this->addCampo(new string("HISTORICO_ESTORNADO","Estornado?",
                1,null,null,false,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("USUARIO_LOGIN","Usuário",
                20,null,null,false,false,false,null,null,null,false,$_conn));
        $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",
                14,null,null,false,false,false,null,null,null,false,$_conn));
        $this->getCampo("PRODUTO_CODIGO")->setComportamento_form("ajax");
        $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                    "estoque/classes/classe_produto.inc");
        $this->getCampo("PRODUTO_CODIGO")->setUpdateElement(
                    "function(li) {atualizaFormulario(li);objBP.buscar();}");
        $this->getCampo("LOCAL_CODIGO")->setComportamento_form("select");
        $this->getCampo("TIPOMOV_CODIGO")->setComportamento_form("select");
        $this->getCampo("HISTORICO_HISTORICO")->setComportamento_form(
                    "textarea");
        $this->_exibe_opcoes = false;
```

```
        $this->_exibe_botao_incluir = true;
    }

    /**
     * Busca a próxima seqüência disponível retornando 1 caso nenhuma exista
     *
     * @return integer
     */
    protected function buscaProximaSequencia() {
        if($this->_conn->executaSQL($this->montaSELECT(
                "COALESCE(MAX(HISTORICO_SEQUENCIA),0)+1 AS SEQ"))!==false) {
            $_dados = $this->_conn->proximo();
            return (int) $_dados["SEQ"];
        }
        return 1;
    }

    /**
     * Preenche a lista de valores para Tipo de movimento
     *
     * @param string $_where
     */
    protected function buscaTiposMovimentos($_where=null) {
        $_tm = new tipomovimento($this->_conn);
        $_tm->getConn()->executaSQL($_tm->montaSELECT("*",$_where));
        if($_tm->getConn()->getNumrows()<1) {
            die(utf8_encode("<span style='color:red;font-size:20px;" .
                            "text-align:center;'>É necessário ter pelo menos " .
                            "1 Tipo de movimento cadastrado</span>"));
        }
        $_lc[] = Array("valor"=>"-","label"=>"selecione o Tipo de movimento");
        while($_tm->proximo()!==false) {
            $_lc[] = Array(
                            "valor"=>$_tm->getCampo('TIPOMOV_CODIGO')->getValor(),
                            "label"=>$_tm->getCampo('TIPOMOV_DESC')->getValor());
        }
        $this->getCampo("TIPOMOV_CODIGO")->setValor_fixo($_lc);
    }

    /**
     * Ajusta o comportamento dos campos do formulário
     *
     * @param string $_nome
     * @param tag $_campo
     * @param string $_metodo
     */
    protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
        if(stripos($_metodo,'getformulario')!==false) {
            switch($_nome) {
                case 'PRODUTO_CODIGO':
                    $_campo->addAtributo(new atributo("ONCHANGE",
                                                "objBP.inicializa();"));
                    break;
                case 'PRODUTO_DESC':
                case 'PRODUTO_ESTOQUE':
                case 'LOCAL_ESTOQUE':
                case 'HISTORICO_DOCUMENTO':
                case 'HISTORICO_HISTORICO':
                case 'HISTORICO_DATA':
                case 'PRODUTO_CUSTOMEDIO':
                    $_campo->addAtributo(new atributo("DISABLED"));
                    break;
                case 'LOCAL_CODIGO':
                    $_campo->addAtributo(new atributo("ONCHANGE",
```

```
                                        "objBP.buscaLocal();"));
                    break;
                case 'TIPOMOV_CODIGO':
                    $_campo->addAtributo(new atributo("ONCHANGE",
                                        "objBP.validaTipomov();"));
                    break;
                case 'HISTORICO_QUANTIDADE':
                case 'HISTORICO_VALOR_UNIT':
                case 'HISTORICO_VALOR_TOTAL':
                    $_campo->addAtributo(new atributo("ONCHANGE",
                                        "recalculoValores(this)"));
                    $_campo->addAtributo(new atributo("DISABLED"));
                    break;
            }
        }
    }

    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(HISTORICO_SEQUENCIA) AS CODIGO FROM " .
                "{$this->_nome_tabela}";
    }

    /**
     * Define uma nova ordem na lista retornada
     *
     * @return string
     */
    public function getOrdem() {
        return "HISTORICO_SEQUENCIA DESC";
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function incluir() {
        $this->getCampo("HISTORICO_SEQUENCIA")->setValor(
                                        $this->buscaProximaSequencia());
        $this->getCampo("HISTORICO_SEQUENCIA")->setIncluir(true);
        $this->setUsuarioeData();
        $this->_conn->startTransaction();
        if(($_res=parent::incluir())!==false) {
            // Acertar os Saldos
            $_prd = new produto($this->_conn);
            $_tpm = new tipomovimento($this->_conn);
            $_tpm->Buscar(Array('TIPOMOV_CODIGO'=>
                                $this->getCampo("TIPOMOV_CODIGO")->getValor()));
            $_tpm->proximo();
            $_prd->Buscar(Array('PRODUTO_CODIGO'=>
                                $this->getCampo("PRODUTO_CODIGO")->getValor()));
            $_prd->proximo();
            if(($_rp=$_prd->acertaEstoque(
                    $this->getCampo("HISTORICO_QUANTIDADE")->toBD(),
                    $this->getCampo("HISTORICO_VALOR_TOTAL")->toBD(),
                    $_tpm))!==false) {
                // Saldo do Local
```

```
                $_lce = new estoquelocal($this->_conn);
                $_lce->Buscar(Array(
                          'PRODUTO_CODIGO'=>
                               $this->getCampo("PRODUTO_CODIGO")->getValor(),
                          'LOCAL_CODIGO'=>
                               $this->getCampo('LOCAL_CODIGO')->getValor()));
                if($_lce->getConn()->getNumRows()<=0) {
                     // Criar o novo local
                     $_re=$_lce->inicializaEstoqueLocal(
                               $this->getCampo('PRODUTO_CODIGO')->getValor(),
                               $this->getCampo('LOCAL_CODIGO')->getValor());
                } else {
                     $_lce->proximo();
                     $_re = true;
                }
                if($_re!==false) {
                     $_re=$_lce->acertaEstoque(
                               $this->getCampo("HISTORICO_QUANTIDADE")->toBD(),
                               $this->getCampo("HISTORICO_VALOR_TOTAL")->toBD(),
                               $_tpm);
                }
                if($_re!==false) {
                     $this->_conn->commit();
                } else {
                     $this->_conn->ROLLBACK();
                     return $_re;
                }
           } else {
                $this->_conn->ROLLBACK();
                return $_rp;
           }
     } else {
          $this->_conn->ROLLBACK();
     }
     return $_res;
}

/**
 * Não existe alteração no movimento
 * Com exceção do Estorno
 *
 * @return mixed
 */
public function Alterar() {
     return false;
}

public function getFormulario($_funcao='INC') {
     $_extra = "";
     // Buscar Locais
     $_local = new localestoque($this->_conn);
     $_local->Buscar();
     if($_local->getConn()->getNumrows()<1) {
          die(utf8_encode("<span style='color:red;font-size:20px;" .
                          "text-align:center;'>É necessário ter pelo menos " .
                          "1 Local cadastrado</span>"));
     }
     $_lc[] = Array("valor"=>"-","label"=>"selecione o Local");
     while($_local->proximo()!==false) {
          $_lc[] = Array(
                    "valor"=>$_local->getCampo('LOCAL_CODIGO')->getValor(),
                    "label"=>$_local->getCampo('LOCAL_DESC')->getValor());
     }
     $this->getCampo("LOCAL_CODIGO")->setValor_fixo($_lc);
```

```
        // Tipos de movimentos
        $this->buscaTiposMovimentos();
        // Carregar javascript especifico
        $_sc = new tag(new tipotag("SCRIPT"),null,
                                "LoadJS('estoque/javascript/movimento.js');");
        $_extra = $_sc->toHTML();
        $this->_num_colunas_form = 2;
        return $_extra . parent::getFormulario($_funcao);
    }

}
```

Comandos javascript relacionados à movimentação de estoque

A movimentação de estoque exige uma série de comandos javascript que são utilizados para o correto controle das operações de entrada e de saída e também das operações de estorno e inventário.

Temos uma classe principal que executa a maioria dos comandos javascript necessários, além de mais algumas funções avulsas para o controle de determinadas operações.

A classe javascript será chamada de **movimento** e contém métodos para realizar as seguintes tarefas:

- Busca dinâmica de produtos
- Ajustes no formulário conforme o produto selecionado
- Busca do local de estoque
- Ajustes extras no formulário conforme o local selecionado
- Habilitação de campos conforme o local selecionado

A classe será instanciada como a variável global objBP, a qual é referenciada no processamento do formulário de movimentação especificado anteriormente.

Além dessa classe precisamos de duas funções extras para poder realizar a formatação de números e o recálculo dos campos HISTORICO_VALOR_UNIT e HISTORICO_VALOR_TOTAL.

O arquivo *moviment.js* que contém o código javascript deve ser salvo no diretório *estoque/javascript*.

Lista 3: movimento.js

```
function movimento() {};
movimento.prototype = {
FOk: false,
FEntrada: true,
inicializa:   function(){
                this.FOk=false;
                document.getElementById('PRODUTO_DESC').value='';
```

```
                document.getElementById('PRODUTO_ESTOQUE').value='';
                document.getElementById('PRODUTO_CUSTOMEDIO').value='';
                document.getElementById('LOCAL_CODIGO').selectedIndex=0;
                document.getElementById('TIPOMOV_CODIGO').selectedIndex=0;
            },
buscar:     function() {
                var params='classe=produto&metodo=buscarXML&' +
                           'arquivo_classe=estoque/classes/classe_produto.inc';
                params += '&campos=PRODUTO_CODIGO&valores=' +
                          document.getElementById('PRODUTO_CODIGO').value;
                ObjProcAjax.runPostXML('executa_busca_ajax.php5',
                                       params,objBP.processa);
            },
processa:   function(xml){
                var l = xml.childNodes[0].childNodes[0];
                var vlr = l.getElementsByTagName('PRODUTO_DESC')[0];
                document.getElementById('PRODUTO_DESC').value =
                              (vlr.hasChildNodes()
                                   ? vlr.firstChild.nodeValue
                                   : '');
                var vlr = l.getElementsByTagName('PRODUTO_ESTOQUE')[0];
                document.getElementById('PRODUTO_ESTOQUE').value =
                              (vlr.hasChildNodes()
                                   ? vlr.firstChild.nodeValue
                                   : '0,000');
                var vlr = l.getElementsByTagName('PRODUTO_CUSTOMEDIO')[0];
                document.getElementById('PRODUTO_CUSTOMEDIO').value =
                              (vlr.hasChildNodes()
                                   ? vlr.firstChild.nodeValue
                                   : '0,00');
                objBP.FOk=true;
                // Se for uma saida, colocar valor_unitario=preco_atual ou
                // custo_atual????
                if(this.FEntrada!==true) {
                    document.getElementById('HISTORICO_VALOR_UNIT').value =
                        document.getElementById('PRODUTO_CUSTOMEDIO').value;
                    document.getElementById('CUSTO_SAIDA').value =
                        document.getElementById('PRODUTO_CUSTOMEDIO').value;
                };
                document.getElementById('LOCAL_CODIGO').focus();
            },
produtoOk:  function() {
                if(this.FOk!==true){
                    alert('Selecione um produto primeiro');
                    document.getElementById('PRODUTO_CODIGO').focus();
                    return false;
                } else {
                    return true;
                }
            },
buscaLocal: function() {
                if(this.produtoOk()!==true){
                    document.getElementById('LOCAL_CODIGO').selectedIndex=0;
                } else {
                    var l = document.getElementById('LOCAL_CODIGO');
                    var local = l.options[l.selectedIndex].value;
                    if(local!='-') {
                        var params='classe=estoquelocal&metodo=buscarXML&' +
                                   'arquivo_classe=estoque/classes/'  +
                                   'classe_estoquelocal.inc';
                        params += '&campos=PRODUTO_CODIGO,' +
                            'LOCAL_CODIGO&valores=' +
                            document.getElementById('PRODUTO_CODIGO').value;
                        params += ','+local;
```

```
                    ObjProcAjax.runPostXML('executa_busca_ajax.php5',
                                                params,objBP.EstoqueLocal);
                } else {
                    document.getElementById('LOCAL_ESTOQUE').value = '-';
                }
            };
            this.habilitaCampos();
        },
validaTipomov:function() {
            if(this.produtoOk()!==true){
                document.getElementById('TIPOMOV_CODIGO').selectedIndex=0;
            };
            this.habilitaCampos();
        },
EstoqueLocal: function(xml) {
            var est = '0,000';
            if(xml.hasChildNodes()&&xml.childNodes[0].hasChildNodes()) {
                var l = xml.childNodes[0].childNodes[0];
                var vlr = l.getElementsByTagName('PRODUTO_ESTOQUE')[0];
                est =  (vlr.hasChildNodes()
                          ? vlr.firstChild.nodeValue
                          : '');
            }
            document.getElementById('LOCAL_ESTOQUE').value = est;
        },
habilitaCampos:
    function() {
        var enabled = false;
        var l= document.getElementById('LOCAL_CODIGO');
        var t= document.getElementById('TIPOMOV_CODIGO');
        if(l.selectedIndex>0&&t.selectedIndex>0) {
            enabled = true;
        };
        document.getElementById('HISTORICO_DOCUMENTO').disabled=!enabled;
        document.getElementById('HISTORICO_QUANTIDADE').disabled=!enabled;
        if(this.FEntrada===true) {
            document.getElementById('HISTORICO_VALOR_UNIT').disabled=!enabled;
            document.getElementById('HISTORICO_VALOR_TOTAL').disabled=!enabled;
        };
        document.getElementById('HISTORICO_HISTORICO').disabled = !enabled;
        document.getElementById('HISTORICO_DATA').disabled = !enabled;
        if(!enabled) {
            document.getElementById('HISTORICO_DOCUMENTO').value = '';
            document.getElementById('HISTORICO_QUANTIDADE').value = '';
            if(this.FEntrada===true) {
                document.getElementById('HISTORICO_VALOR_UNIT').value = '';
                document.getElementById('HISTORICO_VALOR_TOTAL').value = '';
            };
            document.getElementById('HISTORICO_HISTORICO').innerHTML = '';
            document.getElementById('HISTORICO_DATA').value = '';
        } else {
            document.getElementById('HISTORICO_DATA').focus();
        };
    }
};
var objBP = new movimento();

function formataNumero(n) {
    var cent = n%1;
    n -= cent;
    cent = Math.round(cent*100);
    return n + ',' + (cent<10 ? '0' : '')+ cent;
};
```

```
// Recalculo de Valores
function recalculoValores(obj) {
     var q = document.getElementById('HISTORICO_QUANTIDADE');
     var v = document.getElementById('HISTORICO_VALOR_UNIT');
     var t = document.getElementById('HISTORICO_VALOR_TOTAL');
     var qde = parseFloat(q.value.replace(/,/gi,'.'));
     var vun = parseFloat(v.value.replace(/,/gi,'.'));;
     var vtt = parseFloat(t.value.replace(/,/gi,'.'));
     if(isNaN(vun)) { vun = 0};
     if(isNaN(vtt)) { vtt = 0};
     if(obj.id==='HISTORICO_VALOR_UNIT'||obj.id==='HISTORICO_QUANTIDADE') {
          t.value = formataNumero(qde*vun);

     } else {
          v.value = formataNumero(vtt/qde);
     }
};
```

Classe entradas

Com a classe de movimentação definida, podemos criar a classe especializada no processo de entrada de estoque. Essa classe define o filtro fixo para a busca de registros na tabela associada, além de indicar o critério de busca dos tipos de movimentos, e por último ajusta o valor do atributo FEntrada na classe javascript movimento.

O filtro fixo, definido pelo método *filtroFixo()*, indica que somente os registros de entrada, ou seja, cujo campo TIPOMOV_TIPO seja igual a 'E', devem ser listados.

O critério fixado para a busca de tipos de movimentos é TIPOMOV_TIPO = 'E' e TIPOMOV_ESTORNO = 'N', indicando que somente os tipos de entrada e que não sejam estorno serão listados no formulário de entrada de estoque.

O atributo FEntrada é definido como 'E', indicando que o movimento é uma entrada de estoque. Esse atributo é utilizado na classe javascript movimento em alguns processos, alterando o comportamento caso o movimento seja uma entrada ou saída (veja a classe movimento para maiores detalhes).

Além desses métodos, há apenas a definição no construtor da classe dos valores mínimos aceitos para os campos de nome HISTORICO_VALOR_UNIT e HISTORICO_VALOR_TOTAL, uma vez que é obrigatória a digitação de um valor para os movimentos de entrada de estoque.

A classe **entradas** é construída dentro do arquivo *classe_historicomovimento.inc*, uma vez que não faz sentido sua definição separada da classe principal.

Lista 4: classe_historicomovimento.inc

```
/**
 * Classe para movimentação de entradas
 *
 */
class entradas extends historicomovimento {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->getCampo("HISTORICO_VALOR_UNIT")->setMinimo(0.01);
          $this->getCampo("HISTORICO_VALOR_TOTAL")->setMinimo(0.01);
     }

     /**
      * Seleciona apenas os tipos de movimento de entrada
      *
      * @param string $_where
      */
     protected function buscaTiposMovimentos($_where=null) {
          parent::buscaTiposMovimentos("TIPOMOV_TIPO='E' AND TIPOMOV_ESTORNO='N'");
     }

     protected function FiltroFixo() {
          return "TIPOMOV_TIPO='E'";
     }

     /**
      * Informa a classe javascript que é uma Entrada
      *
      * @param string $_funcao
      * @return mixed
      */
     public function getFormulario($_funcao='INC') {
          $_sc = new tag(new tipotag("SCRIPT"),null,"objBP.FEntrada=true;");
          return parent::getFormulario($_funcao) . $_sc->toHTML();
     }

}
```

Para que o framework encontre o arquivo que contém a classe, devemos alterar o método *SISTEMA_autoload* que foi definido em *autoload.inc*. Quando o método receber a classe **entradas** como parâmetro, devemos em seguida buscar o arquivo *classe_historicomovimento.inc* e não o arquivo *classe_entradas.inc*.

```
function SISTEMA_autoload($_classe) {
     switch($_classe) {
          case 'entradas':   $_classe = 'historicomovimento';
                                   break;
     }
     if(file_exists("classe_{$_classe}.inc")) {
          include_once("classe_{$_classe}.inc");
          return true;
     } elseif(file_exists("estoque/classes/classe_{$_classe}.inc")) {
          include_once("estoque/classes/classe_{$_classe}.inc");
          return true;
     } elseif(file_exists("../estoque/classes/classe_{$_classe}.inc")) {
          include_once("../estoque/classes/classe_{$_classe}.inc");
          return true;
     } elseif(file_exists("classes/classe_{$_classe}.inc")) {
          include_once("classe_{$_classe}.inc");
          return true;
```

```
        } else {
            die("<span style='color:red;font-size:20px;" .
                "text-align:center;'>Classe {$_classe} não Existe...</span>");
        }
    }
```

Criação dos tipos de movimentos

Os tipos de movimentos mostrados anteriormente devem ser cadastrados no sistema, na opção Tipos de movimentos do menu cadastro. Cada tipo de movimento deve ter um código identificador, uma descrição condizente e os parâmetros de custo atual e custo médio devidamente ajustados. Os tipos descritos representam o mínimo necessário para que o processo seja corretamente executado, e caso seja necessário, devemos criar tipos adicionais que representem outras situações do dia-a-dia da empresa.

Os movimentos de entrada de nota fiscal de compra e transferência de outro local devem afetar tanto o custo atual quanto o custo médio do produto.

O movimento de devolução de vendas não deve afetar o custo médio nem o custo atual.

O movimento de ajuste manual de estoque deve possuir variações para cobrir todas as possibilidades, ou seja, devemos ter uma variação para afetar tanto o custo médio como o custo atual, outra variação que afete apenas o custo atual, uma terceira que afete apenas o custo médio e uma última variação que não afete os custos médio e atual.

A tabela seguinte mostra os tipos de movimentos relacionados com a entrada de estoque (vamos estudar os estornos de movimentos em outro capítulo).

Código	Descrição	Afeta Custo médio	Afeta Custo atual
ENF	Entrada por NotaFiscal	S	S
ETF	Entrada por Transferência	S	S
EDV	Entrada por devolução de vendas	N	N
EAJSS	Entrada por Ajuste de Estoque	S	S
EAJSN	Entrada por Ajuste de Estoque (CM)	S	N
EAJNS	Entrada por Ajuste de Estoque (CA)	N	S
EAJNN	Entrada por Ajuste de Estoque (S/C)	N	N

Devemos cadastrar estes tipos de movimentos no sistema de controle de estoque.

A Figura 5.1 mostra a lista de tipos de movimentos de entrada cadastrados no sistema.

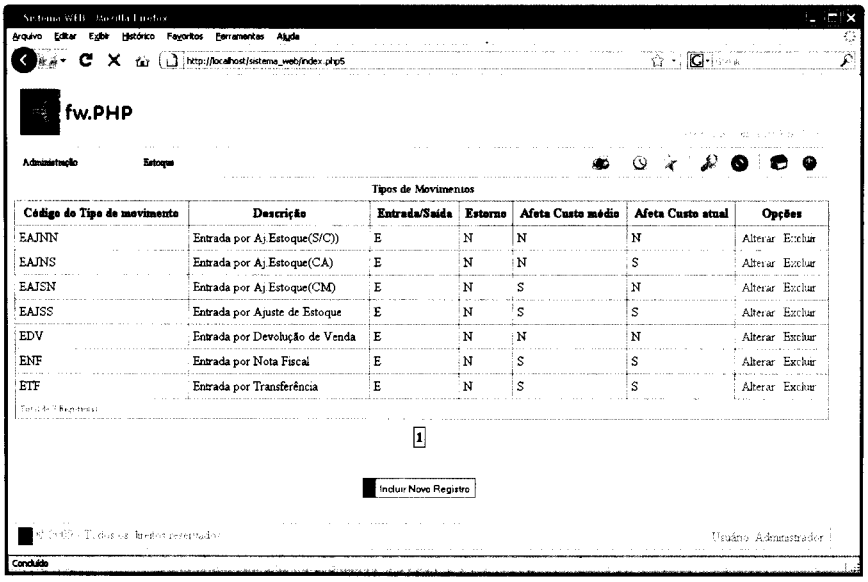


Figura 5.1

Criação de um local de estoque

Não é possível realizar nenhum lançamento no sistema sem que exista pelo menos um local de estoque cadastrado.

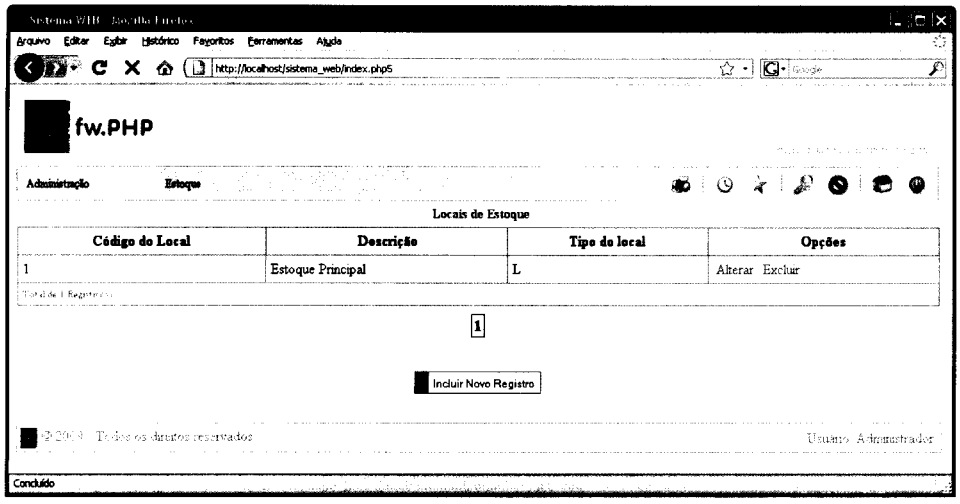


Figura 5.2

Como precisamos testar o sistema e garantir que está tudo funcionando corretamente, vamos cadastrar apenas um local de estoque, que será utilizado como local-padrão, o qual chamaremos de 'Estoque principal'. Toda a movimentação será neste local-padrão. Caso deseje testar o gerenciamento em vários locais, basta criar os locais adicionais desejados e realizar testes de movimentação neles.

O local cadastrado é do tipo 'Loja'. A figura anterior mostrou a tela com o local cadastrado.

Definição do programa de entrada de estoque

Assim como os demais programas construídos até o momento, o programa para gerenciamento de entradas de estoque é bem simples, exigindo apenas umas poucas linhas de código. Precisamos informar o arquivo que contém a classe e o nome da classe, além de um título para a página gerada.

Lista 5: mov_entradas.php5

```
<?php
/**
 * Movimentação, Entradas de estoque
 *
 */
$_classe = Array("arquivo"=>"estoque/classes/classe_historicomovimento.inc",
                 "nome"=>"entradas");
$_FTitulo = "Movimentação - Entradas";
return include_once("../instanciaclasse.php5");
?>
```

Diferentemente dos demais programas, neste o nome do arquivo não é similar ao nome da classe, uma vez que a classe de entradas está dentro do mesmo arquivo que contém a classe principal **historicomovimento**.

Geração do menu de movimentação e de entradas

A última tarefa antes de executar o processo de entrada de estoque é gerar o menu de movimentação e o menu de entrada de estoques. O menu de movimentação deve estar ligado ao menu principal do sistema, ou seja, ao menu Estoque. Já o menu de entradas é um item do menu de movimentação. A Figura 5.3 mostra o menu criado.

Com o menu criado, podemos acessar a página de entrada de estoque (primeiro você deve sair do sistema e entrar novamente, para que o novo menu seja carregado).

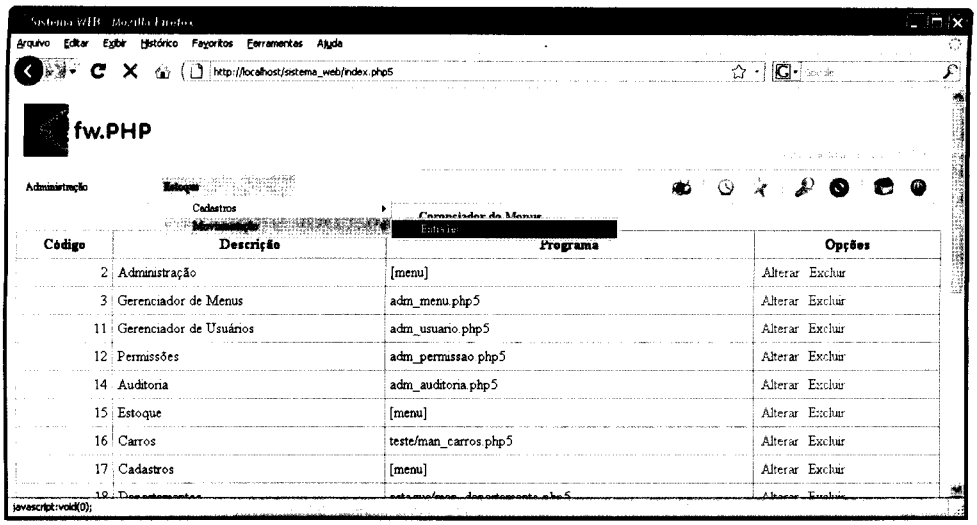


Figura 5.3

A tela de entrada de mercadoria, assim como a de saída e demais telas de movimentação, é iniciada apenas com os campos PRODUTO_CODIGO, LOCAL_CODIGO e TIPOMOV_CODIGO habilitados, Figura 5.4. Os demais estão desabilitados. Mesmo habilitados, os campos LOCAL_CODIGO e TIPOMOV_CODIGO somente podem ser acessados quando o produto for selecionado corretamente.

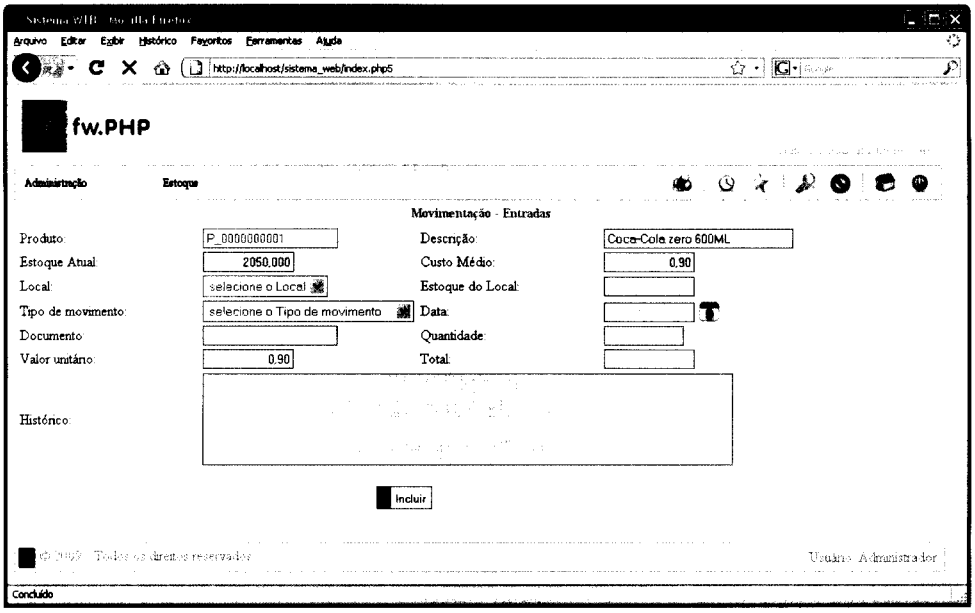


Figura 5.4

Uma vez que o produto, o local e o tipo de movimento tenham sido selecionados, o sistema habilita os demais campos do formulário (em outros processos, parte desses campos continua desabilitada). A Figura 5.5 mostra o formulário com os demais campos do processo de entrada habilitados.

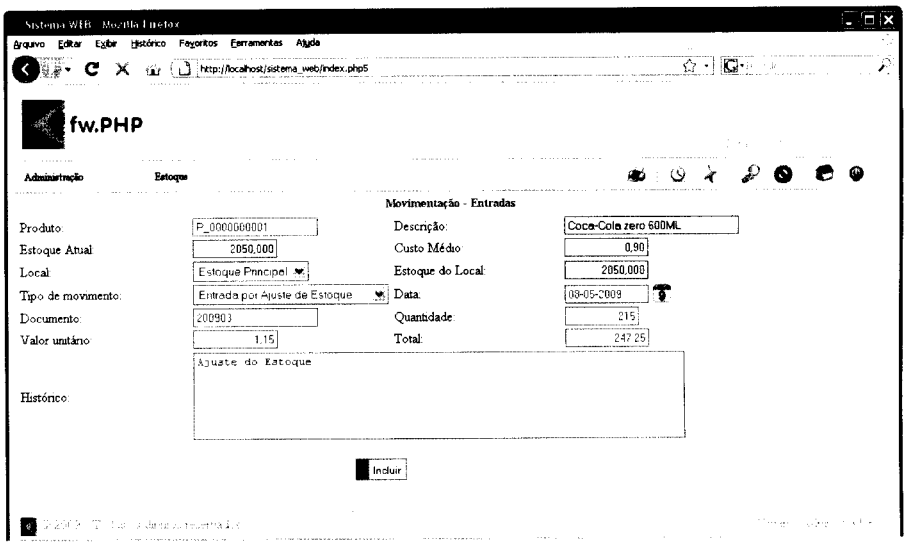


Figura 5.5

Com todos os dados digitados corretamente (com exceção do histórico, todos os campos são obrigatórios e possuem restrições de tamanho mínimo e valor), ao clicar no botão 'incluir', o movimento é gerado e as tabelas de produtos e estoques por local são devidamente atualizadas. A Figura 5.6 exibe a lista de movimentos após uma entrada ter sido efetuada.

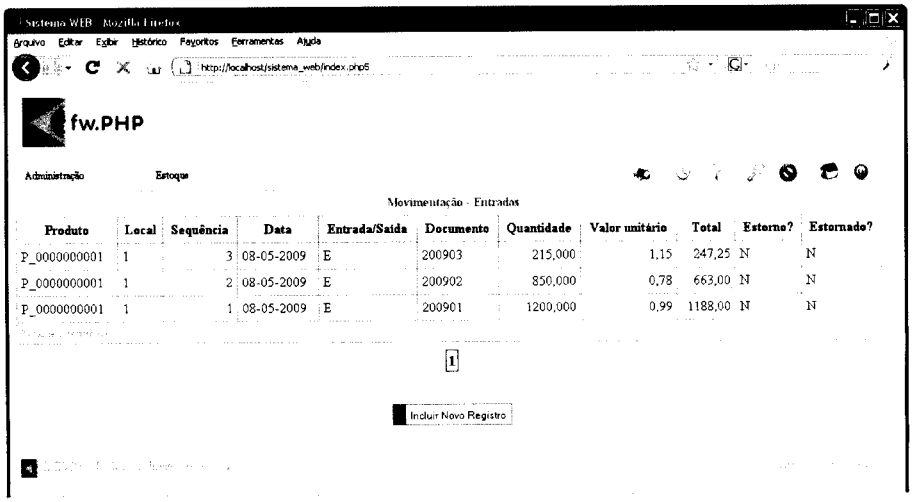
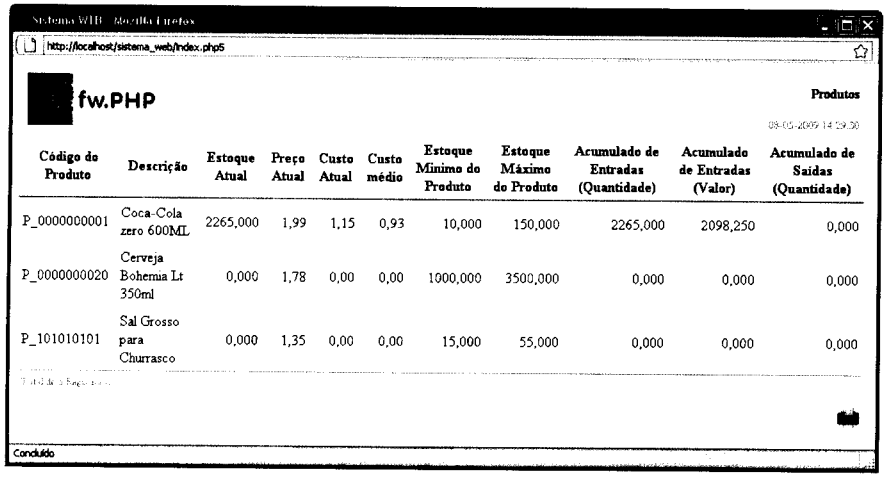


Figura 5.6

A Figura 5.7 apresenta a listagem dos produtos cadastrados no sistema, atualmente três, e os valores dos estoques e acumulados.



Código do Produto	Descrição	Estoque Atual	Preço Atual	Custo Atual	Custo médio	Estoque Mínimo do Produto	Estoque Máximo do Produto	Acumulado de Entradas (Quantidade)	Acumulado de Entradas (Valor)	Acumulado de Saídas (Quantidade)
P_0000000001	Coca-Cola zero 600ML	2265,000	1,99	1,15	0,93	10,000	150,000	2265,000	2098,250	0,000
P_0000000020	Cerveja Bohemia Lt 350ml	0,000	1,78	0,00	0,00	1000,000	3500,000	0,000	0,000	0,000
P_101010101	Sal Grosso para Churrasco	0,000	1,35	0,00	0,00	15,000	55,000	0,000	0,000	0,000

Figura 5.7

Capítulo 6

Saída de Estoque

A movimentação de estoque requer lançamentos de entradas e saídas, o primeiro somando quantidades ao estoque e o segundo, subtraindo. Toda a movimentação gira em torno destes dois tipos de lançamento, seja uma venda, uma compra, ajustes de inventário, estorno etc. No capítulo anterior construímos a base do módulo de movimentação de estoque e criamos o processo de entrada de estoques. Agora vamos tratar a contraparte da entrada, a saída de estoque.

O movimento de saída de estoque pode ser gerado a partir de várias operações, por exemplo:

- Saída por vendas
- Transferência para outro local de estoque
- Quebra (produtos danificados)
- Ajustes manuais

Precisamos de tipos de movimentos que descrevam estas operações, mas sabemos a princípio que nenhuma delas deve afetar o cálculo do custo médio e do custo atual do produto (à frente constam algumas operações que afetam o cálculo).

O desenvolvimento da operação de saída de estoque envolve a construção de uma classe apropriada, especialização da classe principal **historicomovimento**, além de alguns ajustes gerais no sistema. As seguintes tarefas devem ser desenvolvidas:

- Definição da classe de saídas;
- Javascript de movimentação;
- Criação dos tipos de movimentos relacionados ao processo de saídas;
- Definição do programa de saída de estoque;
- Geração do menu para saída de estoque.

A classe de saída de estoque é definida tomando por base a classe principal de movimentação de estoque **historicomovimento**.

Sem os tipos de movimentos criados não conseguimos realizar nenhuma movimentação de saída de estoque no sistema.

Definição da classe de saídas de estoque

Assim como a classe de entradas de estoque, a classe de saídas é uma extensão da classe principal de movimentação **historicomovimento**. Precisamos alterar alguns métodos, definir outros e inclusive criar um campo para ajudar no controle do movimento de saídas.

A movimentação de saídas não altera o custo médio nem o custo atual do produto, portanto não é necessário que o usuário informe o valor unitário ou o valor total da movimentação; em vez disso, devemos valorizar o movimento de saída conforme o custo médio atual do produto. Desta forma não precisamos habilitar esses dois campos no formulário de movimentação.

A ação de habilitar os campos do formulário, após o usuário informar o produto, o local de estoque e o tipo de movimento, é realizada na classe javascript movimento. Precisamos ter um mecanismo que informe à classe que o movimento executado no momento é de saída, indicando assim que esses campos não devem ser habilitados. Essa informação é passada à classe através do atributo FEntrada, que deve receber o valor falso (false) quando o movimento for uma saída de estoque.

A operação de saída tem por objetivo reduzir o estoque atual do produto, bem como o estoque do local selecionado. Essa operação pode, caso não sejam tomadas medidas preventivas, tornar o estoque atual do local ou do produto negativo, o que gera uma inconsistência no sistema, uma vez que fisicamente um estoque não pode ser menor que zero. Para impedir esta situação, devemos limitar a quantidade digitada ao estoque informado do local selecionado, uma vez que o estoque do produto é sempre maior ou igual ao estoque do local (o estoque atual do produto é igual à soma do estoque de todos os locais relacionados ao produto).

Outro item importante é a definição do filtro de busca dos registros, o qual deve ser ajustado para retornar somente as saídas que não sejam um estorno de movimento. Esse filtro deve ser definido, assim como foi feito na classe **entradas**, dentro do método *filtroFixo()*.

Devemos, ainda, estender o método *incluir()* para que o tipo de movimento (TIPOMOV_TIPO) seja gravado corretamente como uma saída, ou seja, devemos ajustar seu valor para 'S'. Nesse mesmo método precisamos valorizar o movimento corretamente, tanto o valor unitário quanto o valor total. Essa valorização é efetuada por meio do campo virtual que vamos criar no construtor da classe (esse campo será do tipo 'hidden' no formulário).

Temos, portanto, as seguintes ações que devem ser executadas na classe **saídas**:

1. Método *__construct*
   - ➲ Criar um campo virtual CUSTO_SAIDA, cujo comportamento form seja ajustado para 'hidden' e tenha como objetivo o envio do custo médio do produto quando o formulário for submetido.

- Limitar o campo HISTORICO_QUANTIDADE ao valor contido no campo LOCAL_ESTOQUE.
- Não limitar os campos denominados HISTORICO_VALOR_UNIT e HISTORICO_VALOR_TOTAL, uma vez que eles são calculados de forma automática no formulário. Vamos utilizar o custo médio atual do produto para cálculo.

2. Método *BuscaTiposMovimentos*
   - Devemos especificar o filtro de seleção para TIPOMOV_TIPO = 'S' e TIPOMOV_ESTORNO='N'.
3. Método *getFormulario*
   - Nesse método vamos definir o valor do atributo FEntrada para falso (false), indicando que este é um movimento de saída de estoque.
4. Método *filtroFixo*
   - Definimos o filtro fixo como TIPOMOV_TIPO='S'.
5. Método *incluir*
   - Ajustamos o valor do campo TIPOMOV_TIPO para 'S', indicando que se trata de um movimento de saída de estoque.
   - Habilitamos o campo TIPOMOV_TIPO para inclusão através da chamada ao método virtual *setIncluir(true)*.
   - Retiramos o campo virtual CUSTO_SAIDA da lista de inclusão, evitando que a classe **base** o inclua no comando SQL de inclusão de registros (o que, evidentemente, ocasionaria um erro, pois o campo não existe na tabela).
   - Atribuímos o valor contido no campo CUSTO_SAIDA para o campo HISTORICO_VALOR_UNIT.
   - Calculamos o valor do campo HISTORICO_CUSTO_TOTAL, que deve ter o seu valor ajustado para o resultado da multiplicação da quantidade informada pelo valor unitário (que é igual ao custo médio do produto).

```
HISOTIROCO_CUSTO_TOTAL= HISTORICO_QUANTIDADE *
                        HISTORICO_CUSTO_UNIT
```

A classe **saidas** deve estar junto com as classes **historicomovimento** e **entradas**, ou seja, deve estar dentro do arquivo *classe_historicomovimento.inc*, o qual está no diretório *estoque/classes/*. A seguir temos a listagem da classe **saidas**.

```
/**
 * Classe para movimentação de saídas
 *
 */
```

```
class saidas extends historicomovimento {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->getCampo("HISTORICO_VALOR_UNIT")->setMinimo(null);
          $this->getCampo("HISTORICO_VALOR_TOTAL")->setMinimo(null);
          $this->getCampo("HISTORICO_QUANTIDADE")->setMaximo('[LOCAL_ESTOQUE]');
          $this->addCampo(new float("CUSTO_SAIDA","",
                              12,null,null,true,false,false,1,null,null,false));
          $this->getCampo("CUSTO_SAIDA")->setComportamento_form('hidden');
     }

     /**
      * Retorna somente os tipos de movimento de saída
      *
      * @param string $_where
      */
     protected function buscaTiposMovimentos($_where=null) {
          parent::buscaTiposMovimentos("TIPOMOV_TIPO='S' AND TIPOMOV_ESTORNO='N'");
     }

     /**
      * Informa a classe javascript que é uma saída
      *
      * @param string $_funcao
      * @return mixed
      */
     public function getFormulario($_funcao='INC') {
          $_sc = new tag(new tipotag("SCRIPT"),null,"objBP.FEntrada=false;");
          return parent::getFormulario($_funcao) . $_sc->toHTML();
     }

     protected function FiltroFixo() {
          return "TIPOMOV_TIPO='S'";
     }

     public function incluir() {
          $this->getCampo("TIPOMOV_TIPO")->setValor('S');
          $this->getCampo("TIPOMOV_TIPO")->setIncluir(true);
          $this->getCampo("CUSTO_SAIDA")->setIncluir(false);
          $this->getCampo("HISTORICO_VALOR_UNIT")->setValor(
                              $this->getCampo("CUSTO_SAIDA")->getValor());
          $this->getCampo("HISTORICO_VALOR_TOTAL")->setValor(
                              $this->getCampo("CUSTO_SAIDA")->toBD()*
                              $this->getCampo("HISTORICO_QUANTIDADE")->toBD());
          $this->getCampo("HISTORICO_VALOR_TOTAL")->setValor(
                              $this->getCampo("HISTORICO_VALOR_TOTAL")->toHTML());
          return parent::incluir();
     }

}
```

Javascript de movimentação

A classe javascript para controle do formulário de movimentação de estoque possui partes de seu código específicas para o controle do movimento de saída de estoque, uma vez que seu comportamento deve ser diferente daquele esperado para a movimentação de entrada de estoque.

Na operação de entradas de estoque espera-se que todos os campos do formulário, com exceção dos campos informativos, sejam ativados após o usuário selecionar o produto, o local de estoque e o tipo de movimento; já na operação de saída de estoque, como não é requisitada a digitação do valor unitário, tampouco do valor total (um é calculado em função do outro e vice-versa), os métodos envolvidos nesse processo devem deixar esses campos, respectivamente HISTORICO_VALOR_UNIT e HISTORICO_VALOR_TOTAL, desabilitados. Além disso, é necessário que no processo de seleção do produto o campo HISTORICO_VALOR_UNIT seja preenchido com o valor determinado para o custo médio do produto, ou seja, com o valor de PRODUTO_CUSTOMEDIO, que também é atribuído ao campo escondido CUSTO_SAIDA, utilizado para valorização do movimento quando o formulário for submetido ao servidor web.

Dois métodos da classe javascript movimento estão ligados a essas operações. São eles: *movimento.processa* e *movimento.habilitaCampos*. O primeiro método tem a tarefa de alocar o custo médio do produto aos campos relacionados, isto é, aos campos HISTORICO_CUSTOMEDIO e CUSTO_SAIDA. O segundo método deve deixar desabilitados os campos de valores quando o movimento for de saída de estoque.

Para saber se o movimento que está sendo executado é de saída de estoque, os métodos testam o valor do atributo FEntrada da classe movimento. Se o valor do atributo for verdadeiro (true), é um movimento de entrada; caso contrário, trata-se de um movimento de saída de estoque.

O método *processa* com os ajustes para a saída de produtos tem o formato mostrado a seguir:

```
processa: function(xml){
              var l = xml.childNodes[0].childNodes[0];
              var vlr = l.getElementsByTagName('PRODUTO_DESC')[0];
              document.getElementById('PRODUTO_DESC').value =
                                  (vlr.hasChildNodes()
                                       ? vlr.firstChild.nodeValue
                                       : ''
                                  );
              var vlr = l.getElementsByTagName('PRODUTO_ESTOQUE')[0];
              document.getElementById('PRODUTO_ESTOQUE').value =
                                  (vlr.hasChildNodes()
                                       ? vlr.firstChild.nodeValue
                                       : '0,000'
                                  );
              var vlr = l.getElementsByTagName('PRODUTO_CUSTOMEDIO')[0];
              document.getElementById('PRODUTO_CUSTOMEDIO').value =
                                  (vlr.hasChildNodes()
                                       ? vlr.firstChild.nodeValue
                                       : '0,00'
                                  );
              objBP.FOk=true;
              if(this.FEntrada!==true) {
                   document.getElementById('HISTORICO_VALOR_UNIT').value =
                             document.getElementById('PRODUTO_CUSTOMEDIO').value;
                   document.getElementById('CUSTO_SAIDA').value =
```

```
                    document.getElementById('PRODUTO_CUSTOMEDIO').value;
                    };
                document.getElementById('LOCAL_CODIGO').focus();
        }
```

Já o método *habilitaCampos* tem a codificação descrita em seguida:

```
habilitaCampos:
    function() {
        var enabled = false;
        if(document.getElementById('LOCAL_CODIGO').selectedIndex>0
           &&document.getElementById('TIPOMOV_CODIGO').selectedIndex>0) {
            enabled = true;
        };
        document.getElementById('HISTORICO_DOCUMENTO').disabled=!enabled;
        document.getElementById('HISTORICO_QUANTIDADE').disabled=!enabled;
        if(this.FEntrada===true) {
            document.getElementById('HISTORICO_VALOR_UNIT').disabled=!enabled;
            document.getElementById('HISTORICO_VALOR_TOTAL').disabled=!enabled;
        }
        document.getElementById('HISTORICO_HISTORICO').disabled = !enabled;
        document.getElementById('HISTORICO_DATA').disabled = !enabled;
        if(!enabled) {
            document.getElementById('HISTORICO_DOCUMENTO').value = '';
            document.getElementById('HISTORICO_QUANTIDADE').value = '';
            if(this.FEntrada===true) {
                document.getElementById('HISTORICO_VALOR_UNIT').value = '';
                document.getElementById('HISTORICO_VALOR_TOTAL').value = '';
            }
            document.getElementById('HISTORICO_HISTORICO').innerHTML = '';
            document.getElementById('HISTORICO_DATA').value = '';
        } else {
            document.getElementById('HISTORICO_DATA').focus();
        }
    }
```

Criação dos tipos de movimento relacionados ao processo de saídas

Os tipos de movimento descritos no início do capítulo devem ser cadastrados no sistema para que seja possível a geração de movimentos de saída de estoque. Os tipos listados representam o mínimo necessário para o correto funcionamento do sistema, podendo o usuário cadastrar novos tipos conforme a necessidade.

Todos os tipos de movimento de saída de estoque não afetam o cálculo do custo médio nem o custo total do produto, com exceção dos tipos que sejam estorno (vamos discutir os estornos de movimento de estoque no próximo capítulo). Desta forma, o formulário de cadastramento deixa desabilitados esses campos quando o tipo de movimento selecionado for de saída de estoque.

A tabela seguinte mostra os tipos de saída de estoque que precisamos cadastrar.

Código	Descrição	Afeta Custo médio	Afeta Custo atual
SVEN	Saída por Venda	N	N
STF	Saída por Transferência	N	N
SAQ	Saída por Quebra	N	N
SAM	Saída por Ajuste manual	N	N

A tabela deve refletir no sistema de controle de estoque. Para isso é preciso entrar no módulo de cadastro, opção Tipos de movimentos, e cadastrar os quatro tipos listados.

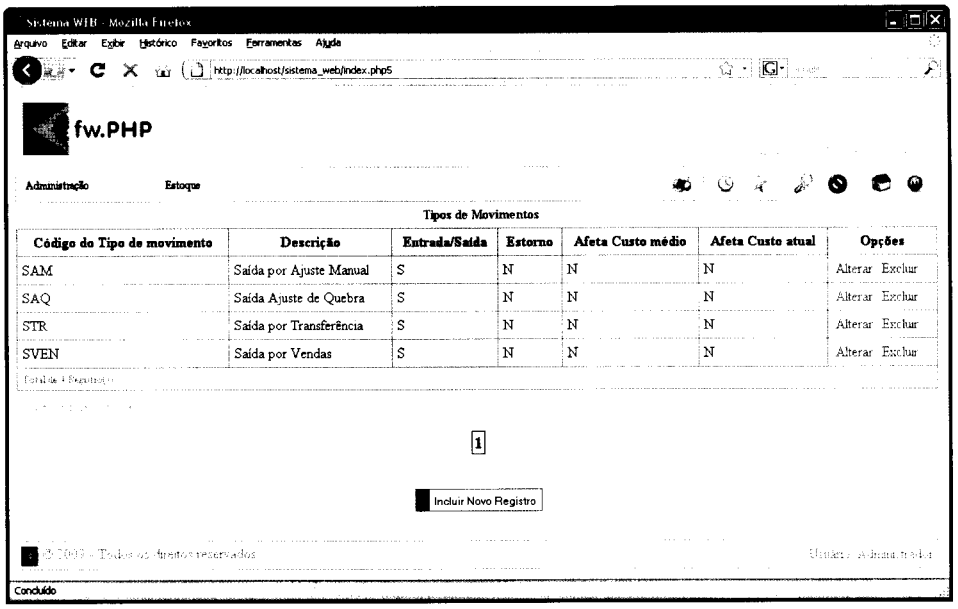


Figura 6.1

Programa de saída de estoque

A listagem seguinte mostra o programa para movimentação de saída de estoque, o qual chamaremos de *mov_saidas.php5*. Nesse programa precisamos definir apenas o arquivo que contém a classe, bem como o nome da classe. Além disso, devemos escolher o título da página, carregar o programa *instanciaclasse.php5* e nada mais.

Lista 1: mov_saidas.php5

```
<?php
/**
 * Movimentação, Saída de estoque
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_historicomovimento.inc",
              "nome"=>"saidas"
           );
$_FTitulo = "Movimentação - Saidas";
return include_once("../instanciaclasse.php5");
?>
```

Menu de saída de estoque

Para que seja possível executar o programa de saída de estoque, assim como os demais programas, precisamos incluí-lo como item do menu do sistema. Vamos incluir o item 'Saídas' no menu Movimentação que pertence ao menu principal Estoque.

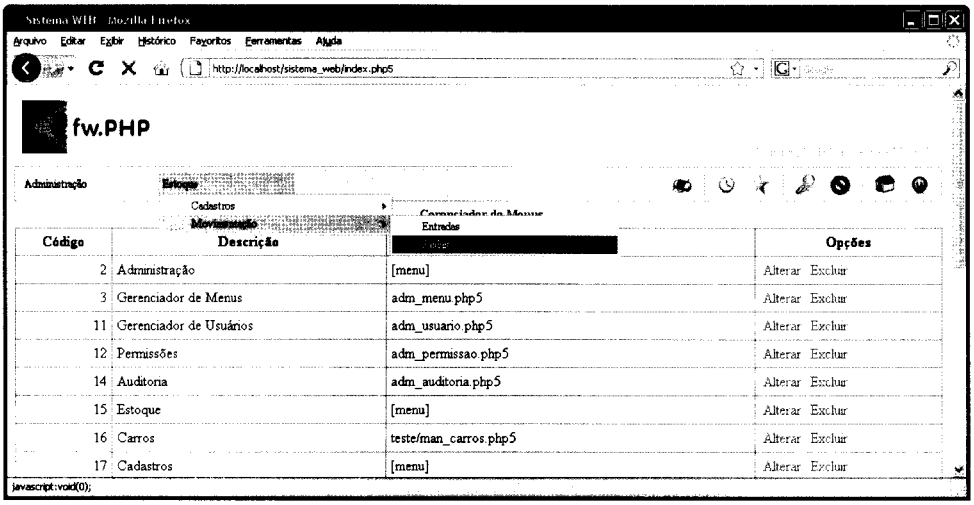


Figura 6.2

Lembre-se de que a inclusão do item no menu libera o acesso para qualquer usuário do sistema. O correto é a restrição deste e de outros itens do sistema somente a pessoas relacionadas à operação de movimentação do estoque. Essa limitação deve ser feita no cadastro de permissões do sistema. Devemos primeiramente criar um usuário e depois liberar o acesso somente aos programas relacionados.

A Figura 6.3 mostra um exemplo em que o usuário 'estoque' tem acesso somente aos processos de entrada e saída de estoque, sendo proibido utilizar as demais rotinas do sistema, que ficam desabilitadas para esse usuário.

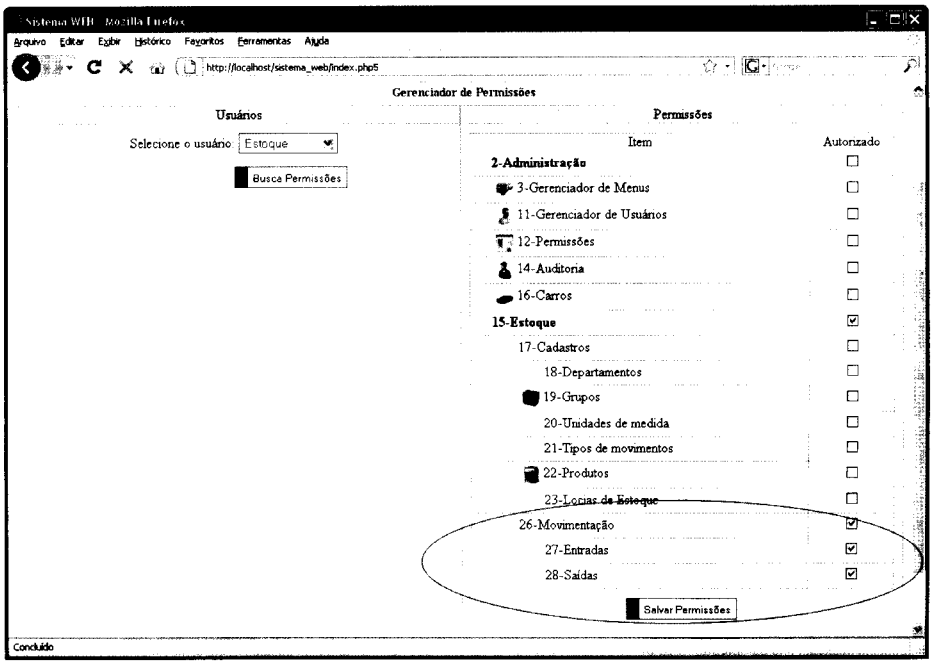


Figura 6.3

A Figura 6.4 exibe o acesso do usuário ao sistema. Note que todos os menus, com exceção do menu de movimentação de estoque, estão desabilitados para esse usuário, ou seja, ele somente tem acesso à entrada e à saída de estoque. Mas lembre-se também de que o ideal é definir as permissões depois que todos os menus tiverem sido criados, pois todo item do menu que for adicionado daqui em diante fica liberado para esse usuário.

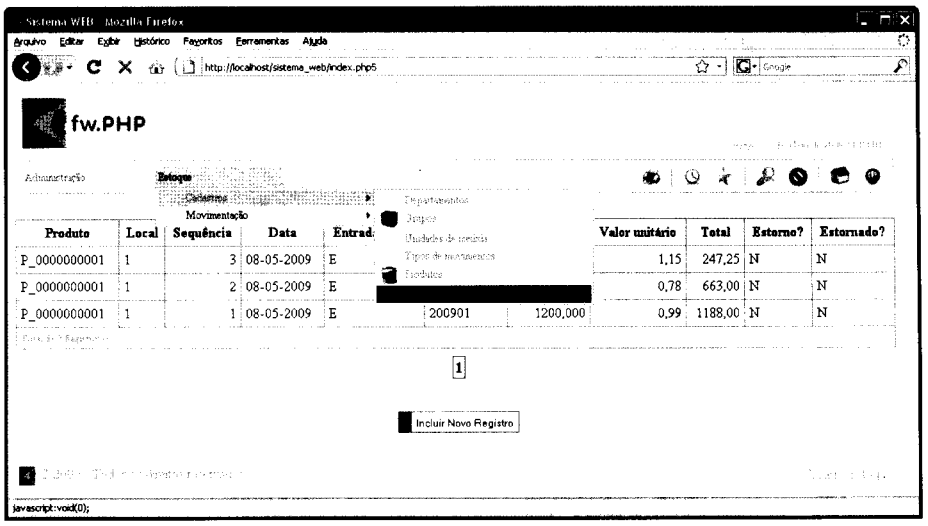


Figura 6.4

Uma vez que temos acesso ao menu de movimentação, podemos complementar a movimentação do produto, executando as saídas necessárias.

As Figuras 6.5 e 6.6 apresentam o processo de saída de estoque. Note que os campos HISTORICO_VALOR_UNIT e HISTORICO_VALOR_TOTAL não são habilitados mesmo depois que produto, local de estoque e tipo de movimento são selecionados pelo usuário.

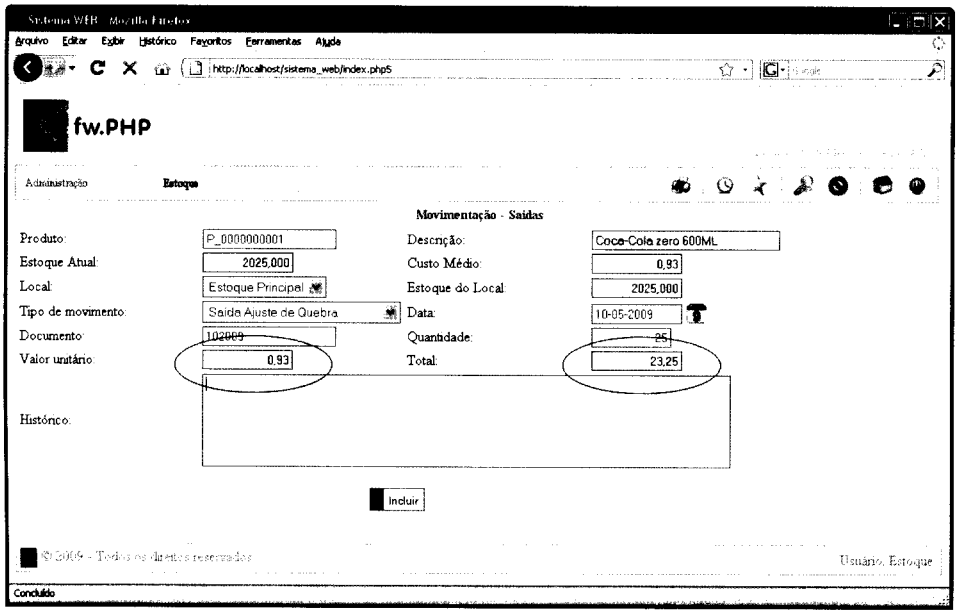


Figura 6.5

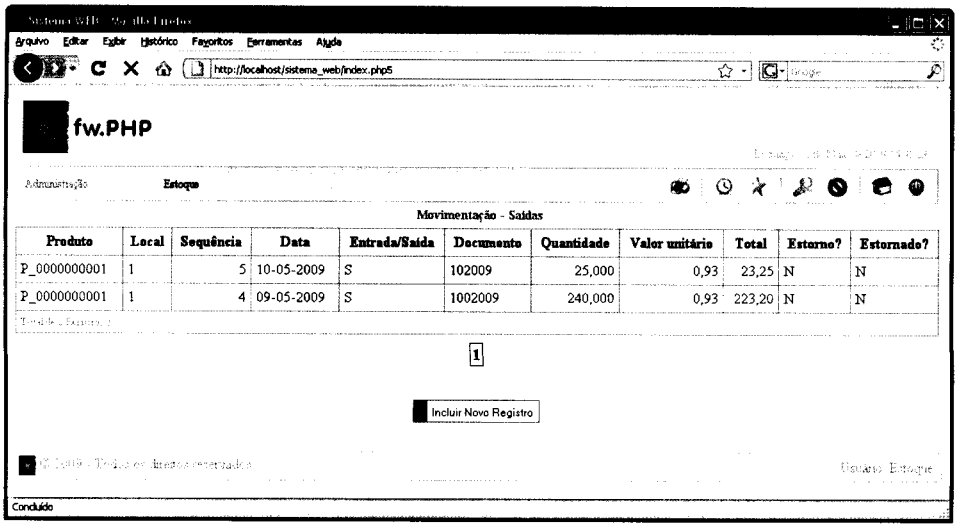
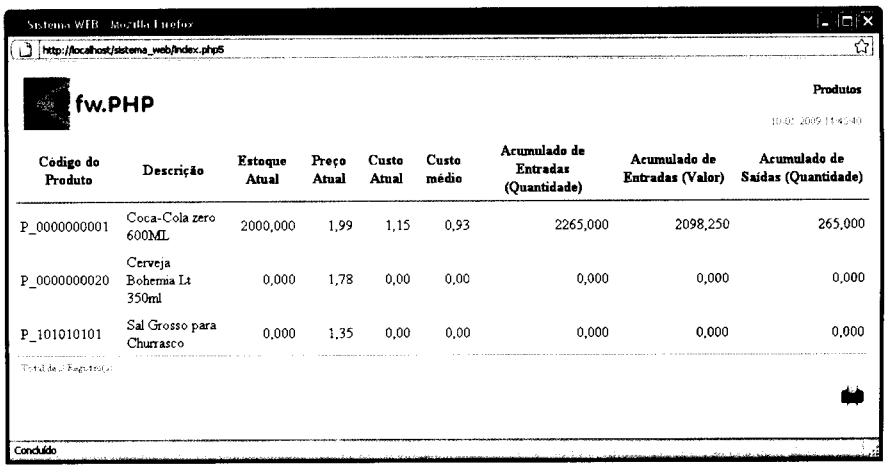


Figura 6.6

Para verificar o impacto no cadastro de produtos, basta solicitarmos a impressão do relatório de produtos na página de cadastro de produtos do sistema.



Código do Produto	Descrição	Estoque Atual	Preço Atual	Custo Atual	Custo médio	Acumulado de Entradas (Quantidade)	Acumulado de Entradas (Valor)	Acumulado de Saídas (Quantidade)
P_0000000001	Coca-Cola zero 600ML	2000,000	1,99	1,15	0,93	2265,000	2098,250	265,000
P_0000000020	Cerveja Bohemia Lt 350ml	0,000	1,78	0,00	0,00	0,000	0,000	0,000
P_101010101	Sal Grosso para Churrasco	0,000	1,35	0,00	0,00	0,000	0,000	0,000

Figura 6.7

A tabela de estoque por produto e local mostra o mesmo resultado apresentado na tabela de produto, uma vez que temos apenas um local de estoque.

Anotações

Capítulo 7

Estorno de Movimentação

Quando um movimento, seja de entrada ou de saída, é efetuado no sistema, ele é considerado pelo sistema como definitivo, ou seja, não é possível excluí-lo. Porém, existe sempre a possibilidade de o movimento estar inconsistente, o que ocasiona uma discrepância entre o estoque físico e o estoque virtual. Como não é possível excluir o movimento, precisamos de um meio para desfazer a transação.

O desfazimento pode ser feito pelo lançamento de um outro movimento no sentido inverso ao que foi realizado anteriormente, isto é, se o movimento incorreto é uma entrada, devemos realizar uma saída e se o movimento for uma saída, devemos efetuar uma entrada de estoque. O problema com esta abordagem é que não temos uma trilha de auditoria confiável, uma vez que dependemos do operador para que o lançamento de desfazimento esteja correto.

Caso o operador erre na entrada de dados digitando o produto errado, a quantidade errada, o local errado, o valor errado, devemos efetuar dois novos lançamentos, um para desfazer o erro e o segundo para desfazer o lançamento inicial. Além deste problema, no desfazimento de transações de entrada os valores lançados não são desfeitos (custo médio, por exemplo). Como último argumento temos que o desfazimento manual permite que o mesmo lançamento seja desfeito várias vezes, uma vez que não há uma relação direta entre o movimento de desfazimento e o lançamento original.

Para contornar esta situação devemos criar um processo que permita reverter o lançamento efetuado com o mínimo de interferência do usuário e que desfaça todo o processo, ajustando estoques e valores. Além disso, o processo deve marcar o lançamento original como estornado, impedindo que ele seja estornado mais de uma vez.

Vamos dividir o estorno em dois tipos. O primeiro é o estorno de lançamentos de entrada de estoque e o segundo é o de lançamentos de saída de estoque. Como o estorno é uma operação inversa à do lançamento original, temos que o estorno de entrada é uma saída e o de saída é uma entrada.

Ambos exigem que sejam cadastrados tipos de movimento com as características necessárias, ou seja, para o estorno de entradas precisamos que exista pelo menos um lançamento de estorno marcado como saída e como estorno, podendo ainda ser marcado ou não para afetar o custo médio e o custo atual do produto (o tipo de movimento deve refletir o lançamento original, ou seja, se o tipo de lançamento original afetou o

custo médio e o atual, o estorno também deve afetar). Desta forma, precisamos de quatro tipos de movimentos de estorno de entrada e um lançamento de estorno de saída.

O desenvolvimento das rotinas de estorno de movimentação exige o acompanhamento dos seguintes passos:

- Criação da classe principal de estorno do movimento;
- Alteração do javascript de movimentos;
- Criação da classe para estorno de entradas;
- Criação da classe para estorno de saídas;
- Criação dos tipos de movimentos de estorno;
- Criação dos programas para estorno de entrada e saída;
- Inclusão do estorno no menu do sistema.

Classe estorno

Como a maioria dos processos de estorno é comum tanto para estorno de entrada quanto para estorno de saída, vamos criar uma classe principal para o gerenciamento do estorno de movimentos, deixando para as classes de estorno de entrada e estorno de saída a definição dos processos específicos para cada operação.

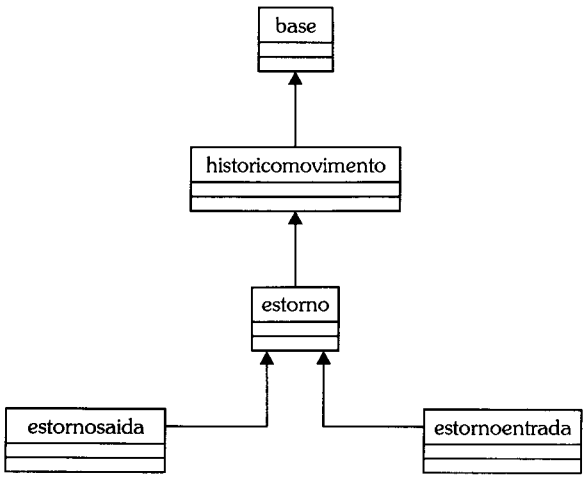


Figura 7.1

Qualquer estorno está sempre ligado a um movimento; logo, a tela de estornos deve listar somente os movimentos que não tenham sido ainda estornados, ou seja, que tenham o campo HISTORICO_ESTORNADO igual a 'N'.

A lista de lançamentos possíveis de serem estornados disponibiliza como opção de lançamento a operação 'Estorno' e somente esta (não temos links para alteração e exclusão de movimento, tampouco o botão para incluir um novo movimento). Para implementar essa funcionalidade, a classe **estorno** redefine o método *getOpcoesLista()*, que na classe **base** está programado para incluir os links para alteração e exclusão. Além disso, a classe redefine o método *getBotaoIncluir()*, retirando a geração do botão de inclusão.

O formulário para inclusão do estorno de movimento deve permitir somente a digitação do tipo de movimento, do número do documento, histórico do movimento e da data do estorno; os demais campos devem ser desabilitados e permanecem deste modo durante todo o processo de estorno (diferentemente dos processos de entrada e saída, em que alguns campos são habilitados durante o processamento).

É necessária a criação de vários campos escondidos no formulário para que os dados do movimento original sejam enviados ao servidor web junto com os dados do formulário de estorno. A lista seguinte mostra os campos que devemos criar no formulário de estorno, todos com comportamento form ajustado para 'hidden'.

1. PRODUTO

   Instância da classe **string**, contém o código do produto presente no movimento original.

2. LOCAL

   Instância da classe **string**, contém o código do local definido no movimento original que está sendo estornado.

3. DATA

   Instância da classe **data**, contém a data do movimento original.

4. SEQUENCIA

   Instância da classe **inteiro**, contém a seqüência do movimento original que está sendo estornado.

5. VALOR_UNITARIO

   Instância da classe **float**, contém o valor unitário do movimento original.

6. QUANTIDADE

   Instância da classe **float**, contém a quantidade do movimento original que está sendo estornado.

Estes campos são criados no método construtor da classe, no qual o atributo comportamento form desses campos é ajustado para 'hidden'.

O método *processaCampoFormulario()* deve ser redefinido para que todos os campos, com exceção de TIPOMOV_CODIGO, sejam desabilitados, ou seja, o método inclui em cada campo o atributo 'DISABLED'.

Nesse mesmo método o campo TIPOMOV_CODIGO recebe, da mesma forma como é feito na classe **historicomovimento**, o atributo 'ONCHANGE' para possibilitar que os campos HISTORICO_DATA, HISTORICO_DOCUMENTO e HISTORICO_HISTORICO sejam habilitados para a digitação pelo usuário (é necessário que a classe javascript movimento seja alterada para suportar essa nova condição).

O método *getOpcoesLista()* redefine totalmente o mesmo método existente na classe **base**, alterando a lista de opções padrão. Na lista-padrão temos as opções para alteração e exclusão do registro vinculado, o que não serve para o processo de estorno. Neste processo precisamos apenas da opção estornar movimento. Uma vez que o processo é muito similar à inclusão de um novo registro, vamos utilizar a ação INC como parâmetro do link de estorno.

O método *getBotaoIncluir()* é redefinido para que não seja exibido o botão de inclusão de um novo registro, pois essa ação não faz sentido no contexto do estorno de movimentos.

O método *getFormulário()* precisa receber alguns incrementos para que seja possível o processamento do estorno. O usuário não pode digitar os dados referentes ao produto, local de estoque, quantidade e valores; logo, precisamos implementar nesse método a busca dos dados referentes ao movimento que será estornado e preencher o formulário. Além dos dados do movimento, precisamos ainda buscar os dados cadastrais do produto (descrição, estoque atual e custo médio) e os dados do local de estoque utilizado (descrição e estoque atual).

Ainda nesse método, devemos informar à classe javascript movimento que o movimento atual é um estorno através do atributo FEstorno (esse atributo será criado adiante, quando falarmos das alterações na classe javascript movimento).

Um cuidado extra que devemos ter nesse método é verificar se o movimento não foi estornado anteriormente, avisando o usuário caso tenha acontecido e impedindo que o processo continue. Para essa validação o método deve verificar o valor do campo HISTORICO_ESTORNADO. Caso esse atributo esteja marcado com 'S', indica que o movimento foi estornado e não pode ser novamente.

Para finalizar temos o método *incluir()* que deve receber alguns processos extras antes que o método da classe **historicomovimento** seja executado (o qual realiza outros processos antes de executar o método principal na classe **base**).

Nesse método devemos marcar o movimento que está sendo incluído como um estorno, além de informar qual movimento foi estornado (a seqüência do movimento). Além disso, esses campos (HISTORICO_ESTORNO e HISTORICO_ESTORNO_SEQ) devem ser habilitados para o processo de inclusão do movimento, o que é feito pelo

método virtual *setIncluir(true)*. Como vários campos do formulário foram definidos como desabilitados (DISABLED), tivemos de criar campos auxiliares para conter os dados básicos do movimento. Esses dados precisam ser repassados aos campos principais da classe e os campos auxiliares devem ser retirados do processo de inclusão (por meio do método virtual *setIncluir(false)*). Depois, precisamos valorizar o movimento, definindo o valor unitário do movimento e calculando o valor total.

Finalmente, após a execução com sucesso da inclusão do movimento de estorno, devemos marcar o movimento original como já estornado, impedindo que ele seja estornado novamente no futuro. Para executar essa marcação, precisamos alterar o valor do campo HISTORICO_ESTORNADO para 'S'.

Esses métodos definem a classe **estorno**, listada a seguir. Essa classe não será chamada diretamente, uma vez que precisamos separar os estornos de entrada e os de saída, pois cada tipo de estorno possui um comportamento específico, embora boa parte dos processos seja genérica.

```
/**
 * Classe base para o Estorno de movimentos
 * Esta classe deve ser instanciada por outras classes
 *
 */
class estorno extends historicomovimento {
    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->getCampo("HISTORICO_VALOR_UNIT")->setMinimo(null);
        $this->getCampo("HISTORICO_VALOR_TOTAL")->setMinimo(null);
        $this->getCampo("HISTORICO_SEQUENCIA")->setIncluir(true);
        // Criar campos auxiliares.. todos hidden
        $_arr = Array('PRODUTO'=>'string','LOCAL'=>'string','DATA'=>'data',
                      'SEQUENCIA'=>'inteiro','VALOR_UNITARIO'=>'float',
                      'QUANTIDADE'=>'float');
        foreach($_arr as $_campo=>$_tipo) {
            $this->addCampo(new $_tipo($_campo,"",1,null,null,
                                true,false,false,null,null,null,false));
            if($_tipo=='string'||$_tipo=='data') {
                $this->getCampo($_campo)->SetConn($_conn);
            }
            $this->getCampo($_campo)->setComportamento_form('hidden');
        }
        $this->_exibe_opcoes = true;
    }

    /**
     * Teremos apenas a opção de estorno na lista
     *
     * @param string $_pk
     * @param tag $_tab
     * @param tipospadrao $_tipos
     */
    protected function getOpcoesLista($_pk,tag & $_tab,tipospadrao $_tipos) {
            $_lnkalt = new tag($_tipos->getTipo("A"),
                        Array(new atributo("HREF","javascript:void(0);"),
                            new atributo("ONCLICK",
                                "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?" .
                                "ACAO=INC{$_pk}','CORPO');")),
                        " Estornar ");
```

```
            $_tab->getLastSubTag()->addSubTag(new tag($_tipos->getTipo('TD'),
                    Array(new atributo("STYLE","border:1px solid #a0a0a0;")),
                    $_lnkalt->toHTML()));
    }

    /**
     * Desabilita todos os campos do formulário, exceto Tipo de movimento
     *
     * @param string $_nome
     * @param tag $_campo
     * @param string $_metodo
     */
    protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
        if(stripos($_metodo,'getformulario')!==false) {
            switch($_nome) {
                case 'PRODUTO_CODIGO':
                case 'PRODUTO_DESC':
                case 'PRODUTO_ESTOQUE':
                case 'LOCAL_CODIGO':
                case 'LOCAL_ESTOQUE':
                case 'HISTORICO_DOCUMENTO':
                case 'HISTORICO_SEQUENCIA':
                case 'HISTORICO_HISTORICO':
                case 'HISTORICO_DATA':
                case 'PRODUTO_CUSTOMEDIO':
                case 'HISTORICO_QUANTIDADE':
                case 'HISTORICO_VALOR_UNIT':
                case 'HISTORICO_VALOR_TOTAL':
                            $_campo->addAtributo(new atributo("DISABLED"));
                            break;
                case 'TIPOMOV_CODIGO':
                            $_campo->addAtributo(new atributo("ONCHANGE",
                                                    "objBP.validaTipomov();"));
                            break;
            }
        }
    }

    /**
     * Ajusta o formulário de Estorno
     *
     * @param string $_funcao
     * @return mixed
     */
    public function getFormulario($_funcao='INC') {
        // Temos os dados de busca, precisamos recuperar o registro...
        if($this->Buscar(Array(
            'PRODUTO_CODIGO'=>$_GET['PRODUTO_CODIGO'],
            'LOCAL_CODIGO'=>$_GET['LOCAL_CODIGO'],
            'HISTORICO_DATA'=>$_GET['HISTORICO_DATA'],
            'HISTORICO_SEQUENCIA'=>$_GET['HISTORICO_SEQUENCIA']))===false) {
            die(utf8_encode("<span style='color:red;font-size:20px;" .
                "text-align:center;'>Registro a Estornar não " .
                "encontrado</span>"));
        }
        $this->proximo();
        if($this->getCampo("HISTORICO_ESTORNADO")=='S') {
            die(utf8_encode("<span style='color:red;font-size:20px;" .
                    "text-align:center;'>Movimento já estornado</span>"));
        }
        // Preencher
        $_sctxt = "document.getElementById('PRODUTO_CODIGO').value=" .
                                        "'{$_GET['PRODUTO_CODIGO']}';" .
                "document.getElementById('LOCAL_CODIGO').value=" .
```

```
                                        "'{$_GET['LOCAL_CODIGO']}';" .
                "document.getElementById('HISTORICO_SEQUENCIA').value=" .
                                        "'{$_GET['HISTORICO_SEQUENCIA']}';" .
                "document.getElementById('HISTORICO_DATA').value=" .
                                        "'{$_GET['HISTORICO_DATA']}';" .
                "document.getElementById('PRODUTO').value=" .
                                    "'{$_GET['PRODUTO_CODIGO']}';" .
                "document.getElementById('LOCAL').value=" .
                                    "'{$_GET['LOCAL_CODIGO']}';" .
                "document.getElementById('SEQUENCIA').value=" .
                                    "'{$_GET['HISTORICO_SEQUENCIA']}';" .
                "document.getElementById('DATA').value=" .
                                    "'{$_GET['HISTORICO_DATA']}';" .
                "document.getElementById('HISTORICO_DOCUMENTO').value='';" .
                "document.getElementById('HISTORICO_QUANTIDADE').value=" .
                     "'{$this->getCampo('HISTORICO_QUANTIDADE')->toHTML()}';" .
                "document.getElementById('QUANTIDADE').value=" .
                     "'{$this->getCampo('HISTORICO_QUANTIDADE')->toHTML()}';" .
                "document.getElementById('HISTORICO_VALOR_UNIT').value=" .
                     "'{$this->getCampo('HISTORICO_VALOR_UNIT')->toHTML()}';" .
                "document.getElementById('VALOR_UNITARIO').value=" .
                     "'{$this->getCampo('HISTORICO_VALOR_UNIT')->toHTML()}';" .
                "document.getElementById('HISTORICO_VALOR_TOTAL').value=" .
                     "'{$this->getCampo('HISTORICO_VALOR_TOTAL')->toHTML()}';" .
                "document.getElementById('HISTORICO_HISTORICO').innerHTML=" .
                     "'Estorno do Documento:" .
                     "{$this->getCampo('HISTORICO_DOCUMENTO')->toHTML()}';";
    // Buscar Dados do produto e do local de estoque
    $_prd = new produto($this->_conn);
    $_prd->Buscar(Array('PRODUTO_CODIGO'=>$_GET['PRODUTO_CODIGO']));
    $_prd->proximo();
    $_sctxt .= "document.getElementById('PRODUTO_DESC').value=" .
                        "'{$_prd->getCampo('PRODUTO_DESC')->toHTML()}';" .
               "document.getElementById('PRODUTO_ESTOQUE').value=" .
                     "'{$_prd->getCampo('PRODUTO_ESTOQUE')->toHTML()}';" .
               "document.getElementById('PRODUTO_CUSTOMEDIO').value=" .
                     "'{$_prd->getCampo('PRODUTO_CUSTOMEDIO')->toHTML()}';";
    $_lcl = new estoquelocal($this->_conn);
    $_lcl->Buscar(Array(
              'PRODUTO_CODIGO'=>$_GET['PRODUTO_CODIGO'],
              'LOCAL_CODIGO'=>$_GET['LOCAL_CODIGO']));
    $_lcl->proximo();
    $_sctxt .= "document.getElementById('LOCAL_ESTOQUE').value=" .
                     "'{$_lcl->getCampo('PRODUTO_ESTOQUE')->toHTML()}';";
    $_sc = new tag(new tipotag("SCRIPT"),
                     null,"objBP.FEstorno=true;{$_sctxt}");
    return parent::getFormulario($_funcao) . $_sc->toHTML();
}


/**
 * Não teremos o botão incluir nesta lista
 *
 * @param tag $_div
 * @param tipospadrao $_tipos
 */
protected function getBotaoIncluir(tag & $_div,tipospadrao $_tipos) {
}

/**
 * Ajusta os campos da classe antes de inlcuir
 *
 * @return mixed
 */
```

```
    public function incluir() {
        $this->getCampo("HISTORICO_ESTORNO_SEQ")->setIncluir(true);
        $this->getCampo("HISTORICO_ESTORNO")->setIncluir(true);
        $this->getCampo("HISTORICO_ESTORNO_SEQ")->setValor(
                                $this->getCampo("SEQUENCIA")->getValor());
        $this->getCampo("HISTORICO_ESTORNO")->setValor('S');
        $_arr = Array('PRODUTO'=>'PRODUTO_CODIGO','LOCAL'=>'LOCAL_CODIGO',
                      'DATA'=>'DATA','SEQUENCIA'=>'SEQUENCIA',
                      'QUANTIDADE'=>'HISTORICO_QUANTIDADE',
                      'VALOR_UNITARIO'=>'HISTORICO_VALOR_UNIT');
        foreach($_arr as $_alias=>$_campo) {
            $this->getCampo($_campo)->setValor(
                                        $this->getCampo($_alias)->getValor());
            $this->getCampo($_alias)->setIncluir(false);
        }
        $this->getCampo("HISTORICO_VALOR_TOTAL")->setValor(
                      $this->getCampo("HISTORICO_VALOR_UNIT")->toBD()*
                      $this->getCampo("HISTORICO_QUANTIDADE")->toBD());
        $this->getCampo("HISTORICO_VALOR_TOTAL")->setValor(
                      $this->getCampo("HISTORICO_VALOR_TOTAL")->toHTML());
        if(($_res=parent::incluir())!==false) {
            // Marcar o Movimento estornado para não aparecer mais na lista
            $_strSQL = "UPDATE {$this->_nome_tabela} SET " .
                       "HISTORICO_ESTORNADO='S' WHERE " .
                       "PRODUTO_CODIGO=" .
                       "{$this->getCampo('PRODUTO_CODIGO')->toBD()} AND " .
                       "LOCAL_CODIGO=" .
                       "{$this->getCampo('LOCAL_CODIGO')->toBD()} AND " .
                       "HISTORICO_DATA=" .
                       "{$this->getCampo('HISTORICO_DATA')->toBD()} AND " .
                       "HISTORICO_SEQUENCIA=" .
                       "{$this->getCampo('HISTORICO_ESTORNO_SEQ')->toBD()}";
            $this->_conn->executaSQL($_strSQL);
        }
        return $_res;
    }
}
```

Classe javascript de movimentos

Precisamos de poucas alterações na classe javascript responsável pelo processamento dinâmico do formulário de movimentação de estoque. Devemos apenas criar um atributo que indica se o movimento atualmente processado é um estorno ou não e alterar quatro métodos da classe para que o processamento considere o estorno no momento de processar o formulário.

O método *inializa()*, responsável por inicializar o objeto e limpar alguns campos do formulário, antes de executar a limpeza deve verificar se não é um movimento de estorno, e neste caso não deve limpar os campos do formulário e precisa, além disso, marcar o atributo FOk como verdadeiro (true), uma vez que o produto já está definido (a classe **estorno** realizou a busca do produto na montagem do formulário).

O método *processa()* somente preenche os valores do custo médio quando não for um estorno, ou seja, quando o atributo FEstorno for diferente de verdadeiro (true).

O método *produtoOk()*, que verifica se o produto já foi informado ou não, devolvendo um erro caso não tenha sido, deve realizar a verificação somente se o movimento não for um estorno.

Por último, o método *habilitaCampos()* deve verificar se o movimento atual é um estorno de movimentação e, neste caso, deve habilitar apenas os campos documento (HISTORICO_DOCUMENTO), data (HISTORICO_DATA) e também histórico (HISTORICO_HISTORICO).

```
function movimento() {};
movimento.prototype = {
FOk: false,
FEntrada: true,
FEstorno: false,
inicializa:
     function(){
          this.FOk=false;
          if(this.FEstorno!==true) {
               document.getElementById('PRODUTO_DESC').value='';
               document.getElementById('PRODUTO_ESTOQUE').value='';
               document.getElementById('PRODUTO_CUSTOMEDIO').value='';
               document.getElementById('LOCAL_CODIGO').selectedIndex=0;
               document.getElementById('TIPOMOV_CODIGO').selectedIndex=0;
          } else {
               this.FOk = true;
          }
     },
buscar:
     function() {
          var params='classe=produto&metodo=buscarXML&' +
                    'arquivo_classe=estoque/classes/classe_produto.inc';
          params += '&campos=PRODUTO_CODIGO&valores=' +
                    document.getElementById('PRODUTO_CODIGO').value;
          ObjProcAjax.runPostXML('executa_busca_ajax.php5',params,objBP.processa);
     },
processa:
     function(xml){
          var l = xml.childNodes[0].childNodes[0];
          var vlr = l.getElementsByTagName('PRODUTO_DESC')[0];
          document.getElementById('PRODUTO_DESC').value = (vlr.hasChildNodes()
                                   ? vlr.firstChild.nodeValue
                                   : '');
          var vlr = l.getElementsByTagName('PRODUTO_ESTOQUE')[0];
          document.getElementById('PRODUTO_ESTOQUE').value = (vlr.hasChildNodes()
                                   ? vlr.firstChild.nodeValue
                                   : '0,000');
          var vlr = l.getElementsByTagName('PRODUTO_CUSTOMEDIO')[0];
          document.getElementById('PRODUTO_CUSTOMEDIO').value =
                                   (vlr.hasChildNodes()
                                            ? vlr.firstChild.nodeValue
                                            : '0,00');
          objBP.FOk=true;
          if(this.FEntrada!==true&&this.FEstorno!==true) {
               document.getElementById('HISTORICO_VALOR_UNIT').value =
                              document.getElementById('PRODUTO_CUSTOMEDIO').value;
               document.getElementById('CUSTO_SAIDA').value =
                              document.getElementById('PRODUTO_CUSTOMEDIO').value;
          };
          document.getElementById('LOCAL_CODIGO').focus();
     },
```

```
produtoOk:
     function() {
          if(this.FOk!==true&&this.FEstorno!==true){
               alert('Selecione um produto primeiro');
               document.getElementById('PRODUTO_CODIGO').focus();
                    return false;
               else {
                    return true;
               }
          },
buscaLocal:
          function() {
               if(this.produtoOk()!==true){
                    document.getElementById('LOCAL_CODIGO').selectedIndex=0;
               } else {
                    var l = document.getElementById('LOCAL_CODIGO');
                    var local =l.options[l.selectedIndex].value;
                    if(local!='-') {
                         var params='classe=estoquelocal&metodo=buscarXML&' +
                                   'arquivo_classe=estoque/classes/' +
                                   'classe_estoquelocal.inc';
                         params += '&campos=PRODUTO_CODIGO,LOCAL_CODIGO&valores='+
                                   document.getElementById('PRODUTO_CODIGO').value;
                         params += ','+local;
                         ObjProcAjax.runPostXML('executa_busca_ajax.php5',
                                                params,objBP.EstoqueLocal);
                    } else {
                         document.getElementById('LOCAL_ESTOQUE').value = '-';
                    }
               };
               this.habilitaCampos();
          },
validaTipomov:
     function() {
          if(this.produtoOk()!==true){
               document.getElementById('TIPOMOV_CODIGO').selectedIndex=0;
          };
          this.habilitaCampos();
     },
EstoqueLocal:
     function(xml) {
          var est = '0,000';
          if(xml.hasChildNodes()&&xml.childNodes[0].hasChildNodes()) {
               var l = xml.childNodes[0].childNodes[0];
               var vlr = l.getElementsByTagName('PRODUTO_ESTOQUE')[0];
               est =  (vlr.hasChildNodes() ? vlr.firstChild.nodeValue : '');
          }
          document.getElementById('LOCAL_ESTOQUE').value = est;
     },
habilitaCampos:
     function() {
          var enabled = false;
          if(document.getElementById('LOCAL_CODIGO').selectedIndex>0&&
                    document.getElementById('TIPOMOV_CODIGO').selectedIndex>0) {
               enabled = true;
          };
          document.getElementById('HISTORICO_DOCUMENTO').disabled = !enabled;
          document.getElementById('HISTORICO_QUANTIDADE').disabled =
                                                  !enabled||(this.FEstorno===true);
          if(this.FEntrada===true) {
               document.getElementById('HISTORICO_VALOR_UNIT').disabled =
                                                  !enabled||(this.FEstorno===true);
               document.getElementById('HISTORICO_VALOR_TOTAL').disabled =
                                                  !enabled||(this.FEstorno===true);
```

```
        }
        document.getElementById('HISTORICO_HISTORICO').disabled = !enabled;
        document.getElementById('HISTORICO_DATA').disabled = !enabled;
        if(!enabled&&this.FEstorno!==true) {
            document.getElementById('HISTORICO_DOCUMENTO').value = '';
            document.getElementById('HISTORICO_QUANTIDADE').value = '';
            if(this.FEntrada===true) {
                document.getElementById('HISTORICO_VALOR_UNIT').value = '';
                document.getElementById('HISTORICO_VALOR_TOTAL').value = '';
            }
            document.getElementById('HISTORICO_HISTORICO').innerHTML = '';
            document.getElementById('HISTORICO_DATA').value = '';
        } else {
            document.getElementById('HISTORICO_DATA').focus();
        }
    }
};
```

Classe para estorno de entradas

O estorno de um movimento de entrada requer que seja executado um movimento de saída de estoque. Além disso, é necessário que o custo médio e os valores acumulados sejam acertados (desde que o lançamento original também tenha afetado esses campos do produto e do local de estoque).

Os seguintes critérios precisam ser especificados na classe para estorno de entradas:

- Somente os movimentos de entrada não estornados devem ser exibidos na lista.
- O movimento gerado deve ser uma saída de estoque.
- Somente os tipos de movimentos de saída e que sejam para estorno devem aparecer na seleção dos tipos de movimentos.
- O estoque não pode ficar negativo, uma vez que se trata de uma saída de estoque.

Para definir que somente os movimentos de entrada não estornados apareçam na lista de registros a estornar, devemos estabelecer um filtro de registros, o que deve ser feito no método *filtroFixo()*. Esse método deve retornar a condição:

```
TIPOMOV_TIPO='E' AND HISTORICO_ESTORNADO='N'
```

Como o movimento gerado deve ser uma saída de estoque, precisamos, no método *incluir()*, definir o atributo TIPOMOV_TIPO como 'S' antes da execução do processo de geração do movimento.

A lista de tipos de movimentos que devem aparecer no formulário de estorno de movimento deve ter o filtro definido para que somente os tipos de movimento de saída e marcados como estorno sejam exibidos. Para isso a seguinte condição de busca deve ser implementada no método *buscaTiposMovimentos()*:

```
TIPOMOV_TIPO='S' AND TIPOMOV_ESTORNO='S'
```

Uma vez que o estoque não pode ficar negativo (esta é uma regra básica do sistema), devemos limitar a quantidade ao estoque do local (assim como foi feito na classe **saidas**). Esse critério é especificado no método de construção da classe.

```
/**
 * Classe para o estorno de entradas de estoque
 *
 */
class estornoentrada extends estorno {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->getCampo("HISTORICO_QUANTIDADE")->setMaximo('[LOCAL_ESTOQUE]');
     }

     /**
      * Define o filtro como Entrada + Não Estornado
      *
      * @return string
      */
     protected function FiltroFixo() {
          return "TIPOMOV_TIPO='E' AND HISTORICO_ESTORNADO='N'";
     }

     /**
      * Ajusta o tipo de movimento para 'S'
      * Uma vez que o estorno de uma entrada é uma saída
      *
      * @return mixed
      */
     public function incluir() {
          $this->getCampo("TIPOMOV_TIPO")->setValor('S');
          $this->getCampo("TIPOMOV_TIPO")->setIncluir(true);
          return parent::incluir();
     }

     /**
      * Retorna somente tipos de movimentos que sejam um estorno de entrada
      *
      * @param string $_where
      */
     protected function buscaTiposMovimentos($_where=null) {
          parent::buscaTiposMovimentos("TIPOMOV_TIPO='S' AND TIPOMOV_ESTORNO='S'");
     }
}
```

Classe para estorno de saídas

No estorno de um movimento de entrada geramos um movimento de saída de estoque; logo, o processo de estornar um movimento de saída deve gerar um movimento de entrada de estoque. Como os movimentos de saída de estoque não afetam a parte financeira do estoque, o tipo de movimento que vamos utilizar para esse processo também não deve afetar o custo médio nem o custo atual do produto e do local de estoque, por isso no cadastro desse tipo de movimento o sistema desabilita os marcadores para custo médio e atual.

A classe de estorno de saídas é tão simples quanto a classe de estorno de entradas. Já que a classe **estorno** realiza a maioria dos processos necessários, precisamos definir apenas alguns processos para que o estorno da saída de estoque seja executado corretamente.

- Somente os movimentos de saída não estornados devem ser exibidos na lista.
- O movimento gerado deve ser uma entrada de estoque.
- Somente os tipos de movimentos de entrada e que sejam para estorno devem aparecer na seleção dos tipos de movimentos.

Para determinar o critério de seleção dos registros para estorno de saída de estoque devemos, assim como foi feito no estorno de entradas, definir o filtro apropriado no método *filtroFixo()*. O critério deve ser:

```
TIPOMOV_TIPO='S' AND HISTORICO_ESTORNADO='N'
```

O método *incluir()* deve ser estendido e nele devemos definir o valor do campo TIPOMOV_TIPO como 'E', ou seja, que o movimento gerado é uma entrada de estoque. Além disso, devemos incluir esse campo na lista de campos que participam da inclusão do movimento (como padrão, o campo não está na lista). Para tanto é preciso ajustar o atributo de inclusão para verdadeiro (true), o que é feito por meio do método virtual *setIncluir(true)*.

A lista de tipos de movimentos que devem aparecer no formulário de estorno de movimento deve ter o filtro definido para que somente os tipos de movimento de entrada e marcados como estorno sejam exibidos. Para isso a seguinte condição de busca deve ser implementada no método *buscaTiposMovimentos()*:

```
TIPOMOV_TIPO='E' AND TIPOMOV_ESTORNO='S'
```

Não precisamos nos preocupar com o estoque do produto ou do local, uma vez que o movimento resultante é uma entrada, o que implica em uma soma no estoque do produto e do local.

```
/**
 * Classe para o estorno de saídas de estoque
 *
 */
class estornosaida extends estorno {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
     }

     /**
      * Define o filtro como Saída + Não Estornado
      *
      * @return string
      */
     protected function FiltroFixo() {
          return "TIPOMOV_TIPO='S' AND HISTORICO_ESTORNADO='N'";
     }

     /**
      * Ajusta o tipo de movimento para 'E'
      * Uma vez que o estorno de uma saída é uma entrada
      *
      * @return mixed
      */
     public function incluir() {
          $this->getCampo("TIPOMOV_TIPO")->setValor('E');
          $this->getCampo("TIPOMOV_TIPO")->setIncluir(true);
          return parent::incluir();
     }

     /**
      * Retorna somente tipos de movimentos que sejam um estorno de saída
      *
      * @param string $_where
      */
     protected function buscaTiposMovimentos($_where=null) {
          parent::buscaTiposMovimentos("TIPOMOV_TIPO='E' AND TIPOMOV_ESTORNO='S'");
     }
}
```

Criação dos tipos de movimento

Assim como ocorreu com os outros processos de movimentação de estoque, aqui também precisamos que tipos específicos de movimentos estejam cadastrados no sistema, tanto para o estorno de entradas quanto para o estorno de saídas.

O estorno do movimento de entrada de estoque exige que pelo menos um movimento de estorno de entrada esteja cadastrado, porém é recomendável que sejam cadastrados vários tipo de estorno de entrada, um para cada situação. A combinação de que precisamos é:

Afeta Custo médio	Afeta Custo atual
S	S
S	N

Afeta Custo médio	Afeta Custo atual
N	S
N	N

Ou seja, precisamos de quatro tipos de movimentos de estorno de entrada. A tabela seguinte mostra os tipos de movimentos para o estorno de entrada de estoque.

Código	Descrição	Afeta Custo médio	Afeta Custo atual
ESTE	Estorno de Entrada	S	S
ESTESN	Estorno de Entrada (CM)	N	N
ESTENS	Estorno de Entrada (CA)	N	S
ESTENN	Estorno de Entrada (S/C)	N	N

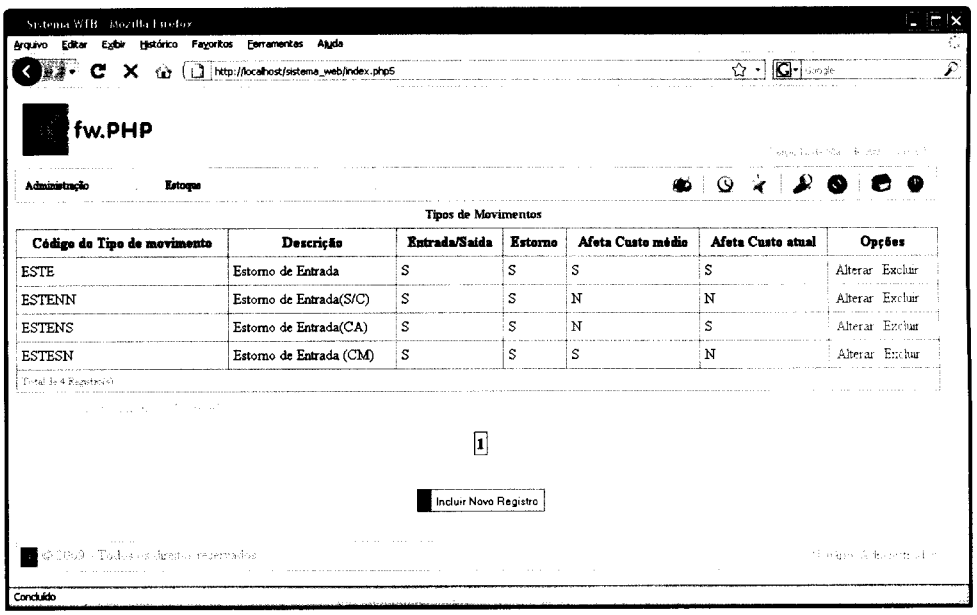


Figura 7.2

Já o estorno de movimento de saída de estoque exige que apenas um tipo de movimento seja cadastrado, o qual deve ser uma entrada que não afete o custo médio nem o custo atual do produto e do local de estoque.

Código	Descrição	Afeta Custo médio	Afeta Custo atual
ESTS	Estorno de Saída	N	N

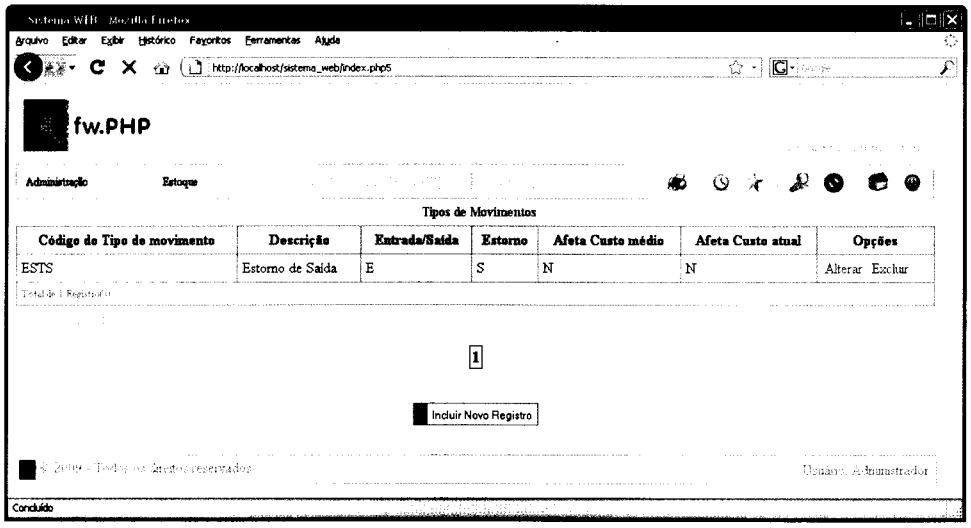


Figura 7.3

Programas para estorno de entrada e estorno de saída

Como de costume, os programas são bem simples. Precisamos apenas informar o arquivo que contém a classe, o nome da classe e o título da página. Por último, basta carregar o programa *instanciaclasse.php5*.

Lista 1: mov_estorno_entrada.php5

```
<?php
/**
 * Movimentação, Estorno de entrada de estoque
 *
 */
$_classe = Array(
                    "arquivo"=>"estoque/classes/classe_historicomovimento.inc",
                    "nome"=>"estornoentrada");
$_FTitulo = "Movimentação - Estorno de Entradas";
return include_once("../instanciaclasse.php5");
?>
```

Lista 2: mov_estorno_saida.php5

```
<?php
/**
 * Movimentação, Estorno de saída de estoque
 *
 */
$_classe = Array(
                "arquivo"=>"estoque/classes/classe_historicomovimento.inc",
                "nome"=>"estornosaida");
$_FTitulo = "Movimentação - Estorno de Saídas";
return include_once("../instanciaclasse.php5");
?>
```

Menu do sistema

Para utilizar os programas de estorno, precisamos criar primeiro os itens no menu de movimentação dentro do controle de estoques. Vamos incluir os dois programas, tanto o de estorno de entradas quanto o de estorno de saídas. A Figura 7.4 exibe os dois menus criados no sistema.

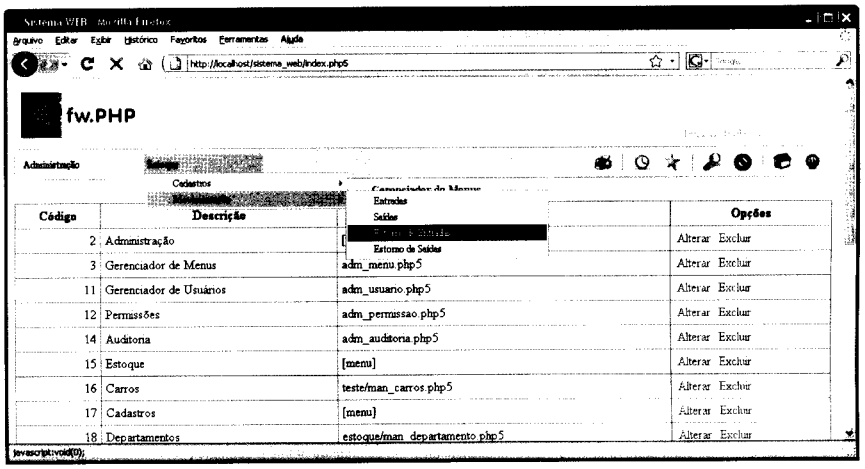


Figura 7.4

As Figuras 7.5 e 7.6 mostram, respectivamente, o lançamento de estorno de entrada e o lançamento de estorno de saída de estoque. Note que somente parte dos campos do formulário está habilitada para digitação.

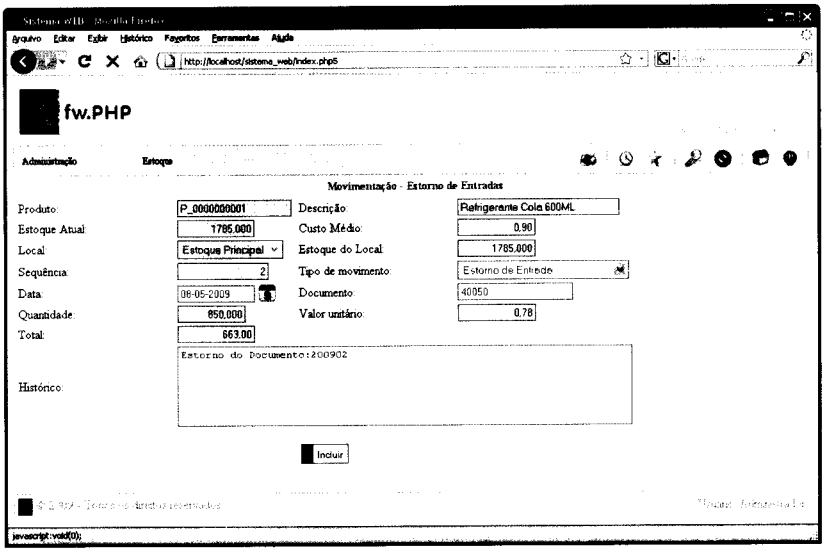


Figura 7.5

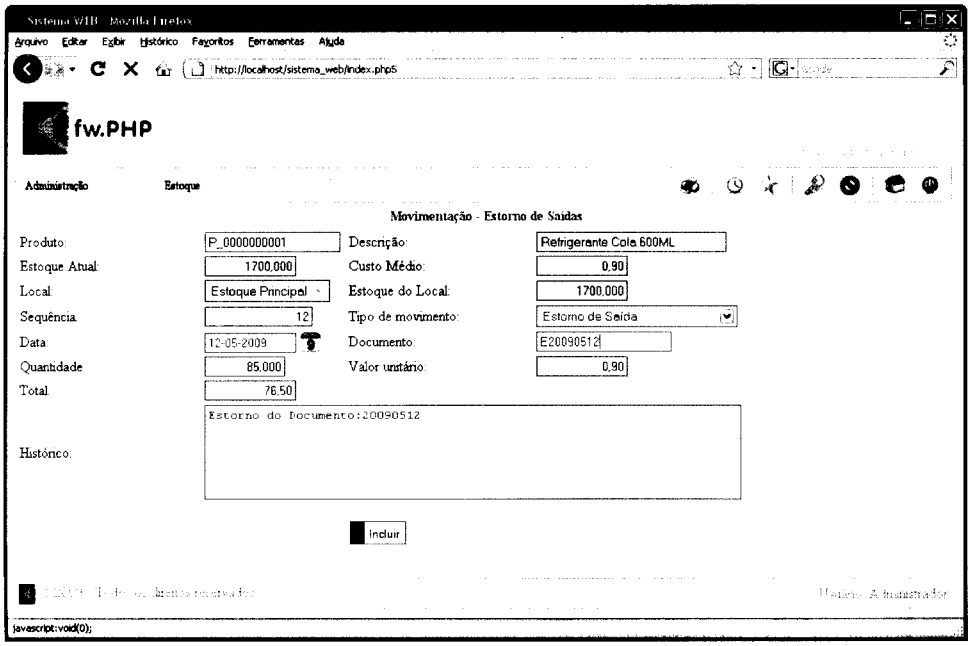


Figura 7.6

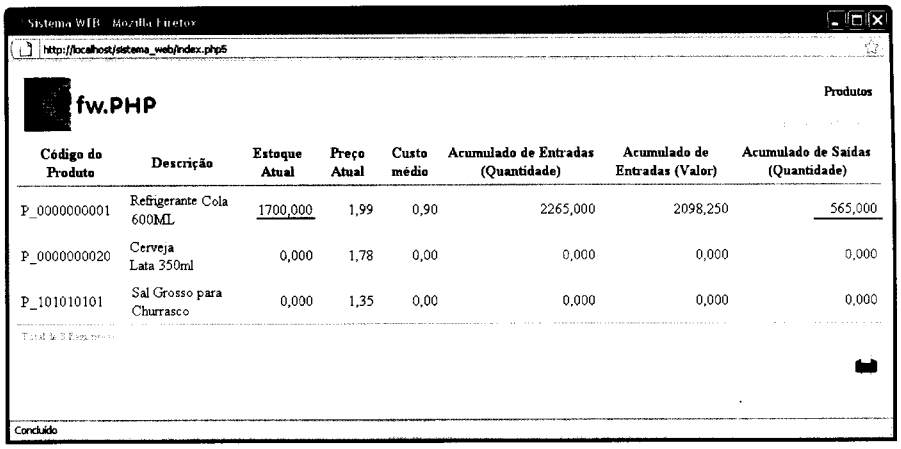
O formulário de lançamento de estorno é um pouco diferente dos formulários gerados até o momento. Vale a pena olhar uma parte do código HTML da página gerada (somente o que é diferente).

```
<div id="CORPO" style="padding: 1px 1px 20px; width: 1024px;">
<script>LoadJS('estoque/javascript/movimento.js');</script>
<form name="FRM_historicomovimento"
          id="FRM_historicomovimento" action="index.php5"
          method="post" onsubmit="return objForm.estaOk();">
     <script>
          objForm.construct();
     </script>
     <input value="/sistema_web/estoque/mov_estorno_saida.php5"
          name="SCRIPT_NAME" id="SCRIPT_NAME" type="hidden">
     <input value="SALVAR" name="ACAO" id="ACAO" type="hidden">
     <input value="INC" name="FUNCAO" id="FUNCAO" type="hidden">
     <input value="P_0000000001" name="PRODUTO" id="PRODUTO" type="hidden">
     <input value="1" name="LOCAL" id="LOCAL" type="hidden">
     <input value="12-05-2009" name="DATA" id="DATA" type="hidden">
     <input value="12" name="SEQUENCIA" id="SEQUENCIA" type="hidden">
     <input value="0,90" name="VALOR_UNITARIO" id="VALOR_UNITARIO" type="hidden">
     <input value="85,000" name="QUANTIDADE" id="QUANTIDADE" type="hidden">
     <script>
          document.getElementById('PRODUTO_CODIGO').focus();
     </script>
     ...
</form>
<script>
     objBP.FEstorno=true;
     document.getElementById('PRODUTO_CODIGO').value='P_0000000001';
     document.getElementById('LOCAL_CODIGO').value='1';
     document.getElementById('HISTORICO_SEQUENCIA').value='12';
```

```
    document.getElementById('HISTORICO_DATA').value='12-05-2009';
    document.getElementById('PRODUTO').value='P_0000000001';
    document.getElementById('LOCAL').value='1';
    document.getElementById('SEQUENCIA').value='12';
    document.getElementById('DATA').value='12-05-2009';
    document.getElementById('HISTORICO_DOCUMENTO').value='';
    document.getElementById('HISTORICO_QUANTIDADE').value='85,000';
    document.getElementById('QUANTIDADE').value='85,000';
    document.getElementById('HISTORICO_VALOR_UNIT').value='0,90';
    document.getElementById('VALOR_UNITARIO').value='0,90';
    document.getElementById('HISTORICO_VALOR_TOTAL').value='76,50';
    document.getElementById('HISTORICO_HISTORICO').innerHTML=
                                            'Estorno do Documento:20090512';
    document.getElementById('PRODUTO_DESC').value='Refrigerante Cola 600ML';
    document.getElementById('PRODUTO_ESTOQUE').value='1700,000';
    document.getElementById('PRODUTO_CUSTOMEDIO').value='0,90';
    document.getElementById('LOCAL_ESTOQUE').value='1700,000';
</script>
</div>
```

Para verificar o impacto na tabela de produtos, vamos primeiro listar a situação atual do produto, efetuar o estorno de um lançamento de saída e em seguida visualizar a nova situação do produto.

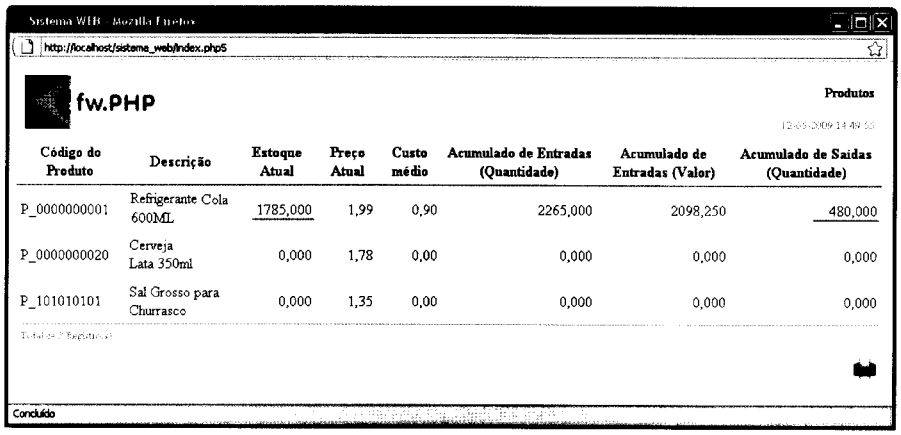
A Figura 7.7 mostra a situação atual do produto, antes do estorno do movimento de saída.



Código do Produto	Descrição	Estoque Atual	Preço Atual	Custo médio	Acumulado de Entradas (Quantidade)	Acumulado de Entradas (Valor)	Acumulado de Saídas (Quantidade)
P_0000000001	Refrigerante Cola 600ML	1700,000	1,99	0,90	2265,000	2098,250	565,000
P_0000000020	Cerveja Lata 350ml	0,000	1,78	0,00	0,000	0,000	0,000
P_101010101	Sal Grosso para Churrasco	0,000	1,35	0,00	0,000	0,000	0,000

Figura 7.7

Vamos estornar o lançamento da Figura 7.6, ou seja, vamos devolver 85 unidades do produto ao estoque. A Figura 7.8 ilustra como ficou o produto após o estorno.



Código do Produto	Descrição	Estoque Atual	Preço Atual	Custo médio	Acumulado de Entradas (Quantidade)	Acumulado de Entradas (Valor)	Acumulado de Saidas (Quantidade)
P_0000000001	Refrigerante Cola 600ML	1785,000	1,99	0,90	2265,000	2098,250	480,000
P_0000000020	Cerveja Lata 350ml	0,000	1,78	0,00	0,000	0,000	0,000
P_101010101	Sal Grosso para Churrasco	0,000	1,35	0,00	0,000	0,000	0,000

Figura 7.8

Capítulo 8

Relatórios

A criação de qualquer sistema, do mais simples ao mais complexo, exige sempre que sejam disponibilizados alguns relatórios operacionais e outros gerenciais.

Na versão original do framework **fw.PHP** temos disponíveis ferramentas para a criação de relatórios operacionais, especificamente os relatórios originados dos cadastros. Mas na esmagadora maioria dos sistemas isso não basta para prover os usuários de informações suficientes para o melhor aproveitamento do sistema e das informações disponíveis nele. Pensando nisso, criamos já no primeiro capítulo do livro uma extensão do framework para que seja possível a implementação de um módulo de relatórios no sistema.

Cada relatório deve ser uma classe derivada da classe **base** ou de outra classe qualquer que seja herdeira da classe **base** (não importando em qual nível). A classe deve descrever os campos do relatório. Os campos não precisam necessariamente pertencer a uma única tabela, nem mesmo serem campos que existam nas tabelas, podem ser expressões, combinações de outros campos ou qualquer operador desejado e necessário para o relatório.

A classe deve também separar corretamente quais campos pertencem ao relatório e quais atuam como filtros na geração dele.

Outro ponto importante é que toda classe que implemente um relatório deve, obrigatoriamente, reescrever o método *processaAcao()* da classe **base**, fazendo com que somente o processamento do relatório esteja disponível, ou seja, a única opção disponível deve ser a exibição do filtro do relatório.

É necessário, ainda, que a classe redefina, se necessário (na maioria das vezes será), o método *montaSELECT()*. Uma vez que em geral precisamos definir um comando SQL do tipo SELECT mais refinado e apropriado para o relatório, o que muitas vezes envolve a montagem de JOINTS (junções) entre várias tabelas, o que, naturalmente, não é suportado pelo método original da classe **base**.

Além disso pode ser necessário implementar o método *processaRelatorio()* para o pré-processamento de cada registro gerado no relatório e muito provavelmente precisaremos definir o método *posRelatorio()* o qual é chamado no encerramento do relatório, através do qual podemos gerar totais, médias e outras informações necessárias ao relatório em questão. Lembrando que é possível a geração de várias linhas extras, quantas sejam necessárias pelo relatório.

O módulo de relatórios utiliza um recurso novo do framework (também descrito no capítulo 1 do livro) desenvolvido para que seja possível que alguns botões sejam escondidos quando uma página desse tipo for executada. No caso dos relatórios devemos inibir os botões de impressão e filtro (em outros módulos será necessária a inibição de outros botões). O processo para inibir os botões é executado no módulo *geraFormularioFiltroRelatorio()*, o qual executa o método *inibebotoes()* da classe **base**. Já o processo normal de execução do framework executa o método *habilitaBotoes()* da mesma classe **base**.

Neste capítulo vamos desenvolver os seguintes relatórios para o sistema de controle de estoque:

- Saldo de produtos
- Estoque por departamento
- Estoque por grupo
- Estoque por local
- Estoque por produto e local
- Produtos com estoque acima do máximo
- Produtos com estoque abaixo do mínimo
- Extrato de movimentação
- Razão de estoque
- Curva ABC de estoque

Cada um desses relatórios será uma classe do sistema, construída com base nas regras mostradas anteriormente para o desenvolvimento de relatórios no framework. Cada classe terá seu grupo de filtros e estrutura própria.

O relatório de saldo de produtos mostra uma foto instantânea dos estoques no momento da emissão do relatório.

Os relatórios de estoque por departamento e estoque por grupo exibirão resumos dos estoques consolidados por departamento e por grupo, separando os estoques por unidade de medida, evitando que sejam somadas quantidades de referências distintas (quilo com litro, por exemplo).

Produtos com estoque acima do máximo exibe a lista de produtos cujos estoques estão acima dos estoques máximos cadastrados no sistema.

Produtos com estoque abaixo do mínimo mostra a relação de produtos cujos estoques estão abaixo do mínimo cadastrado no sistema.

O extrato de movimentação apresenta os movimentos de estoque em um determinado período agregando os movimentos por produto.

Razão de estoque é uma versão mais elaborada do relatório de extrato de movimentação, apresentando o controle de saldo de estoque, tanto inicial quanto final de cada produto. Por isso mesmo tende a ser um relatório mais demorado.

O relatório curva ABC de estoque utiliza a classificação ABC, em que os produtos são classificados como A, B ou C conforme o seu consumo médio mensal em um determinado período (medido em meses). Os produtos classificados como A representam 80% do total, os classificados como B representam 15% do total e os classificados como C representam os 5% restantes.

Saldo de produtos

Este é provavelmente o primeiro relatório que qualquer desenvolvedor imagina quando pensa em um sistema de controle de estoque, pois sempre é necessário ao gerente do estoque saber como está o estoque atual de um ou mais produtos. O relatório de saldo de produtos mostra uma fotografia instantânea dos estoques em um determinado momento (muito provavelmente um novo relatório emitido minutos depois mostrará uma nova situação dos estoques) e deve ser utilizado como tal, ou seja, como uma ferramenta de análise do passado recente.

O relatório deve permitir os seguintes filtros:

- Produto inicial
- Produto final
- Departamento
- Grupo de produtos

Os seguintes campos devem ser exibidos no relatório:

- Produto
- Descrição
- Unidade de medida
- Departamento
- Grupo de produtos
- Estoque atual
- Preço atual
- Valor atual do estoque
- Custo médio atual
- Valorização do estoque pelo custo médio

No final do relatório devemos exibir:

- Total do valor atual do estoque
- Total da valorização do estoque pelo custo médio

Devemos ter cuidado ao definir a classe para que campos do filtro não sejam exibidos no relatório e para que campos exclusivos do relatório não sejam exibidos no formulário de filtragem.

Cada um dos campos, tanto de filtro quanto de relatório, deve ser devidamente definido no construtor da classe.

1. PRODUTO_CODIGO

   Instância da classe **string**, define o código do produto que será exibido no relatório e também define o campo 'Produto inicial' no formulário de filtragem. Esse campo deve ter seu atributo 'comportamento form' definido como 'ajax' no formulário e seu título alterado para 'Produto' no momento da exibição do relatório.

2. PRODUTO_CODIGO_F

   Instância da classe **string**, define o código do produto final no filtro de registros do relatório. Esse campo deve ser utilizado apenas no formulário de filtragem, não sendo exibido no relatório. O atributo 'comportamento form' desse campo deve ser definido como 'ajax'.

3. DEPTO_CODIGO

   Instância da classe **string**, define o departamento que será filtrado no relatório. Não aparece no relatório, sendo exibida no seu lugar a descrição do departamento. O atributo 'comportamento form' desse campo deve ser definido como 'select' e seu conteúdo gerado antes da exibição do formulário de filtragem. A lista de opções deve ter a opção-padrão 'Todos' para que seja possível ignorar o filtro.

4. GRUPO_CODIGO

   Instância da classe **string**, define o grupo de produtos que será filtrado no relatório. Esse campo não aparece no relatório, sendo exibida no seu lugar a descrição do grupo. O atributo 'comportamento form' desse campo deve ser definido como 'select' e seu conteúdo gerado antes da exibição do formulário de filtragem. A lista de opções deve ter a opção-padrão 'Todos' para que seja possível ignorar o filtro.

5. PRODUTO_DESC

   Instância da classe **string**, esse campo é utilizado para a descrição do produto no relatório, não sendo exibido no formulário de filtragem.

6. UNIDADE_CODIGO

   Instância da classe **string**, é utilizado para a descrição do produto no relatório, não sendo exibido no formulário de filtragem.

7. DEPTO_DESC

   Instância da classe **string**, esse campo é utilizado para a descrição do departamento no relatório, não sendo exibido no formulário de filtragem. Ele substitui o campo DEPTO_CODIGO utilizado no filtro do relatório.

1. GRUPO_DESC

   Instância da classe **string**, é utilizado para a descrição do grupo de produtos no relatório, não sendo exibido no formulário de filtragem. Ele substitui o campo GRUPO_CODIGO definido no filtro do relatório.

2. PRODUTO_ESTOQUE

   Instância da classe **float**, define o Estoque atual do produto. Este campo aparece apenas no relatório.

3. PRODUTO_PRECO

   Instância da classe **dinheiro**, contém o preço atual do produto e é exibido apenas no relatório.

4. VALOR_ESTOQUE

   Instância da classe **dinheiro**, esse campo não existe na tabela de produtos e deve ser calculado, sendo sua fórmula:

```
VALOR_ESTOQUE = PRODUTO_ESTOQUE * PRODUTO_PRECO
```

   Esse campo é totalizado no final do relatório.

5. PRODUTO_CUSTOMEDIO

   Instância da classe **dinheiro**, contém o custo médio atual do produto e é exibido apenas no relatório.

6. VALOR_ESTOQUE_CM

   Instância da classe **dinheiro**, esse campo não existe na tabela de produtos e deve ser calculado, sendo sua fórmula:

```
VALOR_ESTOQUE_CM = PRODUTO_ESTOQUE * PRODUTO_CUSTOMEDIO
```

   Esse campo é totalizado no final do relatório.

O método *__construct()* da classe, ou seja, seu construtor, define todos os campos listados e ainda separa os campos que são filtros dos que são parte do relatório. A definição de quem é quem se dá através do ajuste dos atributos *$_filtro* e *$_relatorio* da

instância da classe **campo** de cada um dos campos da classe (não acessamos diretamente os atributos, em vez disso utilizamos os métodos virtuais *setFiltro* e *setRelatorio*). Nesse método definimos também o atributo 'comportamento form' dos campos relacionados ao código do produto, ou seja, PRODUTO_CODIGO e PRODUTO_CODIGO_F. Esses campos terão o comportamento ajustado para 'ajax', permitindo que seja executada uma busca dinâmica nas tabelas relacionadas. Como a classe que realiza a busca é a **produto**, precisamos ajustar também os atributos desses campos que definem a classe e o arquivo que contém a classe.

```
$this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
$this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
$this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                                  "estoque/classes/classe_produto.inc");
$this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
$this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
$this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                                  "estoque/classes/classe_produto.inc");
```

O método *processaAcao()* deve ser modificado para que o processamento padrão do framework (exibir a lista de registros e disponibilizar as opções de manutenção) seja alterado. No caso de relatórios queremos que sempre seja exibido o formulário de filtro (queremos também que alguns botões sejam inibidos) o qual executa a página de impressão de dados. No método *processaAcao()* vamos preencher a lista de valores dos campos DEPTO_CODIGO e GRUPO_CODIGO, além de ajustar o atributo 'comportamento form' desses campos para 'select'. A busca dos dados dar-se-á pela utilização do método *buscar()* de cada uma das classes correspondentes (elas devem ter sido instanciadas primeiramente). Ambos os campos receberão a opção-padrão 'TODOS' como primeira opção das listas.

No final é executado o método *geraFormularioFiltroRelatorio()* da classe **base** para que seja possível a exibição do formulário de filtragem, além da inibição dos botões de impressão e filtro de registros.

```
/**
 * Sempre executamos o módulo de relatório especial
 *
 */
public function processaAcao() {
    // Sempre mostra o filtro do relatório nada mais...
    $_depto = new departamento($this->_conn);
    $_depto->Buscar();
    $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
    $_ldep[] = Array("valor"=>'TODOS',"label"=>'TODOS');
    while($_depto->proximo()!==false) {
        $_ldep[] = Array("valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                         "label"=>$_depto->getCampo('DEPTO_DESC')->getValor());
    }
    $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
    $_grupo = new grupo($this->_conn);
    $_grupo->Buscar();
    $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
    $_lgrp[] = Array("valor"=>'TODOS',"label"=>'TODOS');
    while($_grupo->proximo()!==false) {
```

```
            $_lgrp[] = Array("valor"=>$_grupo->getCampo('GRUPO_CODIGO')->getValor(),
                             "label"=>$_grupo->getCampo('GRUPO_DESC')->getValor());
        }
        $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
        $this->geraFormularioFiltroRelatorio();
}
```

Nos relatórios, via de regra devemos modificar o método *montaSELECT()* da classe **base** para que seja possível a definição de comandos SQL de recuperação de dados mais complexos, uma vez que muitas vezes serão necessárias a junção de várias tabelas e a construção de expressões na busca dos dados. Este é o caso dessa classe, pois precisamos tanto de junções (JOINTS) quanto de expressões. As junções serão utilizadas para a busca da descrição dos departamentos e grupos, já as expressões para o cálculo dos valores totais dos estoques, tanto a preço atual quanto a custo médio. Além disso devemos aplicar os filtros definidos no formulário inicial do relatório.

```
/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                            $_orderby=null,$_limit=null,$_extras=null) {
    $this->getCampo("PRODUTO_CODIGO")->setTitulo("Produto");
    $_strSQL= "SELECT " .
                "PRODUTO_CODIGO, PRODUTO_DESC, D.DEPTO_DESC," .
                "G.GRUPO_DESC,PRODUTO_ESTOQUE,PRODUTO_PRECO," .
                "UNIDADE_CODIGO,PRODUTO_CUSTOMEDIO, " .
                "PRODUTO_PRECO*PRODUTO_ESTOQUE AS VALOR_ESTOQUE," .
                "PRODUTO_CUSTOMEDIO*PRODUTO_ESTOQUE AS VALOR_ESTOQUE_CM " .
                "FROM {$this->_nome_tabela} P    " .
                "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
                "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO ";
    // Filtro
        $_where = Array();
        $_operador = Array('PRODUTO_CODIGO'=>'>=','PRODUTO_CODIGO_F'=>'<=');
        $_alias    = Array('PRODUTO_CODIGO_F'=>'PRODUTO_CODIGO');
        foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
            if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
                $_campo->setValor($_POST[$_nome]);
                $_where[] = "P." . (array_key_exists($_nome,$_alias)
                                    ? $_alias[$_nome]
                                    : $_nome) .
                                   (array_key_exists($_nome,$_operador)
                                    ? $_operador[$_nome]
                                    : '=') .
                                   $_campo->toBD();
            }
        }   if(sizeof($_where)>0) {
        $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
    }
    $_strSQL .=   "ORDER BY P.PRODUTO_CODIGO";
    return $_strSQL;
}
```

O método *processaRelatorio()* é utilizado para a acumulação dos totais da valorização dos estoques, seja a preço atual ou com valor de custo médio dos produtos. Para guardar essa acumulação devemos ter um atributo novo na classe, específico para essa

função. Esses acumulados são utilizados no final do relatório, onde os totais serão exibidos.

```
/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
public function processaRelatorio() {
     // Vamos acumular os valores
     $this->_acumulado['VALOR_ESTOQUE'] +=
                                   $this->getCampo("VALOR_ESTOQUE")->getValor();
     $this->_acumulado['VALOR_ESTOQUE_CM'] +=
                                   $this->getCampo("VALOR_ESTOQUE_CM")->getValor();
}
```

O último método que devemos definir na classe é o que fará o fechamento do relatório, exibindo as linhas finais dele (na verdade temos ainda no final uma linha com o total de registros do relatório). Utilizamos esse método para a impressão dos totais do relatório, que neste caso serão dois, o primeiro representando o acumulado da valorização do estoque a preço atual dos produtos e o segundo com a valorização a custo médio dos produtos. A utilização do método *posRelatorio()* exige que o retorno seja uma lista multidimensional com várias linhas, e cada linha deve conter uma lista de campos (instâncias da classe **campo**) e uma lista de atributos extras para cada campo definido. No caso específico desse relatório precisamos apenas de uma linha de totais.

```
/**
 * Exibe os totais do relatório
 * Valor de estoques a preço atual
 * Valor de estoques a custo médio
 *
 * @return array
 */
public function posRelatorio() {
     $_extra = new atributo("STYLE","height:40px;color:Navy;font-weight:bold;");
     $_extra_2 = new atributo("STYLE","height:40px;color:Navy;" .
                              "font-weight:bold;border-top:2px Solid Navy;");
     $_campos =     Array(
                         Array(
                              Array(
                                   new string("","",0,null,null),
                                   new string("","",0,null,null),
                                   new string("","",0,null,null),
                                   new string("","",0,null,null),
                                   new string("","",0,null,null),
                                   new string("","",0,null,null),
                                   new string("","",0,null,null),
                                   new dinheiro("VALOR_ESTOQUE","",12,null,null),
                                   new string("","",0,null,null),
                                   new dinheiro("VALOR_ESTOQUE_CM","",12,null,null)
                              ),
                              Array(
                                   $_extra,
                                   7=>$_extra_2,
                                   9=>$_extra_2
                              )
                         )
                    );
     $_campos[0][0][0]->setValor('Totais');
     $_campos[0][0][7]->setValor($this->_acumulado['VALOR_ESTOQUE']);
```

```
        $_campos[0][0][9]->setValor($this->_acumulado['VALOR_ESTOQUE_CM']);
        return $_campos;
}
```

Isso fecha a definição da classe para emissão do relatório de saldo de estoque de produtos.

Chamaremos a classe de **estoqueatual**, a qual deve ser gravada no arquivo *classe_estoqueatual*, o qual deve, assim como todas as demais classes desse sistema, estar no diretório *estoque/classes/*.

Lista 1: classe_estoqueatual.inc

```
<?php
/**
 * Classe Estoque Atual
 */
class estoqueatual extends base {
    protected $_acumulado = array();

    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'produto';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("PRODUTO_CODIGO","Produto Inicial",
                        20,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("PRODUTO_CODIGO_F","Produto Final",
                        20,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("DEPTO_CODIGO","Departamento",
                        10,null,null,true,true,true,1,null,null,false,$_conn));
        $this->addCampo(new string("GRUPO_CODIGO","Grupo",
                        10,null,null,true,true,true,1,null,null,false,$_conn));
        // Campos somente do relatório
        $this->addCampo(new string("PRODUTO_DESC","Descrição",60,null,null));
        $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
        $this->addCampo(new string("DEPTO_DESC","Departamento",40,null,null));
        $this->addCampo(new string("GRUPO_DESC","Grupo",40,null,null));
        $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",
                                  12,null,null));
        $this->addCampo(new dinheiro("PRODUTO_PRECO","Preço Atual",
                                  12,null,null));
        $this->addCampo(new dinheiro("VALOR_ESTOQUE","Valor Estoque",
                                  12,null,null));
        $this->addCampo(new dinheiro("PRODUTO_CUSTOMEDIO","Custo médio",
                                  12,null,null));
        $this->addCampo(new dinheiro("VALOR_ESTOQUE_CM","Valor Estoque (CM)",
                                  12,null,null));
        // Acertar filtro
        $this->getCampo("PRODUTO_DESC")->setFiltro(false);
        $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
        $this->getCampo("DEPTO_DESC")->setFiltro(false);
        $this->getCampo("GRUPO_DESC")->setFiltro(false);
        $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
        $this->getCampo("PRODUTO_PRECO")->setFiltro(false);
        $this->getCampo("VALOR_ESTOQUE")->setFiltro(false);
        $this->getCampo("PRODUTO_CUSTOMEDIO")->setFiltro(false);
        $this->getCampo("VALOR_ESTOQUE_CM")->setFiltro(false);
        // Acertar Campos do relatório
        $this->getCampo("PRODUTO_CODIGO_F")->setRelatorio(false);
        $this->getCampo("DEPTO_CODIGO")->setRelatorio(false);
        $this->getCampo("GRUPO_CODIGO")->setRelatorio(false);
        $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
```

```
        $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                                        "estoque/classes/classe_produto.inc");
        $this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                                        "estoque/classes/classe_produto.inc");
    }

    /**
     * Sempre executamos o módulo de relatório especial
     *
     */
    public function processaAcao() {
        // Sempre mostra o filtro do relatório nada mais...
        $_depto = new departamento($this->_conn);
        $_depto->Buscar();
        $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
        $_ldep[] = Array("valor"=>'TODOS',"label"=>'TODOS');
        while($_depto->proximo()!==false) {
            $_ldep[] = Array(
                    "valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                    "label"=>$_depto->getCampo('DEPTO_DESC')->getValor());
        }
        $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
        $_grupo = new grupo($this->_conn);
        $_grupo->Buscar();
        $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
        $_lgrp[] = Array("valor"=>'TODOS',"label"=>'TODOS');
        while($_grupo->proximo()!==false) {
            $_lgrp[] = Array(
                    "valor"=>$_grupo->getCampo('GRUPO_CODIGO')->getValor(),
                    "label"=>$_grupo->getCampo('GRUPO_DESC')->getValor());
        }
        $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
        $this->geraFormularioFiltroRelatorio();
    }

    /**
     * Uma classe de relatórios deve sempre montar seu select
     *
     */
    public function montaSELECT($_campos,$_where=null,
                                $_orderby=null,$_limit=null,$_extras=null) {
        $this->getCampo("PRODUTO_CODIGO")->setTitulo("Produto");
        $_strSQL= "SELECT " .
                  "PRODUTO_CODIGO, PRODUTO_DESC, D.DEPTO_DESC," .
                  "G.GRUPO_DESC,PRODUTO_ESTOQUE,PRODUTO_PRECO," .
                  "UNIDADE_CODIGO,PRODUTO_CUSTOMEDIO, " .
                  "PRODUTO_PRECO*PRODUTO_ESTOQUE AS VALOR_ESTOQUE," .
                  "PRODUTO_CUSTOMEDIO*PRODUTO_ESTOQUE AS VALOR_ESTOQUE_CM " .
                  "FROM {$this->_nome_tabela} P    " .
                  "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
                  "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO ";
        // Filtro
        $_where = Array();
        $_operador = Array('PRODUTO_CODIGO'=>'>=','PRODUTO_CODIGO_F'=>'<=');
        $_alias    = Array('PRODUTO_CODIGO_F'=>'PRODUTO_CODIGO');
        foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
            if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
                $_campo->setValor($_POST[$_nome]);
                $_where[] = "P." . (array_key_exists($_nome,$_alias)
                                        ? $_alias[$_nome]
                                        : $_nome) .
                                    (array_key_exists($_nome,$_operador)
```

```
                                        ? $_operador[$_nome]
                                        : '=') .
                                    $_campo->toBD();
                }
        }           if(sizeof($_where)>0) {
                $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
        }
        $_strSQL .=    "ORDER BY P.PRODUTO_CODIGO";
        return $_strSQL;
    }

    /**
     * Devemos pré-processar o relatório antes de ser gerado o html
     *
     */
    public function processaRelatorio() {
        // Vamos acumular os valores
        $this->_acumulado['VALOR_ESTOQUE'] +=
                                $this->getCampo("VALOR_ESTOQUE")->getValor();
        $this->_acumulado['VALOR_ESTOQUE_CM'] +=
                                $this->getCampo("VALOR_ESTOQUE_CM")->getValor();
    }

    /**
     * Exibe os totais do relatório
     * Valor de estoques a preço atual
     * Valor de estoques a custo médio
     *
     * @return array
     */
    public function posRelatorio() {
        $_extra = new atributo("STYLE","height:40px;color:Navy;" .
                                "font-weight:bold;");
        $_extra_2 = new atributo("STYLE","height:40px;color:Navy;" .
                                "font-weight:bold;border-top:2px Solid Navy;");
        $_campos = Array(
                    Array(
                        Array(
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new dinheiro("VALOR_ESTOQUE","",12,null,null),
                            new string("","",0,null,null),
                            new dinheiro("VALOR_ESTOQUE_CM","",12,null,null)
                        ),
                        Array(
                            $_extra,
                            7=>$_extra_2,
                            9=>$_extra_2
                        )
                    )
                );
        $_campos[0][0][0]->setValor('Totais');
        $_campos[0][0][7]->setValor($this->_acumulado['VALOR_ESTOQUE']);
        $_campos[0][0][9]->setValor($this->_acumulado['VALOR_ESTOQUE_CM']);
        return $_campos;
    }
}
?>
```

Definida a classe, precisamos elaborar o programa que será executado pelo framework. Como já é sabido, esse programa é simples, necessitando apenas de poucas linhas de codificação. Chamaremos o arquivo de *rel_estoqueatual.php5*. Veja sua listagem a seguir.

Lista 2: rel_estoqueatual.php5

```
<?php
/**
 * Relatório: Saldo Atual de estoque
 *
 */
$_classe =     Array(
                    "arquivo"=>"estoque/classes/classe_estoqueatual.inc",
                    "nome"=>"estoqueatual"
               );
$_FTitulo = "Relatório de Saldo de estoque de produtos";
return include_once("../instanciaclasse.php5");
?>
```

O último passo para a execução do relatório é a inclusão do programa no menu do sistema, o que deve ser feito pela opção 'Gerenciador de menus' do menu Administração. Como este é o primeiro relatório que criamos, devemos adicionar primeiro o menu principal de relatório, o qual estará subordinado ao menu principal do sistema de controle de estoque. Em seguida acrescentamos o menu do relatório que acabamos de desenvolver.

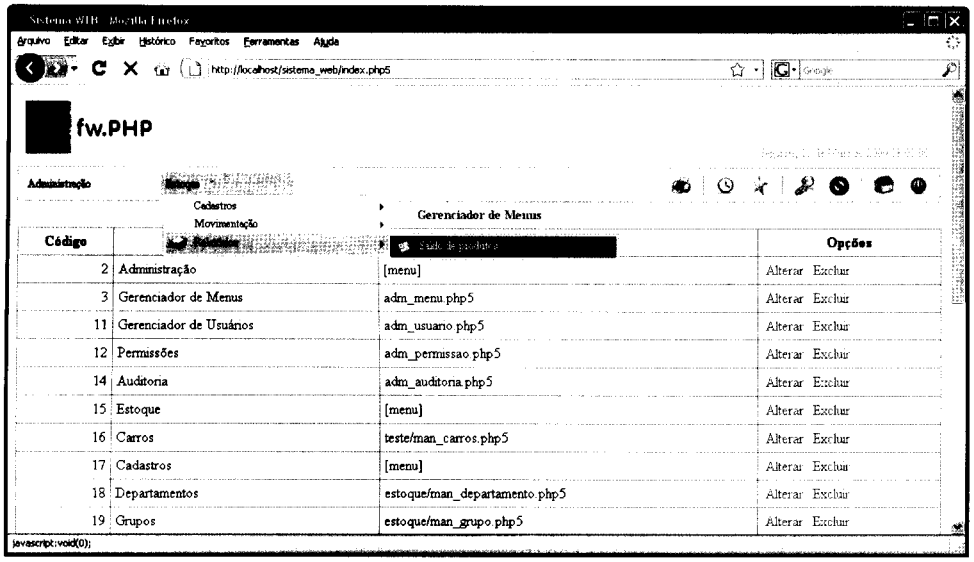


Figura 8.1

As Figuras 8.2 e 8.3 exibem o formulário inicial do relatório para definição dos critérios de filtragem e a página com o relatório respectivamente.

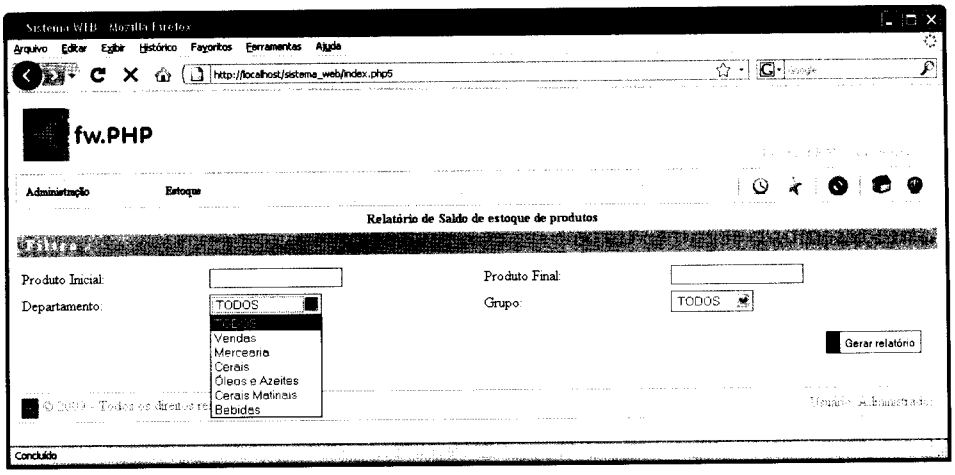
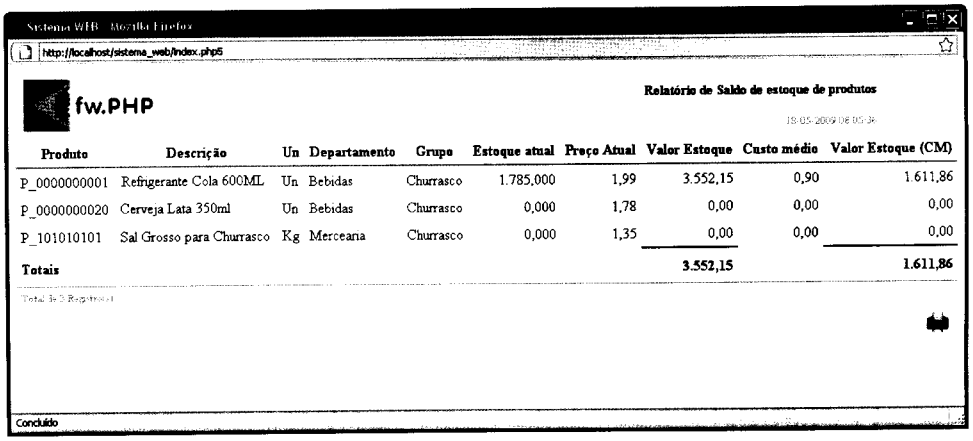


Figura 8.2



Produto	Descrição	Un	Departamento	Grupo	Estoque atual	Preço Atual	Valor Estoque	Custo médio	Valor Estoque (CM)
P_0000000001	Refrigerante Cola 600ML	Un	Bebidas	Churrasco	1.785,000	1,99	3.552,15	0,90	1.611,86
P_0000000020	Cerveja Lata 350ml	Un	Bebidas	Churrasco	0,000	1,78	0,00	0,00	0,00
P_101010101	Sal Grosso para Churrasco	Kg	Merceana	Churrasco	0,000	1,35	0,00	0,00	0,00
Totais							**3.552,15**		**1.611,86**

Figura 8.3

Estoque por departamento

O objetivo desse relatório é exibir um resumo do estoque por departamento, não importando o estoque individual dos produtos. Uma vez que um departamento pode conter vários produtos e cada produto pode ter uma unidade de medida diferente, devemos considerar o estoque por unidade de medida, assim não teremos a mistura de valores de medidas diferentes.

Como a estrutura de departamentos proposta para o sistema é o de árvore, devemos preparar a classe do relatório para que, uma vez informado um departamento, todos os departamentos relacionados sejam considerados no relatório (ou seja, dado um ramo da árvore, devemos considerar todos os galhos e folhas desse ramo). Para

implementar essa busca, precisamos alterar a classe **departamento** incluindo um método que retorne a lista de todos os departamentos relacionados a um departamento dado.

A classe deve acumular por unidade de medida o total de estoque, a valorização a preço atual e a valorização a custo de reposição.

Assim teremos o seguinte filtro para esse relatório:

- Departamento

O relatório deve conter os seguintes campos:

- Código do departamento
- Descrição do departamento
- Unidade de medida
- Estoque total
- Valor do estoque a preço atual
- Valor do estoque a custo de reposição (custo atual)

E como totais, por unidade de medida, teremos:

- Estoque total
- Valor do estoque a preço atual
- Valor do estoque a custo de reposição (custo atual)

Todos os campos, tanto de filtro quanto do relatório (se diferentes), devem ser definidos no construtor da classe.

1. DEPTO_CODIGO

   Instância da classe **string**, define o departamento que será filtrado no relatório. O atributo 'comportamento form' desse campo deve ser definido como 'select' e seu conteúdo gerado antes da exibição do formulário de filtragem. A lista de opções deve ter a opção-padrão 'Todos' para que seja possível ignorar o filtro. Somente esse campo será exibido no formulário de filtragem.

2. DEPTO_DESC

   Instância da classe **string**, é utilizado para a descrição do departamento no relatório, não sendo exibido no formulário de filtragem.

3. UNIDADE_CODIGO

   Instância da classe **string**, esse campo é utilizado para separação dos estoques do departamento, evitando que unidades diferentes sejam somadas. Ele não será exibido no formulário de filtragem.

4. PRODUTO_ESTOQUE

   Instância da classe **float**, contém o estoque total da relação departamento e unidade de medida. Esse campo aparece apenas no relatório.

5. VALOR_ESTOQUE

   Instância da classe **dinheiro**, ele não existe na tabela de produtos e deve ser calculado, sendo sua fórmula:

```
VALOR_ESTOQUE = PRODUTO_ESTOQUE * PRODUTO_PRECO
```

   Esse campo é totalizado no final do relatório por unidade de medida.

6. VALOR_ESTOQUE_RP

   Instância da classe **dinheiro**, ele não existe na tabela de produtos e deve ser calculado, sendo sua fórmula:

```
VALOR_ESTOQUE_RP = PRODUTO_ESTOQUE * PRODUTO_CUSTOATUAL
```

   Esse campo é totalizado no final do relatório por unidade de medida

O método construtor da classe *__construct()* deve definir os campos listados anteriormente, separando os campos que são filtros dos que são parte apenas do relatório.

```
public function __construct(BancoDados $_conn) {
     parent::__construct($_conn);
     $this->_nome_tabela = 'produto';
     $this->_class_path = 'estoque';
     $this->addCampo(new string("DEPTO_CODIGO","Departamento",
                         10,null,null,true,true,true,1,null,null,false,$_conn));
     $this->addCampo(new string("DEPTO_DESC","Descrição",40,null,null));
     $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
     $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",12,null,null));
     $this->addCampo(new dinheiro("VALOR_ESTOQUE","Valor Estoque",12,null,null));
     $this->addCampo(new dinheiro("VALOR_ESTOQUE_RP",
                         "Valor Estoque (C.Reposição)",12,null,null));
     // Acertar filtro
     $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
     $this->getCampo("DEPTO_DESC")->setFiltro(false);
     $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
     $this->getCampo("VALOR_ESTOQUE")->setFiltro(false);
     $this->getCampo("VALOR_ESTOQUE_RP")->setFiltro(false);
}
```

Assim como foi feito na classe anterior, devemos sobrepor o método *processaAcao()* da classe **base**, alterando o comportamento do framework. Nessa classe vamos preencher a lista de opções do campo DEPTO_CODIGO, além de alterar o atributo 'comportamento form' desse campo para 'select'. Finalmente executamos o método *geraFormularioFiltroRelatorio()* para que o formulário de filtragem seja exibido.

```
/**
 * Sempre executamos o módulo de relatório especial
 *
 */
public function processaAcao() {
     // Sempre mostra o filtro do relatório nada mais...
     $_depto = new departamento($this->_conn);
     $_depto->Buscar();
     $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
     $_ldep[] = Array("valor"=>'TODOS',"label"=>'TODOS');
     while($_depto->proximo()!==false) {
          $_ldep[] = Array("valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                           "label"=>$_depto->getCampo('DEPTO_DESC')->getValor());
     }
     $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
     $this->geraFormularioFiltroRelatorio();
}
```

O método *montaSELECT()* deve ser reescrito para que o relatório seja emitido corretamente. A primeira tarefa que devemos executar no método é a busca, caso seja definido um departamento como filtro, de todos os departamentos relacionados ao departamento selecionado. Essa busca deve ser realizada na classe **departamento**, o que exige a implementação desse método nessa classe. Chamaremos o método de busca dos departamentos relacionados de *departamentosfilhos()* e sua implementação está listada a seguir.

```
/**
 * Retorna uma lista com todos os departamentos relacionados ao departamento
 * informado inicialmente
 * A lista retornada é simples, não levando em consideração a posição
 * dos departamentos na árvore
 *
 * @param string $_depto_pai
 * @return array
 */
public function departamentosfilhos($_depto_pai) {
     $_lista = Array();
     do {
          $_strSQL = "SELECT DEPTO_CODIGO FROM {$this->_nome_tabela} " .
                     "WHERE DEPTO_PAI IN({$_depto_pai})";
          $_parar = true;
          if($this->_conn->executaSQL($_strSQL)!==false
             &&$this->_conn->getNumRows()>0) {
               $_arr = Array();
               while(($_dados=$this->_conn->proximo())!==false) {
                    $_arr[] = "'{$_dados['DEPTO_CODIGO']}'";
                    $_lista[] = "'{$_dados['DEPTO_CODIGO']}'";
               }
               $_parar = false;
               $_depto_pai = implode(",",$_arr);
          }
     } while($_parar===false);
     return $_lista;
}
```

A opção foi por um método não recursivo para implementar a busca dos departamentos (seria perfeitamente possível a construção desse método utilizando a recursividade), portanto a compreensão do que está sendo executado no método é mais

simples e rápida. Ao final do método teremos uma lista com todos os departamentos relacionados ao departamento inicialmente informado na chamada ao método.

A lista retornada será utilizada no método *montaSELECT()* para que somente os departamentos presentes na lista sejam exibidos no relatório.

O método retorna o comando SQL, necessário para a busca por departamento e unidade de medida dos estoques e suas valorizações, tanto a preço atual quanto a custo de reposição.

```
/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                            $_orderby=null,$_limit=null,$_extras=null) {
     // 1o. Buscar a lista de departamentos
     $_lista = null;
     if($_POST['DEPTO_CODIGO']!='TODOS') {
          $_depto = new departamento($this->_conn);
          $_arr = $_depto->departamentosfilhos("'{$_POST['DEPTO_CODIGO']}'");
          $_lista[] = "'{$_POST['DEPTO_CODIGO']}'";
          $_lista = array_merge($_lista,$_arr);
     }
     $_strSQL= "SELECT " .
               "P.DEPTO_CODIGO,D.DEPTO_DESC,P.UNIDADE_CODIGO," .
               "SUM(PRODUTO_ESTOQUE) AS PRODUTO_ESTOQUE," .
               "SUM(PRODUTO_ESTOQUE*PRODUTO_PRECO) AS VALOR_ESTOQUE," .
               "SUM(PRODUTO_CUSTOATUAL*PRODUTO_ESTOQUE) AS VALOR_ESTOQUE_RP " .
               "FROM {$this->_nome_tabela} P " .
               "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO";
     // Filtro
     if(sizeof($_lista)>0&&$_lista!==null) {
          $_strSQL .= " WHERE P.DEPTO_CODIGO IN(" . implode(",",$_lista) . ") ";
     }
     $_strSQL .=    " GROUP BY P.DEPTO_CODIGO,D.DEPTO_DESC,P.UNIDADE_CODIGO" .
                    " ORDER BY DEPTO_CODIGO,UNIDADE_CODIGO";
     return $_strSQL;
}
```

O comando lista todos os departamentos que estejam relacionados a produtos, não importando se os totais estão zerados ou não. Caso você ache melhor listar somente os totais, inclua no final do comando SQL, antes de 'ORDER BY', a expressão:

```
HAVING SUM(PRODUTO_ESTOQUE)>0
```

Com isso somente os departamentos com estoque maior que zero são exibidos no relatório.

No método *processaRelatorio()* os totais de estoque e suas valorizações são acumulados por unidade de medida. Esses acumulados são exibidos no final do relatório.

```
/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
public function processaRelatorio() {
     // Vamos acumular os valores
     $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
          ['PRODUTO_ESTOQUE'] += $this->getCampo("PRODUTO_ESTOQUE")->getValor();
     $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
          ['VALOR_ESTOQUE'] += $this->getCampo("VALOR_ESTOQUE")->getValor();
     $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
          ['VALOR_ESTOQUE_RP'] += $this->getCampo("VALOR_ESTOQUE_RP")->getValor();
}
```

Por último temos o método *posRelatorio()* que será utilizado para gerar os acumulados por unidade de medida. Cada unidade de medida ocupará uma linha do relatório.

```
/**
 * Exibe os totais do relatório
 * Valor de estoques a preço atual
 * Valor de estoques a custo reposição
 *
 * @return array
 */
public function posRelatorio() {
     $_extra = new atributo("STYLE","height:20px;color:Navy;font-weight:bold;");
     foreach($this->_acumulado as $_un=>$_acumulado) {
          $_linha = Array(
                         Array(
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("UNIDADE_CODIGO","",2,null,null),
                              new float("PRODUTO_ESTOQUE","",12,null,null),
                              new dinheiro("VALOR_ESTOQUE","",12,null,null),
                              new dinheiro("VALOR_ESTOQUE_RP","",12,null,null)
                         ),
                         Array(
                              $_extra,
                              $_extra,
                              $_extra,
                              $_extra,
                              $_extra,
                              $_extra
                         )
                    );
          $_linha[0][2]->setValor($_un);
          $_linha[0][3]->setValor($_acumulado['PRODUTO_ESTOQUE']);
          $_linha[0][4]->setValor($_acumulado['VALOR_ESTOQUE']);
          $_linha[0][5]->setValor($_acumulado['VALOR_ESTOQUE_RP']);
          $_campos[] = $_linha;
     }
     $_campos[0][0][0]->setValor('Totais');
     return $_campos;
}
```

Note que construímos os campos para cada linha de totais (ou seja para cada unidade de medida).

Pronto, a classe está terminada e vamos chamá-la de classe **estoquedepto**, sendo gravada no arquivo *classe_estoquedepto.inc* dentro do diretório *estoque/classes/*. A lista a seguir mostra a classe por inteiro.

Lista 3: classe_estoquedepto.inc

```
<?php
/**
 * Classe Estoque por Departamento
 */
class estoquedepto extends base {
     protected $_acumulado = array();

     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'produto';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("DEPTO_CODIGO","Departamento",
                         10,null,null,true,true,true,1,null,null,false,$_conn));
          $this->addCampo(new string("DEPTO_DESC","Descrição",40,null,null));
          $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
          $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",
                         12,null,null));
          $this->addCampo(new dinheiro("VALOR_ESTOQUE","Valor Estoque",
                         12,null,null));
          $this->addCampo(new dinheiro("VALOR_ESTOQUE_RP",
                         "Valor Estoque (C.Reposição)",12,null,null));
          // Acertar filtro
          $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
          $this->getCampo("DEPTO_DESC")->setFiltro(false);
          $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
          $this->getCampo("VALOR_ESTOQUE")->setFiltro(false);
          $this->getCampo("VALOR_ESTOQUE_RP")->setFiltro(false);
     }

     /**
      * Sempre executamos o módulo de relatório especial
      *
      */
     public function processaAcao() {
          // Sempre mostra o filtro do relatório nada mais...
          $_depto = new departamento($this->_conn);
          $_depto->Buscar();
          $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
          $_ldep[] = Array("valor"=>'TODOS',"label"=>'TODOS');
          while($_depto->proximo()!==false) {
               $_ldep[] = Array(
                         "valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                         "label"=>$_depto->getCampo('DEPTO_DESC')->getValor());
          }
          $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
          $this->geraFormularioFiltroRelatorio();
     }

     /**
      * Uma classe de relatórios deve sempre montar seu select
      *
      */
     public function montaSELECT($_campos,$_where=null,
                                 $_orderby=null,$_limit=null,$_extras=null) {
          // 1o. Buscar a lista de departamentos
```

```
        $_lista = null;
        if($_POST['DEPTO_CODIGO']!='TODOS') {
            $_depto = new departamento($this->_conn);
            $_arr = $_depto->departamentosfilhos("'{$_POST['DEPTO_CODIGO']}'");
            $_lista[] = "'{$_POST['DEPTO_CODIGO']}'";
            $_lista = array_merge($_lista,$_arr);
        }
        $_strSQL= "SELECT " .
                  "P.DEPTO_CODIGO,D.DEPTO_DESC,P.UNIDADE_CODIGO," .
                  "SUM(PRODUTO_ESTOQUE) AS PRODUTO_ESTOQUE," .
                  "SUM(PRODUTO_ESTOQUE*PRODUTO_PRECO) AS VALOR_ESTOQUE," .
                  "SUM(PRODUTO_CUSTOATUAL*PRODUTO_ESTOQUE) AS VALOR_ESTOQUE_RP" .
                  " FROM {$this->_nome_tabela} P " .
                  "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO";
        // Filtro
        if(sizeof($_lista)>0&&$_lista!==null) {
            $_strSQL .= " WHERE P.DEPTO_CODIGO IN(" .
                        implode(",",$_lista) . ") ";
        }
        $_strSQL .= " GROUP BY P.DEPTO_CODIGO,D.DEPTO_DESC," .
                    "P.UNIDADE_CODIGO ORDER BY DEPTO_CODIGO,UNIDADE_CODIGO";
        return $_strSQL;
    }

    /**
     * Devemos pré-processar o relatório antes de ser gerado o html
     *
     */
    public function processaRelatorio() {
        // Vamos acumular os valores
        $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
        ['PRODUTO_ESTOQUE'] += $this->getCampo("PRODUTO_ESTOQUE")->getValor();
        $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
        ['VALOR_ESTOQUE'] += $this->getCampo("VALOR_ESTOQUE")->getValor();
        $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
        ['VALOR_ESTOQUE_RP'] += $this->getCampo("VALOR_ESTOQUE_RP")->getValor();
    }

    /**
     * Exibe os totais do relatório
     * Valor de estoques a preço atual
     * Valor de estoques a custo reposição
     *
     * @return array
     */
    public function posRelatorio() {
        $_extra = new atributo("STYLE","height:20px;color:Navy;" .
                               "font-weight:bold;");
        foreach($this->_acumulado as $_un=>$_acumulado) {
            $_linha =Array(
                        Array(
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("UNIDADE_CODIGO","",2,null,null),
                            new float("PRODUTO_ESTOQUE","",12,null,null),
                            new dinheiro("VALOR_ESTOQUE","",12,null,null),
                            new dinheiro("VALOR_ESTOQUE_RP","",12,null,null)
                        ),
                        Array(
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra,
```

```
                                        $_extra
                                    )
                                );
                $_linha[0][2]->setValor($_un);
                $_linha[0][3]->setValor($_acumulado['PRODUTO_ESTOQUE']);
                $_linha[0][4]->setValor($_acumulado['VALOR_ESTOQUE']);
                $_linha[0][5]->setValor($_acumulado['VALOR_ESTOQUE_RP']);
                $_campos[] = $_linha;
            }
            $_campos[0][0][0]->setValor('Totais');
            return $_campos;
        }
}
?>
```

Agora basta criar o programa para a execução do relatório. Chamaremos o programa de *rel_estoquedepto.php5* e sua codificação é exibida na lista a seguir.

Lista 4: rel_estoquedepto.php5

```
<?php
/**
 * Relatório: Estoque por Departamento
 *
 */
$_classe = Array(
                "arquivo"=>"estoque/classes/classe_estoquedepto.inc",
                "nome"=>"estoquedepto");
$_FTitulo = "Relatório de Estoque por Departamento";
return include_once("../instanciaclasse.php5");
?>
```

Basta incluir o programa no menu do sistema e já poderemos emitir o relatório de estoque por departamento.

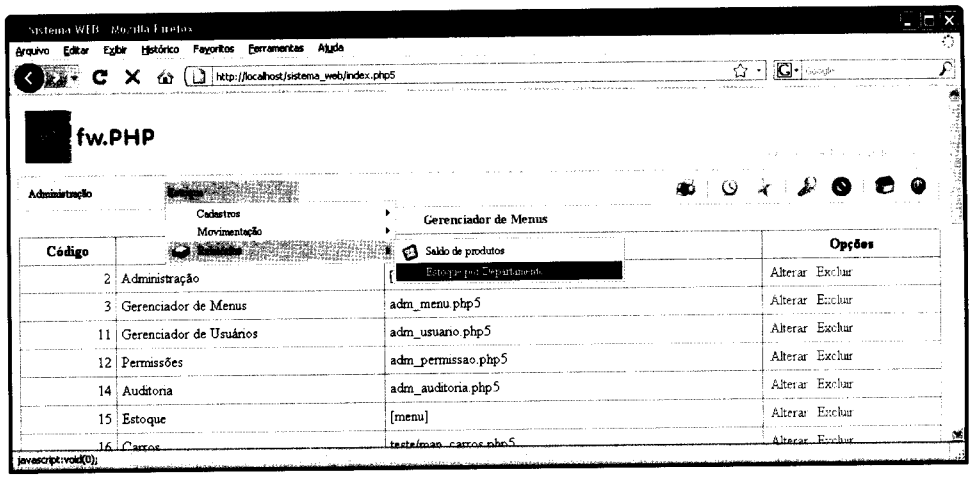


Figura 8.4

As Figuras 8.5 e 8.6 mostram o formulário de definição do filtro do relatório e um exemplo de um relatório de estoque por departamento respectivamente.

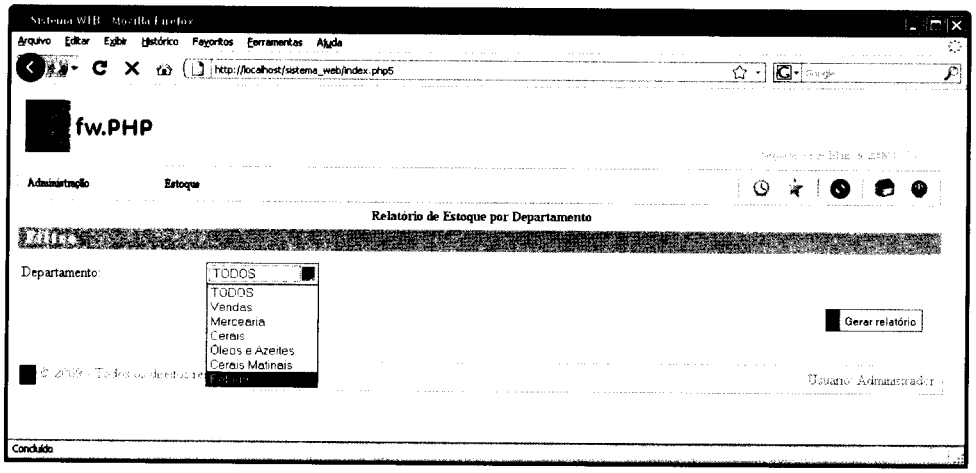


Figura 8.5

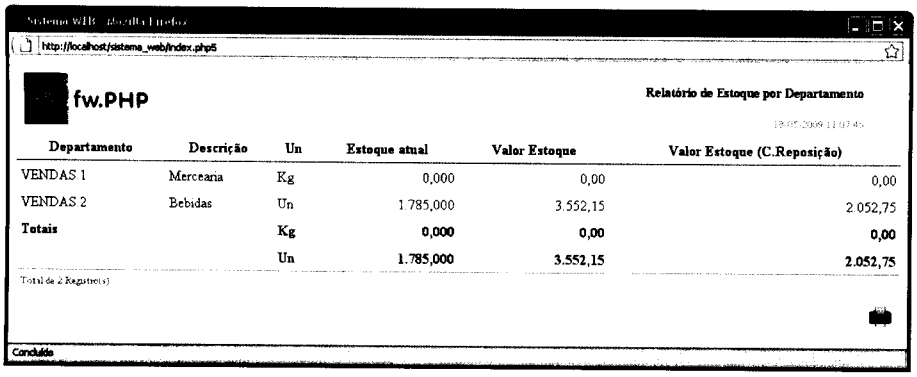


Figura 8.6

Estoque por grupo de produtos

Este é uma variação do relatório anterior. O objetivo é disponibilizar uma forma alternativa de visualização dos totais de estoque, pois enquanto o relatório anterior mostra o resumo por departamento, este exibe o resumo por grupo de produtos.

O usuário pode emitir o relatório de todos os grupos ou opcionalmente escolher um determinado grupo de produtos para a emissão do relatório.

A classe deve acumular por unidade de medida o total de estoque, a valorização a preço atual e a valorização a custo de reposição.

Assim teremos o seguinte filtro para esse relatório:

- Grupo de produtos

O relatório deverá conter os seguintes campos:

- Código do grupo
- Descrição do grupo
- Unidade de medida
- Estoque total
- Valor do estoque a preço atual
- Valor do estoque a custo de reposição (custo atual)

E como totais, por unidade de medida, teremos:

- Estoque total
- Valor do estoque a preço atual
- Valor do estoque a custo de reposição (custo atual)
- Todos os campos, tanto de filtro quanto do relatório (se diferentes), devem ser definidos no construtor da classe.

1. GRUPO_CODIGO

   Instância da classe **string**, define o grupo que será filtrado no relatório. O atributo 'comportamento form' desse campo deve ser definido como 'select' e seu conteúdo gerado antes da exibição do formulário de filtragem. A lista de opções deve ter a opção-padrão 'Todos' para que seja possível ignorar o filtro. Somente esse campo será exibido no formulário de filtragem.

2. GRUPO_DESC

   Instância da classe **string**, é utilizado para a descrição do grupo de produtos no relatório, não sendo exibido no formulário de filtragem.

3. UNIDADE_CODIGO

   Instância da classe **string**, é utilizado para separação dos estoques do departamento, evitando que unidades diferentes sejam somadas. Esse campo não será exibido no formulário de filtragem.

4. PRODUTO_ESTOQUE

   Instância da classe **float**, contém o estoque total da relação departamento e unidade de medida. Esse campo aparece apenas no relatório.

5. VALOR_ESTOQUE

Instância da classe **dinheiro**, ele não existe na tabela de produtos e deve ser calculado, sendo sua fórmula:

```
VALOR_ESTOQUE = PRODUTO_ESTOQUE * PRODUTO_PRECO
```

Esse campo é totalizado no final do relatório por unidade de medida.

6. VALOR_ESTOQUE_RP

Instância da classe **dinheiro**, não existe na tabela de produtos e deve ser calculado, sendo sua fórmula:

```
VALOR_ESTOQUE_RP = PRODUTO_ESTOQUE * PRODUTO_CUSTOATUAL
```

Esse campo é totalizado no final do relatório por unidade de medida.

O método construtor da classe *__construct()* deve definir os campos listados anteriormente, separando os campos que são filtros dos que são parte apenas do relatório.

```
public function __construct(BancoDados $_conn) {
     parent::__construct($_conn);
     $this->_nome_tabela = 'produto';
     $this->_class_path = 'estoque';
     $this->addCampo(new string("GRUPO_CODIGO","Grupo",
                        10,null,null,true,true,true,1,null,null,false,$_conn));
     $this->addCampo(new string("GRUPO_DESC","Descrição",40,null,null));
     $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
     $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",12,null,null));
     $this->addCampo(new dinheiro("VALOR_ESTOQUE","Valor Estoque",12,null,null));
     $this->addCampo(new dinheiro("VALOR_ESTOQUE_RP",
                        "Valor Estoque (C.Reposição)",12,null,null));
     // Acertar filtro
     $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
     $this->getCampo("GRUPO_DESC")->setFiltro(false);
     $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
     $this->getCampo("VALOR_ESTOQUE")->setFiltro(false);
     $this->getCampo("VALOR_ESTOQUE_RP")->setFiltro(false);
}
```

No método *processaAcao()* devemos preencher a lista de grupos disponíveis, inserindo no início a opção-padrão 'TODOS' para que seja possível ignorar o filtro. No final do método executamos o método *geraFormularioFiltroRelatorio()* responsável por montar o formulário de filtragem e chamar o método para exibição do relatório.

```
/**
 * Sempre executamos o módulo de relatório especial
 *
 */
public function processaAcao() {
     // Sempre mostra o filtro do relatório nada mais...
     $_grp = new grupo($this->_conn);
     $_grp->Buscar();
     $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
     $_lgrp[] = Array("valor"=>'TODOS',"label"=>'TODOS');
     while($_grp->proximo()!==false) {
          $_lgrp[] = Array("valor"=>$_grp->getCampo('GRUPO_CODIGO')->getValor(),
```

```
                    "label"=>$_grp->getCampo('GRUPO_DESC')->getValor());
    }
    $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
    $this->geraFormularioFiltroRelatorio();
}
```

O método *montaSELECT()* dessa classe é mais simples que o do relatório anterior, pois precisamos apenas aplicar o filtro de grupo caso ele seja definido pelo usuário.

```
/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                            $_orderby=null,$_limit=null,$_extras=null) {
    // 1o. Buscar a lista de departamentos
    $_strSQL= "SELECT " .
              "P.GRUPO_CODIGO,G.GRUPO_DESC,P.UNIDADE_CODIGO," .
              "SUM(PRODUTO_ESTOQUE) AS PRODUTO_ESTOQUE," .
              "SUM(PRODUTO_ESTOQUE*PRODUTO_PRECO) AS VALOR_ESTOQUE," .
              "SUM(PRODUTO_CUSTOATUAL*PRODUTO_ESTOQUE) AS VALOR_ESTOQUE_RP " .
              "FROM {$this->_nome_tabela} P " .
              "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO";
    // Filtro
    if($_POST['GRUPO_CODIGO']!='TODOS') {
        $_strSQL .= " WHERE P.GRUPO_CODIGO='{$_POST['GRUPO_CODIGO']}' ";
    }
    $_strSQL .= " GROUP BY P.GRUPO_CODIGO,G.GRUPO_DESC,P.UNIDADE_CODIGO " .
                "ORDER BY GRUPO_CODIGO,UNIDADE_CODIGO";
    return $_strSQL;
}
```

O método *processaRelatorio()* faz a acumulação do estoque e de suas valorizações tanto a preço atual quanto a custo de reposição (custo atual). Além disso o método limpa o conteúdo dos campos GRUPO_CODIGO e GRUPO_DESC quando a linha impressa for da segunda em diante para o mesmo grupo de produtos (deixando o relatório com uma visualização mais clara e limpa). É necessária a definição de um novo atributo da classe que conterá o último grupo processado no relatório.

```
/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
public function processaRelatorio() {
    // Vamos acumular os valores
    $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
    ['PRODUTO_ESTOQUE'] += $this->getCampo("PRODUTO_ESTOQUE")->getValor();
    $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
    ['VALOR_ESTOQUE'] += $this->getCampo("VALOR_ESTOQUE")->getValor();
    $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
    ['VALOR_ESTOQUE_RP'] += $this->getCampo("VALOR_ESTOQUE_RP")->getValor();
    if($this->_grupo_atual==$this->getCampo("GRUPO_CODIGO")->getValor()) {
        $this->getCampo("GRUPO_CODIGO")->setValor("");
        $this->getCampo("GRUPO_DESC")->setValor("");
    } else {
        $this->_grupo_atual = $this->getCampo("GRUPO_CODIGO")->getValor();
    }
}
```

O método *posRelatorio()* é idêntico ao mostrado no relatório anterior.

Agora temos a classe **estoquegrupo** finalizada. A lista apresentada em seguida exibe a listagem completa da classe, incluindo os atributos necessários ao seu correto processamento.

Lista 5: classe_estoquegrupo.inc

```
<?php
/**
 * Classe Estoque por Grupo de produtos
 */
class estoquegrupo extends base {
     protected $_acumulado = array();
     protected $_grupo_atual = null;

     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'produto';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("GRUPO_CODIGO","Grupo",
                         10,null,null,true,true,true,1,null,null,false,$_conn));
          $this->addCampo(new string("GRUPO_DESC","Descrição",40,null,null));
          $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
          $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",
                         12,null,null));
          $this->addCampo(new dinheiro("VALOR_ESTOQUE","Valor Estoque",
                         12,null,null));
          $this->addCampo(new dinheiro("VALOR_ESTOQUE_RP",
                         "Valor Estoque (C.Reposição)",12,null,null));
          // Acertar filtro
          $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
          $this->getCampo("GRUPO_DESC")->setFiltro(false);
          $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
          $this->getCampo("VALOR_ESTOQUE")->setFiltro(false);
          $this->getCampo("VALOR_ESTOQUE_RP")->setFiltro(false);
     }

     /**
      * Sempre executamos o módulo de relatório especial
      *
      */
     public function processaAcao() {
          // Sempre mostra o filtro do relatório nada mais...
          $_grp = new grupo($this->_conn);
          $_grp->Buscar();
          $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
          $_lgrp[] = Array("valor"=>'TODOS',"label"=>'TODOS');
          while($_grp->proximo()!==false) {
               $_lgrp[] = Array(
                              "valor"=>$_grp->getCampo('GRUPO_CODIGO')->getValor(),
                              "label"=>$_grp->getCampo('GRUPO_DESC')->getValor());
          }
          $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
          $this->geraFormularioFiltroRelatorio();
     }

     /**
      * Uma classe de relatórios deve sempre montar seu select
      *
      */
     public function montaSELECT($_campos,$_where=null,
                                 $_orderby=null,$_limit=null,$_extras=null) {
          // 1o. Buscar a lista de departamentos
```

```
        $_strSQL= "SELECT " .
                  "P.GRUPO_CODIGO,G.GRUPO_DESC,P.UNIDADE_CODIGO," .
             "SUM(PRODUTO_ESTOQUE) AS PRODUTO_ESTOQUE," .
             "SUM(PRODUTO_ESTOQUE*PRODUTO_PRECO) AS VALOR_ESTOQUE," .
             "SUM(PRODUTO_CUSTOATUAL*PRODUTO_ESTOQUE) AS VALOR_ESTOQUE_RP " .
             "FROM {$this->_nome_tabela} P " .
             "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO";
        // Filtro
        if($_POST['GRUPO_CODIGO']!='TODOS') {
             $_strSQL .= " WHERE P.GRUPO_CODIGO='{$_POST['GRUPO_CODIGO']}' ";
        }
        $_strSQL .= " GROUP BY P.GRUPO_CODIGO,G.GRUPO_DESC," .
                    "P.UNIDADE_CODIGO ORDER BY GRUPO_CODIGO,UNIDADE_CODIGO";
        return $_strSQL;
   }

   /**
    * Devemos pré-processar o relatório antes de ser gerado o html
    *
    */
   public function processaRelatorio() {
        // Vamos acumular os valores
        $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
        ['PRODUTO_ESTOQUE'] += $this->getCampo("PRODUTO_ESTOQUE")->getValor();
        $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
        ['VALOR_ESTOQUE'] += $this->getCampo("VALOR_ESTOQUE")->getValor();
        $this->_acumulado[$this->getCampo("UNIDADE_CODIGO")->getValor()]
        ['VALOR_ESTOQUE_RP'] += $this->getCampo("VALOR_ESTOQUE_RP")->getValor();
        if($this->_grupo_atual==$this->getCampo("GRUPO_CODIGO")->getValor()) {
             $this->getCampo("GRUPO_CODIGO")->setValor("");
             $this->getCampo("GRUPO_DESC")->setValor("");
        } else {
             $this->_grupo_atual = $this->getCampo("GRUPO_CODIGO")->getValor();
        }
   }

   /**
    * Exibe os totais do relatório
    * Valor de estoques a preço atual
    * Valor de estoques a custo reposição
    *
    * @return array
    */
   public function posRelatorio() {
        $_extra = new atributo("STYLE","height:20px;color:Navy;" .
                                       "font-weight:bold;");
        foreach($this->_acumulado as $_un=>$_acumulado) {
             $_linha = Array(
                            Array(
                                 new string("","",0,null,null),
                                 new string("","",0,null,null),
                                 new string("UNIDADE_CODIGO","",2,null,null),
                                 new float("PRODUTO_ESTOQUE","",12,null,null),
                                 new dinheiro("VALOR_ESTOQUE","",12,null,null),
                                 new dinheiro("VALOR_ESTOQUE_RP","",12,null,null)
                            ),
                            Array(
                                 $_extra,
                                 $_extra,
                                 $_extra,
                                 $_extra,
                                 $_extra,
                                 $_extra
                            )
                       );
```

```
                $_linha[0][2]->setValor($_un);
                $_linha[0][3]->setValor($_acumulado['PRODUTO_ESTOQUE']);
                $_linha[0][4]->setValor($_acumulado['VALOR_ESTOQUE']);
                $_linha[0][5]->setValor($_acumulado['VALOR_ESTOQUE_RP']);
                $_campos[] = $_linha;
            }
            $_campos[0][0][0]->setValor('Totais');
            return $_campos;
        }
}
?>
```

A lista seguinte mostra o programa *rel_estoquegrupo.php5* o qual deve ser incluído no menu do sistema de estoque.

Lista 6: rel_estoquegrupo.php5

```
<?php
/**
 * Relatório: Estoque por Departamento
 *
 */
$_classe = Array(
                "arquivo"=>"estoque/classes/classe_estoquegrupo.inc",
                "nome"=>"estoquegrupo");
$_FTitulo = "Relatório de Estoque por Grupo de produtos";
return include_once("../instanciaclasse.php5");
?>
```

A Figura 8.7 indica como deve ficar o menu de relatórios dentro do sistema de controle de estoque.

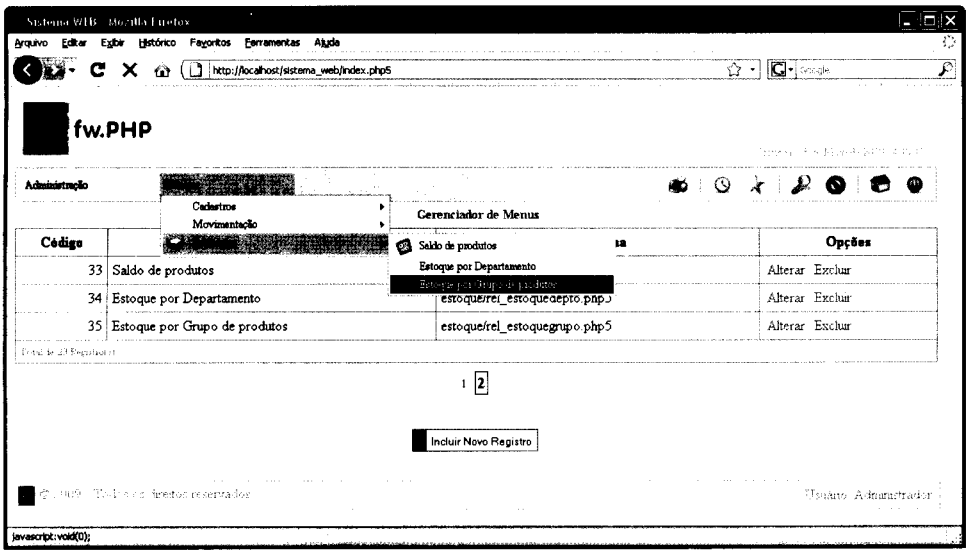


Figura 8.7

Com o programa incluído no menu do sistema podemos (após executar a saída do sistema e uma nova autenticação para que o menu seja reconstruído) executar o novo relatório. As Figuras 8.8 e 8.9 mostram o formulário de filtro do relatório e um exemplo de sua emissão.

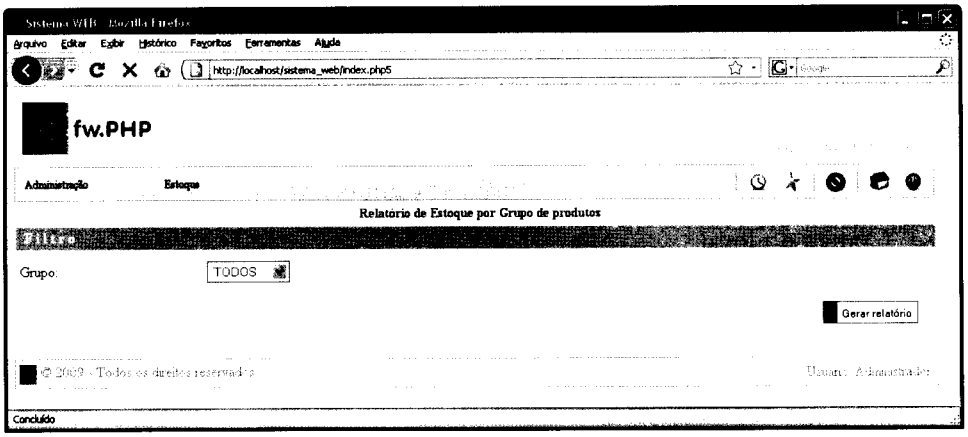


Figura 8.8

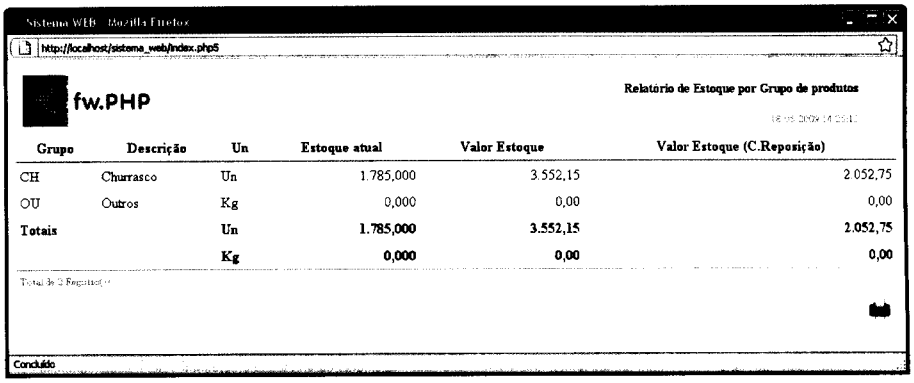


Figura 8.9

Estoque por local

Esse relatório apresenta uma outra forma de visualizar os estoques da empresa, separando-os por local de estoque. O objetivo do relatório é apresentar os locais de estoque e os produtos que possuem estoque nesses locais (ou que já possuíram).

O usuário pode selecionar um intervalo de locais e de produtos para a formação do relatório.

Ao final do relatório teremos um resumo por local de estoque e unidade de medida.

O relatório disponibilizará os filtros:

- Local inicial
- Local final
- Produto inicial
- Produto final

A emissão do relatório conterá os seguintes campos:

- Código do local
- Descrição do local
- Produto
- Unidade de medida
- Departamento
- Grupo de produtos
- Estoque atual
- Custo de reposição (custo atual)
- Valorização dos estoques

A totalização no final do relatório é por local e por unidade de medida, contendo os seguintes totalizadores:

- Total de estoques
- Total da valorização a custo de reposição

Os campos, quer sejam para o formulário de filtragem, quer sejam para o relatório, devem ser definidos no construtor da classe.

1. LOCAL_CODIGO

   Instância da classe **string**, define o código do local que será exibido no relatório e também o campo 'Local inicial' no formulário de filtragem. Esse campo deve ter seu atributo 'comportamento form' definido como 'select' no formulário e seu título alterado para 'Local' no momento da exibição do relatório.

2. LOCAL_CODIGO_F

   Instância da classe **string**, define o código do local final para o filtro do relatório. Esse campo deve ter seu atributo 'comportamento form' definido como 'select' no formulário e não será exibido no formulário. No momento de sua utilização para

a construção da cláusula WHERE no comando SQL, deve-se alterar o nome do campo para LOCAL_CODIGO.

3. PRODUTO_CODIGO

   Instância da classe **string**, define o código do produto que será exibido no relatório e também o campo 'Produto inicial' no formulário de filtragem. Esse campo deve ter seu atributo 'comportamento form' definido como 'ajax' no formulário e não será exibido no formulário (em vez do código exibimos a descrição do produto).

4. PRODUTO_CODIGO_F

   Instância da classe **string**, define o código do produto final no filtro de registros do relatório. Esse campo deve ser utilizado apenas no formulário de filtragem, não sendo exibido no relatório. O atributo 'comportamento form' desse campo deve ser definido como 'ajax'. No momento de sua utilização para a construção da cláusula WHERE no comando SQL, deve-se alterar o nome do campo para PRODUTO_CODIGO.

5. LOCAL_DESC

   Instância da classe **string**, ele é utilizado para a descrição do Local de estoque, não sendo exibido no formulário de filtragem.

6. PRODUTO_DESC

   Instância da classe **string**, esse campo é utilizado para a descrição do produto no relatório, não sendo exibido no formulário de filtragem.

7. UNIDADE_CODIGO

   Instância da classe **string**, é utilizado para a descrição do produto no relatório, não sendo exibido no formulário de filtragem.

8. DEPTO_DESC

   Instância da classe **string**, esse campo é utilizado para a descrição do departamento no relatório, não sendo exibido no formulário de filtragem.

9. GRUPO_DESC

   Instância da classe **string**, é utilizado para a descrição do grupo de produtos no relatório, não sendo exibido no formulário de filtragem.

10. PRODUTO_ESTOQUE

    Instância da classe **float**, define o Estoque atual do produto no local de estoque. Ele aparece apenas no relatório e é totalizado no final do relatório.

11. PRODUTO_CUSTOATUAL

Instância da classe **dinheiro**, contém o custo de reposição (custo atual) do produto e será exibido apenas no relatório.

12. VALOR_ESTOQUE_RP

Instância da classe **dinheiro**, ele não existe na tabela de locais de estoques e deve ser calculado, sendo sua fórmula:

```
VALOR_ESTOQUE_RP = PRODUTO_CUSTOATUAL * PRODUTO_ESTOQUE
```

Esse campo é totalizado no final do relatório.

No método construtor da classe *__construct()* teremos todos os campos, descritos anteriormente, definidos e separados, quando for o caso, os que são somente filtro dos que são parte integrante do relatório. Ajustamos, ainda, nesse método o atributo 'comportamento form' dos campos relacionados ao produto, PRODUTO_CODIGO e PRODUTO_CODIGO_F, para 'ajax' informando também a classe que deve ser utilizada e o arquivo que contém a classe.

```
public function __construct(BancoDados $_conn) {
    parent::__construct($_conn);
    $this->_nome_tabela = 'estoquelocal';
    $this->_class_path = 'estoque';
    $this->addCampo(new string("LOCAL_CODIGO","Local Inicial",
                    10,null,null,true,false,true,1,null,null,false,$_conn));
    $this->addCampo(new string("LOCAL_CODIGO_F","Local Final",
                    10,null,null,true,false,true,1,null,null,false,$_conn));
    $this->addCampo(new string("LOCAL_DESC","Descrição",30,null,null));
    $this->addCampo(new string("PRODUTO_CODIGO","Produto Inicial",
                    20,null,null,true,false,true,1,null,null,false,$_conn));
    $this->addCampo(new string("PRODUTO_CODIGO_F","Produto Final",
                    20,null,null,true,false,true,1,null,null,false,$_conn));
    $this->addCampo(new string("PRODUTO_DESC","Produto",60,null,null));
    $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
    $this->addCampo(new string("DEPTO_DESC","Departamento",40,null,null));
    $this->addCampo(new string("GRUPO_DESC","Grupo",40,null,null));
    $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",12,null,null));
    $this->addCampo(new dinheiro("PRODUTO_CUSTOATUAL","Custo reposição",
                    12,null,null));
    $this->addCampo(new dinheiro("VALOR_ESTOQUE_RP","Valor Estoque",
                    12,null,null));
    // Acertar filtro
    $this->getCampo("LOCAL_DESC")->setFiltro(false);
    $this->getCampo("PRODUTO_DESC")->setFiltro(false);
    $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
    $this->getCampo("DEPTO_DESC")->setFiltro(false);
    $this->getCampo("GRUPO_DESC")->setFiltro(false);
    $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
    $this->getCampo("PRODUTO_CUSTOATUAL")->setFiltro(false);
    $this->getCampo("VALOR_ESTOQUE_RP")->setFiltro(false);
    // Acertar Campos do relatório
    $this->getCampo("LOCAL_CODIGO_F")->setRelatorio(false);
    $this->getCampo("PRODUTO_CODIGO")->setRelatorio(false);
    $this->getCampo("PRODUTO_CODIGO_F")->setRelatorio(false);
    $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
    $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
```

```
        $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                                        "estoque/classes/classe_produto.inc");
        $this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                                        "estoque/classes/classe_produto.inc");
}
```

O método *processaAcao()* preenche a lista de opções dos campos para definição dos locais de estoque inicial e final. Além disso teremos no final do método a chamada ao método *geraFormularioFiltroRelatorio()* da classe **base**.

```
/**
 * Sempre executamos o módulo de relatório especial
 *
 */
public function processaAcao() {
        // Sempre mostra o filtro do relatório nada mais...
        $_lc = new localestoque($this->_conn);
        $_lc->Buscar();
        $this->getCampo("LOCAL_CODIGO")->setComportamento_form('select');
        $this->getCampo("LOCAL_CODIGO_F")->setComportamento_form('select');
        $_llc[] = Array("valor"=>'TODOS',"label"=>'TODOS');
        while($_lc->proximo()!==false) {
                $_llc[] = Array("valor"=>$_lc->getCampo('LOCAL_CODIGO')->getValor(),
                                "label"=>$_lc->getCampo('LOCAL_DESC')->getValor());
        }
        $this->getCampo("LOCAL_CODIGO")->setValor_fixo($_llc);
        $this->getCampo("LOCAL_CODIGO_F")->setValor_fixo($_llc);
        $this->geraFormularioFiltroRelatorio();
}
```

O método *montaSELECT()* dessa classe realiza a busca dos dados na tabela de estoques por local, fazendo a junção com as tabelas de locais, produtos, departamentos e grupos. Teremos ainda o cálculo dos campos de valores e a aplicação dos filtros selecionados, quando for o caso. Na aplicação do filtro deve-se trocar os campos LOCAL_CODIGO_F por LOCAL_CODIGO e PRODUTO_CODIGO_F por PRODUTO_CODIGO.

```
/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                        $_orderby=null,$_limit=null,$_extras=null) {
        $this->getCampo("LOCAL_CODIGO")->setTitulo("Local");
        $_strSQL= "SELECT " .
                  "L.LOCAL_CODIGO,LL.LOCAL_DESC,P.PRODUTO_DESC, D.DEPTO_DESC," .
                  "G.GRUPO_DESC,L.PRODUTO_ESTOQUE,P.PRODUTO_CUSTOATUAL," .
                  "P.UNIDADE_CODIGO,L.PRODUTO_CUSTOMEDIO, " .
                  "L.PRODUTO_CUSTOMEDIO*L.PRODUTO_ESTOQUE AS VALOR_ESTOQUE_RP " .
                  "FROM {$this->_nome_tabela} L " .
                  "LEFT JOIN LOCALESTOQUE LL ON L.LOCAL_CODIGO=LL.LOCAL_CODIGO " .
                  "LEFT JOIN PRODUTO P ON L.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
                  "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
                  "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO ";
        // Filtro
        $_where = Array();
```

```
        $_operador = Array('PRODUTO_CODIGO'=>'>=','PRODUTO_CODIGO_F'=>'<=',
                           'LOCAL_CODIGO'=>'>=','LOCAL_CODIGO_F'=>'<=',);
        $_alias    = Array('PRODUTO_CODIGO_F'=>'PRODUTO_CODIGO',
                           'LOCAL_CODIGO_F'=>'LOCAL_CODIGO');
        foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
            if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
                $_campo->setValor($_POST[$_nome]);
                $_where[] = "L." . (array_key_exists($_nome,$_alias)
                                        ? $_alias[$_nome]
                                        : $_nome) .
                            (array_key_exists($_nome,$_operador)
                                        ? $_operador[$_nome]
                                        : '=') .
                            $_campo->toBD();
            }
        }
        if(sizeof($_where)>0) {
            $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
        }

        $_strSQL .=   "ORDER BY L.LOCAL_CODIGO,L.PRODUTO_CODIGO";
        return $_strSQL;
    }
```

É no método *processaRelatorio()* que os totais são acumulados. Teremos nesse relatório dois níveis de totais. O primeiro nível será para os locais de estoque e o segundo para as unidades de medida. Trataremos, ainda, da exibição dos locais de estoque, evitando o mesmo local repetidamente mostrado no relatório. Assim, quando não for a primeira exibição do local, limpamos os campos LOCAL_CODIGO e LOCAL_DESC do registro atual para impressão, deixando o relatório com um visual mais limpo e fácil de ler.

```
    /**
     * Devemos pré-processar o relatório antes de ser gerado o html
     *
     */
    public function processaRelatorio() {
        // Vamos acumular os valores
        $_un = $this->getCampo("UNIDADE_CODIGO")->getValor();
        $_lc = $this->getCampo("LOCAL_CODIGO")->getValor();
        $this->_acumulado[$_lc][$_un][0] = $this->getCampo("LOCAL_DESC")->getValor();
        $this->_acumulado[$_lc][$_un]['PRODUTO_ESTOQUE'] +=
                                    $this->getCampo("PRODUTO_ESTOQUE")->getValor();
        $this->_acumulado[$_lc][$_un]['VALOR_ESTOQUE_RP'] +=
                                    $this->getCampo("VALOR_ESTOQUE_RP")->getValor();
        if($this->_local_atual==$_lc) {
            $this->getCampo("LOCAL_CODIGO")->setValor("");
            $this->getCampo("LOCAL_DESC")->setValor("");
        } else {
            $this->_local_atual = $_lc;
        }
    }
```

Por fim teremos o método *posRelatorio()* que gera as linhas de totais do relatório. Como temos dois níveis de totais, um para o local e outro para a unidade de medida, devemos gerar várias linhas de totalização, tomando o cuidado de não repetir os títulos

dos locais de estoque (vale o mesmo critério estabelecido para o relatório, ou seja, devemos exibir o código e o nome do local somente na primeira linha de totais).

```
/**
 * Exibe os totais do relatório
 * Valor de estoques a preço atual
 * Valor de estoques a custo médio
 *
 * @return array
 */
public function posRelatorio() {
     $_extra = new atributo("STYLE","height:20px;color:Navy;font-weight:bold;");
     $_campos[0] = Array(
                         Array(
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("","",0,null,null),
                              new string("","",0,null,null)
                         ),
                         Array(
                              $_extra,
                              $_extra,
                              $_extra,
                              $_extra,
                              $_extra,
                              $_extra
                         )
                    );
     foreach($this->_acumulado as $_lc=>$_ac) {
          $_first = true;
          foreach($_ac as $_un=>$_acumulado) {
               $_linha = Array(
                              Array(
                                   new string("LOCAL_CODIGO","",0,null,null),
                                   new string("","",0,null,null),
                                   new string("","",0,null,null),
                                   new string("UNIDADE_CODIGO","",2,null,null),
                                   new string("","",2,null,null),
                                   new string("","",2,null,null),
                                   new float("PRODUTO_ESTOQUE","",12,null,null),
                                   new string("","",2,null,null),
                                   new dinheiro("VALOR_ESTOQUE_RP","",12,null,null)
                              ),
                              Array(
                                   $_extra,
                                   $_extra,
                                   $_extra,
                                   $_extra,
                                   $_extra,
                                   $_extra,
                                   $_extra,
                                   $_extra,
                                   $_extra
                              )
                         );
               $_linha[0][3]->setValor($_un);
               $_linha[0][6]->setValor($_acumulado['PRODUTO_ESTOQUE']);
```

```
                $_linha[0][8]->setValor($_acumulado['VALOR_ESTOQUE_RP']);
                if($_first===true) {
                    $_linha[0][0]->setValor($_lc);
                    $_linha[0][1]->setValor($_acumulado[0]);
                }
                $_first = false;
                $_campos[] = $_linha;
            }
        }
        $_campos[0][0][0]->setValor('Totais');
        return $_campos;
}
```

Observe que geramos a primeira linha nos totais, separada das linhas geradas pelos acumuladores do relatório, a qual terá apenas a palavra 'Totais'.

A classe **estoquelocalatuallocal** está terminada, pois definimos todos os métodos necessários para o correto processamento do relatório desejado. A lista seguinte exibe a classe em seu formato completo, lembrando que essa classe deve ser salva no arquivo *classe_estoqueatualloca.inc* dentro do diretório *estoque/classes/*.

Lista 7: classe_estoqueatuallocal.inc

```
<?php
/**
 * Classe Estoque Atual por local de estoque
 */
class estoqueatuallocal extends base {
    protected $_acumulado = array();
    protected $_local_atual = null;

    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'estoquelocal';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("LOCAL_CODIGO","Local Inicial",
                    10,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("LOCAL_CODIGO_F","Local Final",
                    10,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("LOCAL_DESC","Descrição",30,null,null));
        $this->addCampo(new string("PRODUTO_CODIGO","Produto Inicial",
                    20,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("PRODUTO_CODIGO_F","Produto Final",
                    20,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("PRODUTO_DESC","Produto",60,null,null));
        $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
        $this->addCampo(new string("DEPTO_DESC","Departamento",40,null,null));
        $this->addCampo(new string("GRUPO_DESC","Grupo",40,null,null));
        $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",
                    12,null,null));
        $this->addCampo(new dinheiro("PRODUTO_CUSTOATUAL","Custo reposição",
                    12,null,null));
        $this->addCampo(new dinheiro("VALOR_ESTOQUE_RP","Valor Estoque",
                    12,null,null));
        // Acertar filtro
        $this->getCampo("LOCAL_DESC")->setFiltro(false);
        $this->getCampo("PRODUTO_DESC")->setFiltro(false);
        $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
        $this->getCampo("DEPTO_DESC")->setFiltro(false);
        $this->getCampo("GRUPO_DESC")->setFiltro(false);
        $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
```

```
        $this->getCampo("PRODUTO_CUSTOATUAL")->setFiltro(false);
        $this->getCampo("VALOR_ESTOQUE_RP")->setFiltro(false);
        // Acertar Campos do relatório
        $this->getCampo("LOCAL_CODIGO_F")->setRelatorio(false);
        $this->getCampo("PRODUTO_CODIGO")->setRelatorio(false);
        $this->getCampo("PRODUTO_CODIGO_F")->setRelatorio(false);
        $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                                    "estoque/classes/classe_produto.inc");
        $this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                                    "estoque/classes/classe_produto.inc");
    }

    /**
     * Sempre executamos o módulo de relatório especial
     *
     */
    public function processaAcao() {
        // Sempre mostra o filtro do relatório nada mais...
        $_lc = new localestoque($this->_conn);
        $_lc->Buscar();
        $this->getCampo("LOCAL_CODIGO")->setComportamento_form('select');
        $this->getCampo("LOCAL_CODIGO_F")->setComportamento_form('select');
        $_llc[] = Array("valor"=>'TODOS',"label"=>'TODOS');
        while($_lc->proximo()!==false) {
            $_llc[] = Array("valor"=>$_lc->getCampo('LOCAL_CODIGO')->getValor(),
                            "label"=>$_lc->getCampo('LOCAL_DESC')->getValor());
        }
        $this->getCampo("LOCAL_CODIGO")->setValor_fixo($_llc);
        $this->getCampo("LOCAL_CODIGO_F")->setValor_fixo($_llc);
        $this->geraFormularioFiltroRelatorio();
    }

    /**
     * Uma classe de relatórios deve sempre montar seu select
     *
     */
    public function montaSELECT($_campos,$_where=null,
                                $_orderby=null,$_limit=null,$_extras=null) {
        $this->getCampo("LOCAL_CODIGO")->setTitulo("Local");
        $_strSQL= "SELECT " .
            "L.LOCAL_CODIGO,LL.LOCAL_DESC,P.PRODUTO_DESC, D.DEPTO_DESC," .
            "G.GRUPO_DESC,L.PRODUTO_ESTOQUE,L.PRODUTO_CUSTOATUAL," .
            "P.UNIDADE_CODIGO, " .
            "L.PRODUTO_CUSTOATUAL*L.PRODUTO_ESTOQUE AS VALOR_ESTOQUE_RP " .
            "FROM {$this->_nome_tabela} L " .
            "LEFT JOIN LOCALESTOQUE LL ON L.LOCAL_CODIGO=LL.LOCAL_CODIGO " .
            "LEFT JOIN PRODUTO P ON L.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
            "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
            "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO ";
        // Filtro
        $_where = Array();
        $_operador = Array('PRODUTO_CODIGO'=>'>=','PRODUTO_CODIGO_F'=>'<=',
                           'LOCAL_CODIGO'=>'>=','LOCAL_CODIGO_F'=>'<=',);
        $_alias    = Array('PRODUTO_CODIGO_F'=>'PRODUTO_CODIGO',
                           'LOCAL_CODIGO_F'=>'LOCAL_CODIGO');
        foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
            if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
                $_campo->setValor($_POST[$_nome]);
                $_where[] = "L." . (array_key_exists($_nome,$_alias)
                                    ? $_alias[$_nome]
                                    : $_nome) .
```


```
                                (array_key_exists($_nome,$_operador)
                                        ? $_operador[$_nome]
                                        : '=') .
                                $_campo->toBD();
                }
        }
        if(sizeof($_where)>0) {
                $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
        }

        $_strSQL .=    "ORDER BY L.LOCAL_CODIGO,L.PRODUTO_CODIGO";
        return $_strSQL;
}

/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
public function processaRelatorio() {
        // Vamos acumular os valores
        $_un = $this->getCampo("UNIDADE_CODIGO")->getValor();
        $_lc = $this->getCampo("LOCAL_CODIGO")->getValor();
        $this->_acumulado[$_lc][$_un][0] =
                                        $this->getCampo("LOCAL_DESC")->getValor();
        $this->_acumulado[$_lc][$_un]['PRODUTO_ESTOQUE'] +=
                                $this->getCampo("PRODUTO_ESTOQUE")->getValor();
        $this->_acumulado[$_lc][$_un]['VALOR_ESTOQUE_RP'] +=
                                $this->getCampo("VALOR_ESTOQUE_RP")->getValor();
        if($this->_local_atual==$_lc) {
                $this->getCampo("LOCAL_CODIGO")->setValor("");
                $this->getCampo("LOCAL_DESC")->setValor("");
        } else {
                $this->_local_atual = $_lc;
        }
}

/**
 * Exibe os totais do relatório
 * Valor de estoques a preço atual
 * Valor de estoques a custo médio
 *
 * @return array
 */
public function posRelatorio() {
        $_extra = new atributo("STYLE","height:20px;color:Navy;" .
                                        "font-weight:bold;");
        $_campos[0] = Array(
                        Array(
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null)
                        ),
                        Array(
                                $_extra,
                                $_extra,
                                $_extra,
                                $_extra,
                                $_extra,
                                $_extra
```

```
                    )
                );
        foreach($this->_acumulado as $_lc=>$_ac) {
            $_first = true;
            foreach($_ac as $_un=>$_acumulado) {
                $_linha =
                    Array(
                        Array(
                            new string("LOCAL_CODIGO","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("UNIDADE_CODIGO","",2,null,null),
                            new string("","",2,null,null),
                            new string("","",2,null,null),
                            new float("PRODUTO_ESTOQUE","",12,null,null),
                            new string("","",2,null,null),
                            new dinheiro("VALOR_ESTOQUE_RP","",12,null,null)
                        ),
                        Array(
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra,
                            $_extra
                        )
                    );
                $_linha[0][3]->setValor($_un);
                $_linha[0][6]->setValor($_acumulado['PRODUTO_ESTOQUE']);
                $_linha[0][8]->setValor($_acumulado['VALOR_ESTOQUE_RP']);
                if($_first===true) {
                    $_linha[0][0]->setValor($_lc);
                    $_linha[0][1]->setValor($_acumulado[0]);
                }
                $_first = false;
                $_campos[] = $_linha;
            }
        }
        $_campos[0][0][0]->setValor('Totais');
        return $_campos;
    }
}
?>
```

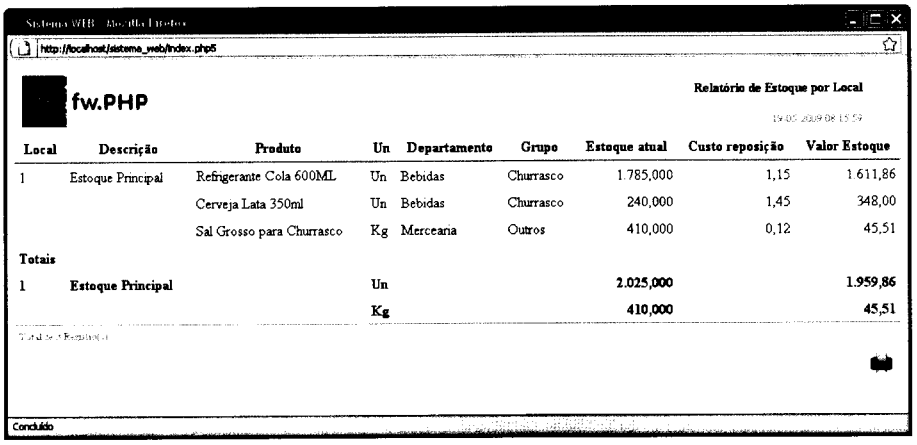
A lista seguinte mostra como deve ser o programa *rel_estoqueatuallocal.php5*.

Lista 8: rel_estoqueatualloca.php5

```
<?php
/**
 * Relatório: Estoque por Departamento
 *
 */
$_classe = Array(  "arquivo"=>"estoque/classes/classe_estoqueatuallocal.inc",
                   "nome"=>"estoqueatuallocal");
$_FTitulo = "Relatório de Estoque por Local";
return include_once("../instanciaclasse.php5");
?>
```

O programa deve ser incluído no menu do sistema antes de ser utilizado.

A Figura 8.10 apresenta um exemplo do relatório que é gerado pela classe que acabamos de construir.



fw.PHP

Relatório de Estoque por Local

Local	Descrição	Produto	Un	Departamento	Grupo	Estoque atual	Custo reposição	Valor Estoque
1	Estoque Principal	Refrigerante Cola 600ML	Un	Bebidas	Churrasco	1.785,000	1,15	1.611,86
		Cerveja Lata 350ml	Un	Bebidas	Churrasco	240,000	1,45	348,00
		Sal Grosso para Churrasco	Kg	Mercearia	Outros	410,000	0,12	45,51
Totais								
1	**Estoque Principal**		**Un**			**2.025,000**		**1.959,86**
			Kg			**410,000**		**45,51**

Concluído

Figura 8.10

Estoque por produto e local

O relatório anterior mostra uma visão dos estoques com foco nos locais de estoque e dos produtos pertencentes ao local. Ele propõe uma visão dos estoques por local, só que com foco no produto, ou seja, dado um produto, em quais locais esse produto possui estoque (ou possuiu, uma vez que estoques zerados também serão exibidos). As estruturas das classes são praticamente idênticas, com algumas poucas alterações, principalmente na ordem dos campos do relatório e ordenação da busca.

Assim como na classe **estoqueatuallocal**, teremos a opção de selecionar produtos, mas não locais, uma vez que desejamos ver todos os locais nos quais o produto aparece. Logo, teremos filtro apenas por produto.

Ao final do relatório não teremos nenhum resumo, pois o relatório de saldo de produtos já disponibiliza os totais por produto.

A emissão do relatório conterá os seguintes campos:

- Código do produto
- Descrição do produto
- Unidade de medida
- Local de estoque
- Departamento

- Grupo de produtos
- Estoque atual
- Custo de reposição (custo atual)
- Valorização dos estoques

Os métodos são muito parecidos, não sendo necessária uma discussão detalhada sobre cada um deles. Devemos lembrar apenas de retirar os campos de filtros por local do relatório e de que não teremos acumuladores ou totais nesse relatório.

O método *montaSELECT()* deve ser alterado para que a ordenação seja por produto e não por local, além de tratar somente o produto no filtro.

A listagem da classe **estoqueatualprodutolocal** é mostrada a seguir. Lembre-se de que a classe deve ser salva no arquivo *classe_estoqueatualprodutolocal.inc* dentro do diretório *estoque/classes/*.

Lista 9: classe_estoqueatualprodutolocal.inc

```
<?php
/**
 * Classe Estoque Atual por produto e local de estoque
 */
class estoqueatualprodutolocal extends base {
     protected $_produto_atual = null;

     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'estoquelocal';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("PRODUTO_CODIGO","Produto Inicial",
                         20,null,null,true,false,true,1,null,null,false,$_conn));
          $this->addCampo(new string("PRODUTO_CODIGO_F","Produto Final",
                         20,null,null,true,false,true,1,null,null,false,$_conn));
          $this->addCampo(new string("PRODUTO_DESC","Produto",60,null,null));
          $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
          $this->addCampo(new string("LOCAL_DESC","Local",30,null,null));
          $this->addCampo(new string("DEPTO_DESC","Departamento",40,null,null));
          $this->addCampo(new string("GRUPO_DESC","Grupo",40,null,null));
          $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",
                         12,null,null));
          $this->addCampo(new dinheiro("PRODUTO_CUSTOATUAL","Custo reposição",
                         12,null,null));
          $this->addCampo(new dinheiro("VALOR_ESTOQUE_RP","Valor Estoque",
                         12,null,null));
          // Acertar filtro
          $this->getCampo("LOCAL_DESC")->setFiltro(false);
          $this->getCampo("PRODUTO_DESC")->setFiltro(false);
          $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
          $this->getCampo("DEPTO_DESC")->setFiltro(false);
          $this->getCampo("GRUPO_DESC")->setFiltro(false);
          $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
          $this->getCampo("PRODUTO_CUSTOATUAL")->setFiltro(false);
          $this->getCampo("VALOR_ESTOQUE_RP")->setFiltro(false);
          // Acertar Campos do relatório
          $this->getCampo("PRODUTO_CODIGO_F")->setRelatorio(false);
          $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
          $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
```

```
        $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                                        "estoque/classes/classe_produto.inc");
        $this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                                        "estoque/classes/classe_produto.inc");
    }

    /**
     * Sempre executamos o módulo de relatório especial
     *
     */
    public function processaAcao() {
        $this->geraFormularioFiltroRelatorio();
    }

    /**
     * Uma classe de relatórios deve sempre montar seu select
     *
     */
    public function
montaSELECT($_campos,$_where=null,$_orderby=null,$_limit=null,$_extras=null) {
        $this->getCampo("PRODUTO_CODIGO")->setTitulo("Produto");
        $_strSQL= "SELECT " .
            "L.PRODUTO_CODIGO,LL.LOCAL_DESC,P.PRODUTO_DESC, D.DEPTO_DESC," .
            "G.GRUPO_DESC,L.PRODUTO_ESTOQUE,L.PRODUTO_CUSTOATUAL," .
            "P.UNIDADE_CODIGO," .
            "L.PRODUTO_CUSTOATUAL*L.PRODUTO_ESTOQUE AS VALOR_ESTOQUE_RP " .
            "FROM {$this->_nome_tabela} L " .
            "LEFT JOIN LOCALESTOQUE LL ON L.LOCAL_CODIGO=LL.LOCAL_CODIGO " .
            "LEFT JOIN PRODUTO P ON L.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
            "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
            "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO ";
        // Filtro
        $_where = Array();
        $_operador = Array('PRODUTO_CODIGO'=>'>=','PRODUTO_CODIGO_F'=>'<=');
        $_alias    = Array('PRODUTO_CODIGO_F'=>'PRODUTO_CODIGO');
        foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
            if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
                $_campo->setValor($_POST[$_nome]);
                $_where[] = "L." . (array_key_exists($_nome,$_alias)
                                    ? $_alias[$_nome]
                                    : $_nome) .
                            (array_key_exists($_nome,$_operador)
                                    ? $_operador[$_nome]
                                    : '=') .
                            $_campo->toBD();
            }
        }
        if(sizeof($_where)>0) {
            $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
        }

        $_strSQL .=    "ORDER BY L.PRODUTO_CODIGO,L.LOCAL_CODIGO";
        return $_strSQL;
    }

    /**
     * Devemos pré-processar o relatório antes de ser gerado o html
     *
     */
    public function processaRelatorio() {
        if($this->_produto_atual==$this->getCampo("PRODUTO_CODIGO")->getValor()){
            $this->getCampo("PRODUTO_CODIGO")->setValor("");
            $this->getCampo("PRODUTO_DESC")->setValor("");
```

```
            } else {
                $this->_produto_atual =
                                $this->getCampo("PRODUTO_CODIGO")->getValor();
            }
        }
}
?>
```

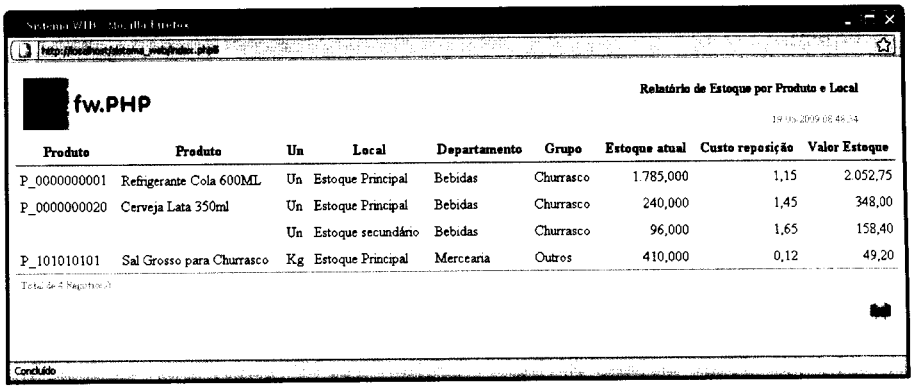
O próximo passo é criar o programa de emissão do relatório, cujo nome será *rel_estoqueatualprodutolocal.php5*.

Lista 10: rel_estoqueatualprodutolocal.php5

```
<?php
/**
 * Relatório: Estoque por Produto e Local
 *
 */
$_classe = Array(
                "arquivo"=>"estoque/classes/classe_estoqueatualprodutolocal.inc",
                "nome"=>"estoqueatualprodutolocal");
$_FTitulo = "Relatório de Estoque por Produto e Local";
return include_once("../instanciaclasse.php5");
?>
```

Finalmente para utilizar o relatório, basta incluí-lo no menu de relatórios no sistema de estoque e acertar a autorização de acesso, se for o caso.

A Figura 8.11 mostra um exemplo do relatório que acabamos de construir.



fw.PHP

Relatório de Estoque por Produto e Local

Produto	Produto	Un	Local	Departamento	Grupo	Estoque atual	Custo reposição	Valor Estoque
P_0000000001	Refrigerante Cola 600ML	Un	Estoque Principal	Bebidas	Churrasco	1.785,000	1,15	2.052,75
P_0000000020	Cerveja Lata 350ml	Un	Estoque Principal	Bebidas	Churrasco	240,000	1,45	348,00
		Un	Estoque secundário	Bebidas	Churrasco	96,000	1,65	158,40
P_101010101	Sal Grosso para Churrasco	Kg	Estoque Principal	Mercearia	Outros	410,000	0,12	49,20

Concluído

Figura 8.11

Produtos com estoque acima do máximo

Um dos objetivos mais importantes da utilização de um sistema de controle de estoque é o gerenciamento eficaz dos produtos e seus níveis de estoques, ajudando o tomador de decisões a manter os níveis dos estoques em patamares adequados, ou seja, nem estoque demais nem de menos.

O relatório atual mostra ao administrador os possíveis desperdícios no estoque, exibindo os produtos que estão com estoque acima do máximo estabelecido. Esses produtos devem receber um tratamento diferenciado, evitando-se que mais estoque seja agregado (compras, transferências etc.) ou preparando ações para que a saída do produto seja incrementada para que seu nível chegue a patamares aceitáveis. A preocupação deve ser ainda maior com produtos com prazo de validade curto, pois uma falta de ação nestes casos pode implicar na perda total dos lotes vencidos.

Neste relatório teremos os seguintes filtros:

- Departamento
- Grupo de produtos

Os seguintes campos são mostrados no relatório:

- Código do produto
- Descrição do produto
- Unidade de medida
- Departamento
- Grupo do produto
- Estoque máximo
- Estoque atual
- Diferença (7-6)
- Valorização da diferença a custo de reposição (custo atual)

No final do relatório exibiremos o total da valorização das diferenças, mostrando ao usuário o montante financeiro do descontrole no estoque.

Os campos, quer sejam para o formulário de filtragem ou para o relatório, devem ser definidos no construtor da classe.

1. PRODUTO_CODIGO

   Instância da classe **string**, define o código do produto que será exibido no relatório. Esse campo será mostrado apenas no relatório.

2. PRODUTO_DESC

   Instância da classe **string**, é utilizado para a descrição do produto no relatório, não sendo exibido no formulário de filtragem.

3. UNIDADE_CODIGO

   Instância da classe **string**, é utilizado para a descrição do produto no relatório, não sendo exibido no formulário de filtragem.

4. DEPTO_DESC

   Instância da classe **string**, esse campo é utilizado para a descrição do departamento no relatório, não sendo exibido no formulário de filtragem.

5. GRUPO_DESC

   Instância da classe **string**, é utilizado para a descrição do grupo de produtos no relatório, não sendo exibido no formulário de filtragem.

6. PRODUTO_ESTOQUE

   Instância da classe **float**, define o estoque atual do produto. Ele aparece apenas no relatório.

7. PRODUTO_EST_MAXIMO

   Instância da classe **float**, contém o estoque máximo do produto, sendo exibido apenas no relatório.

8. DIFERENCA

   Instância da classe **float**, esse campo contém a diferença existente entre o estoque atual (PRODUTO_ESTOQUE) e o estoque máximo cadastrado (PRODUTO_EST_MAXIMO). Esse campo não existe na tabela e será criado como uma expressão no comando SQL. Sua fórmula é:

```
DIFERENCA = PRODUTO_ESTOQUE - PRODUTO_EST_MAXIMO
```

9. VALOR_DIFERENCA

   Instância da classe **dinheiro**, contém a valorização da diferença a custo de reposição (custo atual) entre o estoque atual e o estoque máximo. Como esse campo não existe na tabela, utilizaremos uma expressão para criá-lo no comando SQL de seleção dos produtos. Sua fórmula será:

```
VALOR_DIFERENCA = DIFERENCA * PRODUTO_CUSTOATUAL
```

   O método construtor da classe deve definir os campos listados anteriormente, respeitando os tipos definidos. O método deve ainda separar os campos que são filtros dos que são parte do relatório.

   O método *processaAcao()* preenche as listas de departamentos e grupos de produtos, inserindo no início de cada uma das listas a opção 'TODOS' para que seja possível ignorar um filtro ou os dois.

   O método *montaSELECT()* gera o comando SQL apropriado para a busca dos produtos com estoque acima do máximo. Nesse método teremos também a criação dos campos de diferença entre o estoque atual e o estoque máximo e a valorização dessa diferença. Deve-se, ainda, aplicar os filtros definidos pelo usuário

para departamento e grupo de produtos. Esse método estabelece também o filtro fixo:

```
PRODUTO_ESTOQUE>PRODUTO_EST_MAXIMO
```

Com isso somente os produtos com estoque acima do máximo são listados.

No método *processaRelatorio()* teremos a acumulação apenas do valor da diferença entre o estoque atual e o estoque máximo.

O método *posRelatorio()* exibe a totalização das diferenças.

A listagem da classe **estoqueacimamaximo** é exibida a seguir.

Lista 11: classe_estoqueacimamaximo.inc

```
<?php
/**
 * Classe Estoque Acima do máximo
 */
class estoqueacimamaximo extends base {
     protected $_acumulado = array();

     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'produto';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("PRODUTO_CODIGO","Produto",
                         20,null,null,true,false,true,1,null,null,false,$_conn));
          $this->addCampo(new string("DEPTO_CODIGO","Departamento",
                         10,null,null,true,true,true,1,null,null,false,$_conn));
          $this->addCampo(new string("GRUPO_CODIGO","Grupo",
                         10,null,null,true,true,true,1,null,null,false,$_conn));
          // Campos somente do relatório
          $this->addCampo(new string("PRODUTO_DESC","Descrição",60,null,null));
          $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
          $this->addCampo(new string("DEPTO_DESC","Departamento",40,null,null));
          $this->addCampo(new string("GRUPO_DESC","Grupo",40,null,null));
          $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",
                         12,null,null));
          $this->addCampo(new float("PRODUTO_EST_MAXIMO","Máximo",12,null,null));
          $this->addCampo(new float("DIFERENCA","Diferença",12,null,null));
          $this->addCampo(new dinheiro("VALOR_DIFERENCA","Valor",12,null,null));
          // Acertar filtro
          $this->getCampo("PRODUTO_CODIGO")->setFiltro(false);
          $this->getCampo("PRODUTO_DESC")->setFiltro(false);
          $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
          $this->getCampo("DEPTO_DESC")->setFiltro(false);
          $this->getCampo("GRUPO_DESC")->setFiltro(false);
          $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
          $this->getCampo("PRODUTO_EST_MAXIMO")->setFiltro(false);
          $this->getCampo("DIFERENCA")->setFiltro(false);
          $this->getCampo("VALOR_DIFERENCA")->setFiltro(false);
          // Acertar Campos do relatório
          $this->getCampo("DEPTO_CODIGO")->setRelatorio(false);
          $this->getCampo("GRUPO_CODIGO")->setRelatorio(false);
     }

     /**
      * Sempre executamos o módulo de relatório especial
```

```
 *
 */
public function processaAcao() {
     // Sempre mostra o filtro do relatório nada mais...
     $_depto = new departamento($this->_conn);
     $_depto->Buscar();
     $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
     $_ldep[] = Array("valor"=>'TODOS',"label"=>'TODOS');
     while($_depto->proximo()!==false) {
          $_ldep[] =
               Array(
                    "valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                     "label"=>$_depto->getCampo('DEPTO_DESC')->getValor()
               );
     }
     $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
     $_grupo = new grupo($this->_conn);
     $_grupo->Buscar();
     $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
     $_lgrp[] = Array("valor"=>'TODOS',"label"=>'TODOS');
     while($_grupo->proximo()!==false) {
          $_lgrp[] =
               Array(
                    "valor"=>$_grupo->getCampo('GRUPO_CODIGO')->getValor(),
                    "label"=>$_grupo->getCampo('GRUPO_DESC')->getValor()
               );
     }
     $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
     $this->geraFormularioFiltroRelatorio();
}

/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                            $_orderby=null,$_limit=null,$_extras=null) {
     $_strSQL= "SELECT " .
               "PRODUTO_CODIGO, PRODUTO_DESC, D.DEPTO_DESC," .
               "G.GRUPO_DESC,PRODUTO_ESTOQUE,PRODUTO_EST_MAXIMO," .
               "UNIDADE_CODIGO," .
               "(PRODUTO_ESTOQUE-PRODUTO_EST_MAXIMO) AS DIFERENCA," .
               "(PRODUTO_ESTOQUE-PRODUTO_EST_MAXIMO)*PRODUTO_CUSTOATUAL " .
               "AS VALOR_DIFERENCA " .
               "FROM {$this->_nome_tabela} P    " .
               "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
               "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO ";
     // Filtro
     $_where = Array("P.PRODUTO_ESTOQUE>P.PRODUTO_EST_MAXIMO");
     foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
          if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
               $_campo->setValor($_POST[$_nome]);
               $_where[] = "P.{$_nome}={$_campo->toBD()}";
          }
     }
     $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
     $_strSQL .=   "ORDER BY P.PRODUTO_CODIGO";
     return $_strSQL;
}

/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
```

```
    public function processaRelatorio() {
        // Vamos acumular os valores
        $this->_acumulado['VALOR_DIFERENCA'] +=
                                $this->getCampo("VALOR_DIFERENCA")->getValor();
    }

    /**
     * Exibe os totais do relatório
     *
     * @return array
     */
    public function posRelatorio() {
        $_extra = new atributo("STYLE",
                                "height:40px;color:Navy;font-weight:bold;");
        $_extra_2 = new atributo("STYLE",
                                "height:40px;color:Navy;font-weight:bold;" .
                                "border-top:2px Solid Navy;");
        $_campos = Array(
                    Array(
                        Array(
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new string("","",0,null,null),
                            new dinheiro("VALOR_DIFERENCA","",12,null,null)
                        ),
                        Array(
                            $_extra,
                            8=>$_extra_2
                        )
                    )
                );
        $_campos[0][0][0]->setValor('Totais');
        $_campos[0][0][8]->setValor($this->_acumulado['VALOR_DIFERENCA']);
        return $_campos;
    }
}
?>
```

Com a classe criada precisamos do programa que gera o relatório no sistema. O programa está listado a seguir.

Lista 12: rel_estoqueacimamaximo.php5

```
<?php
/**
 * Relatório: Produtos com estoque acima do máximo
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_estoqueacimamaximo.inc",
              "nome"=>"estoqueacimamaximo"
           );
$_FTitulo = "Relatório de Produtos com estoque acima do máximo";
return include_once("../instanciaclasse.php5");
?>
```

Agora basta seguir o procedimento padrão de incluir o programa no menu do sistema antes de utilizá-lo.

A Figura 8.12 mostra um exemplo do relatório.

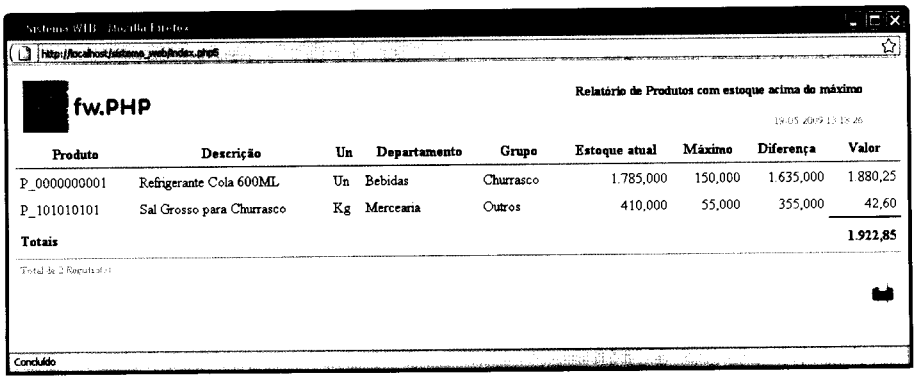


Figura 8.12

Produtos com estoque abaixo do mínimo

Esse relatório oferece ao tomador de decisões uma visão inversa do que foi mostrado no relatório anterior. Enquanto no relatório anterior mostramos como encontrar os produtos que estão com excesso de estoque, nesse vamos exibir a lista de produtos com estoques perigosamente baixos, abaixo do mínimo exigido pela empresa.

O estoque mínimo, conhecido também como estoque de segurança ou estoque de reserva, tem por finalidade determinar a quantidade mínima de estoque de cada produto para que seja possível contornar situações atípicas, tais como aumento na demanda, atraso do fornecedor, excesso de quebras e outras situações não previstas no dia-a-dia do gerenciamento dos estoques.

O que queremos evitar com o controle do estoque mínimo e de outras técnicas de gestão do estoque é a ruptura, ou seja, a falta do produto seja no almoxarifado, no depósito e na loja. A ruptura leva ao atraso na fabricação, à perda de tempo precioso no processo de distribuição e na ponta final leva ao prejuízo, pois muitas vezes o consumidor, ao não encontrar o produto desejado, desiste de toda a compra e certamente procura o concorrente para suprir sua necessidade.

Não devemos confundir o estoque mínimo com o ponto de pedido, ou seja, não se deve esperar o estoque chegar ao ponto mínimo para realizar o pedido de reposição dos estoques. O cálculo do ponto de pedido envolve variáveis como média de consumo do produto, tempo médio de reposição e o próprio estoque mínimo.

O relatório de produtos com estoque abaixo do mínimo possui a mesma estrutura que o relatório de produtos com estoque acima do máximo (mostrado no item

anterior). É necessário somente trocar o campo PRODUTO_EST_MAXIMO por PRODUTO_EST_MINIMO. Além disso devemos alterar o cálculo da diferença para que ela continue positiva. A nova fórmula para o cálculo da diferença é:

```
DIFERENCA = PRODUTO_EST_MINIMO - PRODUTO_ESTOQUE
```

Devemos também alterar o filtro fixo no método *montaSELECT()* para:

```
PRODUTO_EST_MINIMO>PRODUTO_ESTOQUE
```

O restante é uma cópia da classe anterior.

Lista 13: classe_estoqueabaixominimo

```
<?php
/**
 * Classe Estoque Abaixo do mínimo
 */
class estoqueabaixominimo extends base {
    protected $_acumulado = array();

    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'produto';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("PRODUTO_CODIGO","Produto",
                    20,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("DEPTO_CODIGO","Departamento",
                    10,null,null,true,true,true,1,null,null,false,$_conn));
        $this->addCampo(new string("GRUPO_CODIGO","Grupo",
                    10,null,null,true,true,true,1,null,null,false,$_conn));
        // Campos somente do relatório
        $this->addCampo(new string("PRODUTO_DESC","Descrição",60,null,null));
        $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
        $this->addCampo(new string("DEPTO_DESC","Departamento",40,null,null));
        $this->addCampo(new string("GRUPO_DESC","Grupo",40,null,null));
        $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque atual",
                    12,null,null));
        $this->addCampo(new float("PRODUTO_EST_MINIMO","Mínimo",12,null,null));
        $this->addCampo(new float("DIFERENCA","Diferença",12,null,null));
        $this->addCampo(new dinheiro("VALOR_DIFERENCA","Valor",12,null,null));
        // Acertar filtro
        $this->getCampo("PRODUTO_CODIGO")->setFiltro(false);
        $this->getCampo("PRODUTO_DESC")->setFiltro(false);
        $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
        $this->getCampo("DEPTO_DESC")->setFiltro(false);
        $this->getCampo("GRUPO_DESC")->setFiltro(false);
        $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
        $this->getCampo("PRODUTO_EST_MINIMO")->setFiltro(false);
        $this->getCampo("DIFERENCA")->setFiltro(false);
        $this->getCampo("VALOR_DIFERENCA")->setFiltro(false);
        // Acertar Campos do relatório
        $this->getCampo("DEPTO_CODIGO")->setRelatorio(false);
        $this->getCampo("GRUPO_CODIGO")->setRelatorio(false);
    }

    /**
     * Sempre executamos o módulo de relatório especial
     *
```

```
 */
public function processaAcao() {
     // Sempre mostra o filtro do relatório nada mais...
     $_depto = new departamento($this->_conn);
     $_depto->Buscar();
     $this->getCampo("DEPTO_CODIGO")->setComportamento_form('select');
     $_ldep[] = Array("valor"=>'TODOS',"label"=>'TODOS');
     while($_depto->proximo()!==false) {
          $_ldep[] =
               Array(
                    "valor"=>$_depto->getCampo('DEPTO_CODIGO')->getValor(),
                    "label"=>$_depto->getCampo('DEPTO_DESC')->getValor()
               );
     }
     $this->getCampo("DEPTO_CODIGO")->setValor_fixo($_ldep);
     $_grupo = new grupo($this->_conn);
     $_grupo->Buscar();
     $this->getCampo("GRUPO_CODIGO")->setComportamento_form('select');
     $_lgrp[] = Array("valor"=>'TODOS',"label"=>'TODOS');
     while($_grupo->proximo()!==false) {
          $_lgrp[] =
               Array(
                    "valor"=>$_grupo->getCampo('GRUPO_CODIGO')->getValor(),
                    "label"=>$_grupo->getCampo('GRUPO_DESC')->getValor()
               );
     }
     $this->getCampo("GRUPO_CODIGO")->setValor_fixo($_lgrp);
     $this->geraFormularioFiltroRelatorio();
}

/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                            $_orderby=null,$_limit=null,$_extras=null) {
     $_strSQL= "SELECT " .
               "PRODUTO_CODIGO, PRODUTO_DESC, D.DEPTO_DESC," .
               "G.GRUPO_DESC,PRODUTO_ESTOQUE,PRODUTO_EST_MINIMO," .
               "UNIDADE_CODIGO," .
               "(PRODUTO_EST_MINIMO-PRODUTO_ESTOQUE) AS DIFERENCA," .
               "(PRODUTO_EST_MINIMO-PRODUTO_ESTOQUE)*PRODUTO_CUSTOATUAL " .
               "AS VALOR_DIFERENCA " .
               "FROM {$this->_nome_tabela} P     " .
               "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
               "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO ";
     // Filtro
     $_where = Array("P.PRODUTO_EST_MINIMO>P.PRODUTO_ESTOQUE");
     foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
          if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
               $_campo->setValor($_POST[$_nome]);
               $_where[] = "P.{$_nome}={$_campo->toBD()}";
          }
     }
     $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
     $_strSQL .=    "ORDER BY P.PRODUTO_CODIGO";
     return $_strSQL;
}

/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
public function processaRelatorio() {
```

```
            // Vamos acumular os valores
            $this->_acumulado['VALOR_DIFERENCA'] +=
                        $this->getCampo("VALOR_DIFERENCA")->getValor();
        }

        /**
         * Exibe os totais do relatório
         *
         * @return array
         */
        public function posRelatorio() {
            $_extra = new atributo("STYLE",
                                   "height:40px;color:Navy;font-weight:bold;");
            $_extra_2 = new atributo("STYLE",
                                   "height:40px;color:Navy;font-weight:bold;" .
                                   "border-top:2px Solid Navy;");
            $_campos = Array(
                        Array(
                            Array(
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new string("","",0,null,null),
                                new dinheiro("VALOR_DIFERENCA","",12,null,null)
                            ),
                            Array(
                                $_extra,
                                8=>$_extra_2
                            )
                        )
                    );
            $_campos[0][0][0]->setValor('Totais');
            $_campos[0][0][8]->setValor($this->_acumulado['VALOR_DIFERENCA']);
            return $_campos;
        }
}
?>
```

A lista seguinte mostra o programa *rel_estoqueabaixominimo.php5* que será utilizado para gerar o relatório no menu do sistema.

Lista 14: rel_estoqueabaixominimo.php5

```
<?php
/**
 * Relatório: Produtos com estoque abaixo do mínimo
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_estoqueabaixominimo.inc",
              "nome"=>"estoqueabaixominimo"
           );
$_FTitulo = "Relatório de Produtos com estoque abaixo do mínimo";
return include_once("../instanciaclasse.php5");
?>
```

O passo final é incluir o programa no menu do sistema.

A Figura 8.13 mostra um exemplo do relatório de produtos abaixo do estoque mínimo.

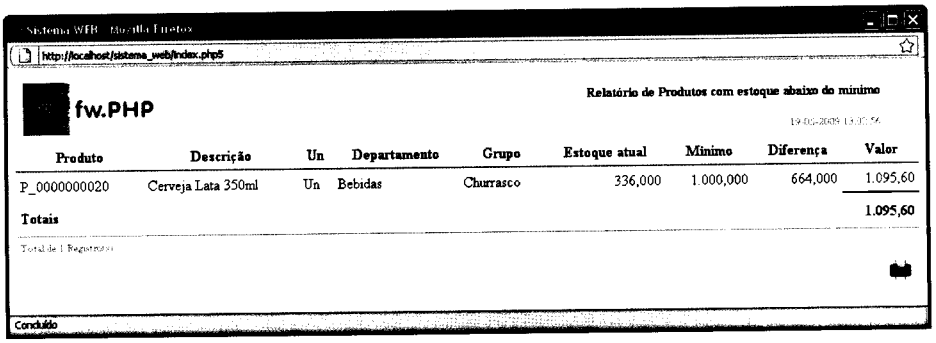


Figura 8.13

Extrato de movimentação

O objetivo desse relatório é apresentar a lista de movimentos de um ou mais produtos em um período escolhido ou, caso o usuário não selecione nenhum período, o histórico completo dos produtos selecionados.

O relatório mostra por produtos todos os movimentos realizados, classificando-os por data e local de estoque.

Os movimentos são valorizados e totalizados para cada produto. Ele mostra os totais de entradas e saídas no período, bem como o valor final acumulado do período (entradas menos saídas).

Filtros do relatório

- Produto inicial
- Produto final
- Data inicial
- Data final

Campos do relatório

Cabeçalho

- Código do produto
- Descrição do produto
- Unidade de medida

Detalhe

- Data do movimento
- Local
- Tipo de movimento
- Entrada/Saída
- Documento
- Quantidade movimentada
- Valor unitário do movimento
- Valor total

Totalização por produto

- Entradas
- Saídas
- Valor total

Esse relatório é diferente dos mostrados anteriormente, pois desta vez temos a geração do relatório com linhas de cabeçalho, para identificação do produto, e de detalhes, com a movimentação do produto. Além disso teremos um totalizador por produto em vez de um totalizador final do relatório. O recurso para executar esse tipo de relatório foi mostrado no capítulo 1 do livro, quando desenvolvemos os recursos adicionais do framework. O exemplo seguinte mostra como desejamos que o relatório seja exibido.

Data	Local	Tipo de movimento	E/S	Documento	Quantidade	Unitário	Total
P_0000000020-Cerveja Lata 350ml, UN:Un							
18-05-2009	Estoque Principal	Entrada por Nota Fiscal	E	1010	240,000	1,45	348,00
19-05-2009	Estoque secundário	Entrada por Nota Fiscal	E	2233	96,000	1,65	158,40
Totais		Entradas:	336,000	Saídas:	0,000	Saldo Valor:	506,40

Figura 8.14

Até o relatório anterior estávamos fundamentando-os ora na tabela de produtos, ora na tabela de estoque por local. Agora vamos trabalhar com a tabela de histórico de movimentação, uma vez que ela contém os detalhes de todos os movimentos dos produtos.

O método construtor da classe descreve os campos relacionados com o relatório, alguns pertecentes à tabela histórico de movimentos (historicomovimento) e outros pertecentes às tabelas auxiliares de produtos (produto), tipos de movimentos (tipomovimento) e locais de estoque (localestoque).

Esse método, como nas demais classes de relatório, separa os campos que são apenas filtros dos que participam da composição do relatório.

```
public function __construct(BancoDados $_conn) {
     parent::__construct($_conn);
     $this->_nome_tabela = 'historicomovimento';
     $this->_class_path = 'estoque';
     $this->addCampo(new string("PRODUTO_CODIGO","Produto Inicial",20,null,null));
     $this->addCampo(new string("PRODUTO_CODIGO_F","Produto Final",20,null,null));
     $this->addCampo(new string("PRODUTO_DESC","Descrição",60,null,null));
     $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
     $this->addCampo(new data("HISTORICO_DATA","Data Inicial",
                    10,null,null,false,false,false,null,null,null,false,$_conn));
     $this->addCampo(new data("HISTORICO_DATA_F","Data Final",10,null,null));
     $this->addCampo(new string("LOCAL_DESC","Local",30,null,null));
     $this->addCampo(new string("TIPOMOV_DESC","Tipo de movimento",30,null,null));
     $this->addCampo(new string("TIPOMOV_TIPO","E/S",1,null,null));
     $this->addCampo(new string("HISTORICO_DOCUMENTO","Documento",20,null,null));
     $this->addCampo(new float("HISTORICO_QUANTIDADE","Quantidade",10,null,null));
     $this->addCampo(new dinheiro("HISTORICO_VALOR_UNIT","Unitário",
                    12,null,null));
     $this->addCampo(new dinheiro("HISTORICO_VALOR_TOTAL","Total",12,null,null));
     // Fitros
     $this->getCampo("LOCAL_DESC")->setFiltro(false);
     $this->getCampo("PRODUTO_DESC")->setFiltro(false);
     $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
     $this->getCampo("TIPOMOV_DESC")->setFiltro(false);
     $this->getCampo("TIPOMOV_TIPO")->setFiltro(false);
     $this->getCampo("HISTORICO_DOCUMENTO")->setFiltro(false);
     $this->getCampo("HISTORICO_QUANTIDADE")->setFiltro(false);
     $this->getCampo("HISTORICO_VALOR_UNIT")->setFiltro(false);
     $this->getCampo("HISTORICO_VALOR_TOTAL")->setFiltro(false);
     //
     $this->getCampo("PRODUTO_CODIGO")->setRelatorio(false);
     $this->getCampo("PRODUTO_CODIGO_F")->setRelatorio(false);
     $this->getCampo("PRODUTO_DESC")->setRelatorio(false);
     $this->getCampo("UNIDADE_CODIGO")->setRelatorio(false);
     $this->getCampo("HISTORICO_DATA_F")->setRelatorio(false);
     $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
     $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
     $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                                  "estoque/classes/classe_produto.inc");
     $this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
     $this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
     $this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                                  "estoque/classes/classe_produto.inc");
}
```

Os campos relacionados ao produto não estão incluídos no relatório. Como temos dois tipos de linhas no relatório, uma para cabeçalho e outra para o detalhe, precisamos deixar marcados como pertecentes ao relatório apenas os campos que compõem a linha de detalhe e os dados do produto estão na linha de cabeçalho do relatório.

O método *processaAcao()* é bem simples. Ele deve apenas executar o método da classe **base** para a exibição do formulário de definição dos filtros do relatório.

O método *montaSELECT()* constrói o comando SQL para recuperação do histórico de movimento dos produtos selecionados no período de tempo estabelecido.

```
/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                            $_orderby=null,$_limit=null,$_extras=null) {
     $this->getCampo("PRODUTO_CODIGO")->setTitulo("Produto");
     $this->getCampo("HISTORICO_DATA")->setTitulo("Data");
     $_strSQL= "SELECT H.*,P.PRODUTO_DESC,P.UNIDADE_CODIGO," .
               "L.LOCAL_DESC,T.TIPOMOV_DESC " .
               "FROM {$this->_nome_tabela} H " .
               "LEFT JOIN PRODUTO P ON H.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
               "LEFT JOIN LOCALESTOQUE L ON H.LOCAL_CODIGO=L.LOCAL_CODIGO " .
               "LEFT JOIN TIPOMOVIMENTO T ON H.TIPOMOV_CODIGO=T.TIPOMOV_CODIGO ";
     // Filtro
     $_where = Array();
     $_operador = Array('PRODUTO_CODIGO'=>'>=','PRODUTO_CODIGO_F'=>'<=',
                        'HISTORICO_DATA'=>'>=','HISTORICO_DATA_F'=>'<=',);
     $_alias    = Array('PRODUTO_CODIGO_F'=>'PRODUTO_CODIGO',
                        'HISTORICO_DATA_F'=>'HISTORICO_DATA');
     foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
          if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
               $_campo->setValor($_POST[$_nome]);
               $_where[] = "H." . (array_key_exists($_nome,$_alias)
                                        ? $_alias[$_nome]
                                        : $_nome) .
                                   (array_key_exists($_nome,$_operador)
                                        ? $_operador[$_nome]
                                        : '=') .
                                   $_campo->toBD();
          }
     }
     if(sizeof($_where)>0) {
          $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
     }
     $_strSQL .= "ORDER BY H.PRODUTO_CODIGO,H.HISTORICO_DATA," .
                 "H.HISTORICO_SEQUENCIA";
     return $_strSQL;
}
```

O método *processaRelatorio()* acumula os totais de entradas e saídas e a valorização da movimentação. Nesse método também controlamos a impressão da data do movimento, evitando que a mesma data seja impressa repetidas vezes para o mesmo produto. Como teremos quebra por produto, não precisamos de acumuladores em vários níveis.

```
/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
public function processaRelatorio() {
     // Vamos acumular os valores
     if($this->getCampo("TIPOMOV_TIPO")->getValor()=='E') {
          $this->_acumulado['TOTAL_ENTRADAS'] +=
                                $this->getCampo('HISTORICO_QUANTIDADE')->getValor();
          $this->_acumulado['TOTAL_VALOR'] +=
                                $this->getCampo('HISTORICO_VALOR_TOTAL')->getValor();
     } else {
          $this->_acumulado['TOTAL_SAIDAS'] +=
                                $this->getCampo('HISTORICO_QUANTIDADE')->getValor();
          $this->_acumulado['TOTAL_VALOR'] -=
                                $this->getCampo('HISTORICO_VALOR_TOTAL')->getValor();
```

```
        }
        if($this->_data_atual==$this->getCampo("HISTORICO_DATA")->getValor()) {
            $this->getCampo("HISTORICO_DATA")->setValor("");
        } else {
            $this->_data_atual = $this->getCampo("HISTORICO_DATA")->getValor();
        }
}
```

Para imprimir os totais dos produtos, bem como para a impressão da linha de cabeçalho em que teremos os dados cadastrais do produto (código, descrição e unidade de medida), utilizamos o método *processaQuebra()*.

O método realiza as seguintes ações, quando necessário:

1. Impressão dos totais do último produto processado

   Na ocorrência da quebra, ou seja, caso o produto atual seja diferente do último processado e não seja o primeiro produto (neste caso não teremos nenhum total para imprimir), ou caso seja o fim do relatório (parâmetro $_fim igual a verdadeiro), o método fará a impressão dos totais do último produto processado.

2. Impressão da linha de cabeçalho do relatório

   Caso tenhamos a quebra do relatório e não seja o fim do relatório ($_fim igual a verdadeiro), o método gera a linha de cabeçalho com os dados do produto.

3. Limpeza das variáveis de acumulação

   Os atributos de acumulação são zerados após a impressão dos totais.

4. Atualização das variáveis de controle

   Os atributos de controle, $_produto_atual e $_data_atual serão atualizados. O primeiro recebe o produto corrente e o segundo o valor null, forçando a impressão da data.

A listagem a seguir mostra o código completo da classe, incluindo o método *processaQuebra()*.

Observe que não definimos o método *posRelatorio()*, pois não temos totais gerais do relatório, apenas os totais por produto.

Lista 15: classe_extratomovimento.inc

```
<?php
/**
 * Classe Extrato de movimentação
 */

class extratomovimento extends base {
    protected $_acumulado  = Array();
    protected $_data_atual = null;
    protected $_produto_atual = null;

    public function __construct(BancoDados $_conn) {
```

```
        parent::__construct($_conn);
        $this->_nome_tabela = 'historicomovimento';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("PRODUTO_CODIGO","Produto Inicial",
                20,null,null));
        $this->addCampo(new string("PRODUTO_CODIGO_F","Produto Final",
                20,null,null));
        $this->addCampo(new string("PRODUTO_DESC","Descrição",60,null,null));
        $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
        $this->addCampo(new data("HISTORICO_DATA","Data Inicial",
                10,null,null,false,false,false,null,null,null,false,$_conn));
        $this->addCampo(new data("HISTORICO_DATA_F","Data Final",10,null,null));
        $this->addCampo(new string("LOCAL_DESC","Local",30,null,null));
        $this->addCampo(new string("TIPOMOV_DESC","Tipo de movimento",
                30,null,null));
        $this->addCampo(new string("TIPOMOV_TIPO","E/S",1,null,null));
        $this->addCampo(new string("HISTORICO_DOCUMENTO","Documento",
                20,null,null));
        $this->addCampo(new float("HISTORICO_QUANTIDADE","Quantidade",
                10,null,null));
        $this->addCampo(new dinheiro("HISTORICO_VALOR_UNIT","Unitário",
                12,null,null));
        $this->addCampo(new dinheiro("HISTORICO_VALOR_TOTAL","Total",
                12,null,null));
        // Fitros
        $this->getCampo("LOCAL_DESC")->setFiltro(false);
        $this->getCampo("PRODUTO_DESC")->setFiltro(false);
        $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
        $this->getCampo("TIPOMOV_DESC")->setFiltro(false);
        $this->getCampo("TIPOMOV_TIPO")->setFiltro(false);
        $this->getCampo("HISTORICO_DOCUMENTO")->setFiltro(false);
        $this->getCampo("HISTORICO_QUANTIDADE")->setFiltro(false);
        $this->getCampo("HISTORICO_VALOR_UNIT")->setFiltro(false);
        $this->getCampo("HISTORICO_VALOR_TOTAL")->setFiltro(false);
        //
        $this->getCampo("PRODUTO_CODIGO")->setRelatorio(false);
        $this->getCampo("PRODUTO_CODIGO_F")->setRelatorio(false);
        $this->getCampo("PRODUTO_DESC")->setRelatorio(false);
        $this->getCampo("UNIDADE_CODIGO")->setRelatorio(false);
        $this->getCampo("HISTORICO_DATA_F")->setRelatorio(false);
        $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                                "estoque/classes/classe_produto.inc");
        $this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                                "estoque/classes/classe_produto.inc");
}

/**
 * Sempre executamos o módulo de relatório especial
 *
 */
public function processaAcao() {
        $this->geraFormularioFiltroRelatorio();
}

/**
 * Uma classe de relatórios deve sempre montar seu select
 *
 */
public function montaSELECT($_campos,$_where=null,
                                $_orderby=null,$_limit=null,$_extras=null) {
```

```
        $this->getCampo("PRODUTO_CODIGO")->setTitulo("Produto");
        $this->getCampo("HISTORICO_DATA")->setTitulo("Data");
        $_strSQL= "SELECT H.*,P.PRODUTO_DESC,P.UNIDADE_CODIGO," .
             "L.LOCAL_DESC,T.TIPOMOV_DESC " .
             "FROM {$this->_nome_tabela} H " .
             "LEFT JOIN PRODUTO P ON H.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
             "LEFT JOIN LOCALESTOQUE L ON H.LOCAL_CODIGO=L.LOCAL_CODIGO " .
             "LEFT JOIN TIPOMOVIMENTO T ON H.TIPOMOV_CODIGO=T.TIPOMOV_CODIGO ";
        // Filtro
        $_where = Array();
        $_operador = Array('PRODUTO_CODIGO'=>'>=','PRODUTO_CODIGO_F'=>'<=',
                           'HISTORICO_DATA'=>'>=','HISTORICO_DATA_F'=>'<=',);
        $_alias    = Array('PRODUTO_CODIGO_F'=>'PRODUTO_CODIGO',
                           'HISTORICO_DATA_F'=>'HISTORICO_DATA');
        foreach($this->filtrarCampos("filtro") as $_nome=>$_campo) {
             if($_POST[$_nome]!=""&&$_POST[$_nome]!="TODOS") {
                  $_campo->setValor($_POST[$_nome]);
                  $_where[] = "H." . (array_key_exists($_nome,$_alias)
                                      ? $_alias[$_nome]
                                      : $_nome) .
                                 (array_key_exists($_nome,$_operador)
                                      ? $_operador[$_nome]
                                      : '=') .
                                 $_campo->toBD();
             }
        }
        if(sizeof($_where)>0) {
             $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
        }
        $_strSQL .=    "ORDER BY H.PRODUTO_CODIGO,H.HISTORICO_DATA," .
                       "H.HISTORICO_SEQUENCIA";
        return $_strSQL;
   }

   /**
    * Devemos pré-processar o relatório antes de ser gerado o html
    *
    */
   public function processaRelatorio() {
        // Vamos acumular os valores
        if($this->getCampo("TIPOMOV_TIPO")->getValor()=='E') {
             $this->_acumulado['TOTAL_ENTRADAS'] +=
                       $this->getCampo('HISTORICO_QUANTIDADE')->getValor();
             $this->_acumulado['TOTAL_VALOR'] +=
                       $this->getCampo('HISTORICO_VALOR_TOTAL')->getValor();
        } else {
             $this->_acumulado['TOTAL_SAIDAS'] +=
                       $this->getCampo('HISTORICO_QUANTIDADE')->getValor();
             $this->_acumulado['TOTAL_VALOR'] -=
                       $this->getCampo('HISTORICO_VALOR_TOTAL')->getValor();
        }
        if($this->_data_atual==$this->getCampo("HISTORICO_DATA")->getValor()) {
             $this->getCampo("HISTORICO_DATA")->setValor("");
        } else {
             $this->_data_atual = $this->getCampo("HISTORICO_DATA")->getValor();
        }
   }

   public function processaQuebra($_fim=false) {
        if($this->_produto_atual!=$this->getCampo("PRODUTO_CODIGO")->getValor()||
           $_fim===true) {
             $_tags = Array();
             if($this->_produto_atual!==null||$_fim===true) {
                  // Totais do anterior
```

```
        $_tr = new tag(new tipotag('TR'));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("COLSPAN",2),

               new atributo("STYLE",
                       "COLOR:#A0A0A0;FONT-WEIGHT:Bold;HEIGHT:25px;" .
                       "border-top:1px solid black;" .
                       "border-bottom:1px solid black;")),
             "Totais"));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("STYLE",
                       "COLOR:#A0A0A0;FONT-WEIGHT:Bold;HEIGHT:25px;" .
                       "text-align:right;border-top:1px solid black;" .
                       "border-bottom:1px solid black;")),
             "Entradas:"));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("STYLE",
                       "COLOR:#A0A0A0;FONT-WEIGHT:Bold;HEIGHT:25px;" .
                       "text-align:right; border-top:1px solid black;" .
                       "border-bottom:1px solid black;")),
             number_format(
                       $this->_acumulado['TOTAL_ENTRADAS'],
                       3,",","."
             )));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("STYLE",
                       "COLOR:#A0A0A0;FONT-WEIGHT:Bold;HEIGHT:25px;" .
                       "text-align:right;border-top:1px solid black;" .
                       "border-bottom:1px solid black;")),
             "Saídas:"));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("STYLE",
                       "COLOR:#A0A0A0;FONT-WEIGHT:Bold;HEIGHT:25px;" .
                       "text-align:right; border-top:1px solid black;" .
                       "border-bottom:1px solid black;")),
             number_format(
                       $this->_acumulado['TOTAL_SAIDAS'],
                       3,",","."
             )));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("STYLE",
                       "COLOR:#A0A0A0;FONT-WEIGHT:Bold;HEIGHT:25px;" .
                       "text-align:right;border-top:1px solid black;" .
                       "border-bottom:1px solid black;")),
             "Saldo Valor:"));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("STYLE",
                       "COLOR:#A0A0A0;FONT-WEIGHT:Bold;HEIGHT:25px;" .
                       "text-align:right; border-top:1px solid black;" .
                       "border-bottom:1px solid black;")),
             number_format(
                       $this->_acumulado['TOTAL_VALOR'],
                       2,",","."
             )));
        $_tags[] = $_tr;
    }
    // Cabeçalho do produto
    if($_fim!==true) {
        $_tr = new tag(new tipotag('TR'));
        $_tr->addSubTag(new tag(new tipotag("TD"),
             Array(new atributo("COLSPAN",7),
                    new atributo("STYLE",
                       "COLOR:Navy;FONT-WEIGHT:Bold;HEIGHT:25px;")),
             "{$this->getCampo("PRODUTO_CODIGO")->getValor()}-" .
```

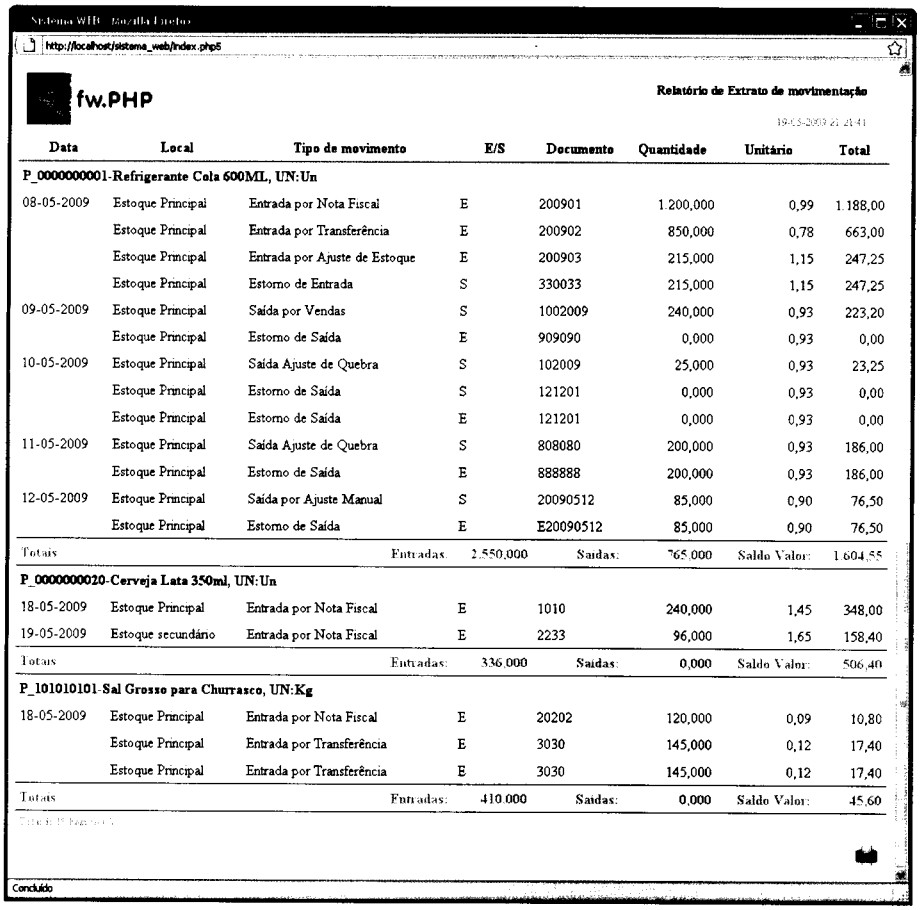
```
                    "{$this->getCampo("PRODUTO_DESC")->toHTML()}, " .
                    "UN:{$this->getCampo("UNIDADE_CODIGO")->getValor()}"));
                $this->_produto_atual =
                            $this->getCampo("PRODUTO_CODIGO")->getValor();
                $this->_acumulado['TOTAL_ENTRADAS'] = 0;
                $this->_acumulado['TOTAL_SAIDAS']   = 0;
                $this->_acumulado['TOTAL_VALOR']    = 0;
                $this->_data_atual = null;
                $_tags[] = $_tr;
            }
            return $_tags;
        } else {
            return null;
        }
    }

}
?>
```

Basta, agora, gerar o programa *rel_extratomovimento.php5* (mostrado na listagem a seguir) e incluí-lo no menu do sistema para que seja possível o acesso ao relatório.

Listing 16: rel_extratomovimento.php5

```
<?php
/**
 * Relatório: Extrato de movimentação
 *
 */
$_classe = Array(
            "arquivo"=>"estoque/classes/classe_extratomovimento.inc",
            "nome"=>"extratomovimento");
$_FTitulo = "Relatório de Extrato de movimentação";
return include_once("../instanciaclasse.php5");
?>
```

A Figura 8.15 exibe um exemplo do relatório de extrato de movimento que acabamos de criar.



fw.PHP

Relatório de Extrato de movimentação

Data	Local	Tipo de movimento	E/S	Documento	Quantidade	Unitário	Total
P_0000000001-Refrigerante Cola 600ML, UN:Un							
08-05-2009	Estoque Principal	Entrada por Nota Fiscal	E	200901	1.200,000	0,99	1.188,00
	Estoque Principal	Entrada por Transferência	E	200902	850,000	0,78	663,00
	Estoque Principal	Entrada por Ajuste de Estoque	E	200903	215,000	1,15	247,25
	Estoque Principal	Estorno de Entrada	S	330033	215,000	1,15	247,25
09-05-2009	Estoque Principal	Saída por Vendas	S	1002009	240,000	0,93	223,20
	Estoque Principal	Estorno de Saída	E	909090	0,000	0,93	0,00
10-05-2009	Estoque Principal	Saída Ajuste de Quebra	S	102009	25,000	0,93	23,25
	Estoque Principal	Estorno de Saída	S	121201	0,000	0,93	0,00
	Estoque Principal	Estorno de Saída	E	121201	0,000	0,93	0,00
11-05-2009	Estoque Principal	Saída Ajuste de Quebra	S	808080	200,000	0,93	186,00
	Estoque Principal	Estorno de Saída	E	888888	200,000	0,93	186,00
12-05-2009	Estoque Principal	Saída por Ajuste Manual	S	20090512	85,000	0,90	76,50
	Estoque Principal	Estorno de Saída	E	E20090512	85,000	0,90	76,50
Totais		Entradas:	2.550,000	Saídas:	765,000	Saldo Valor:	1.604,55
P_0000000020-Cerveja Lata 350ml, UN:Un							
18-05-2009	Estoque Principal	Entrada por Nota Fiscal	E	1010	240,000	1,45	348,00
19-05-2009	Estoque secundário	Entrada por Nota Fiscal	E	2233	96,000	1,65	158,40
Totais		Entradas:	336,000	Saídas:	0,000	Saldo Valor:	506,40
P_1010101010-Sal Grosso para Churrasco, UN:Kg							
18-05-2009	Estoque Principal	Entrada por Nota Fiscal	E	20202	120,000	0,09	10,80
	Estoque Principal	Entrada por Transferência	E	3030	145,000	0,12	17,40
	Estoque Principal	Entrada por Transferência	E	3030	145,000	0,12	17,40
Totais		Entradas:	410,000	Saídas:	0,000	Saldo Valor:	45,60

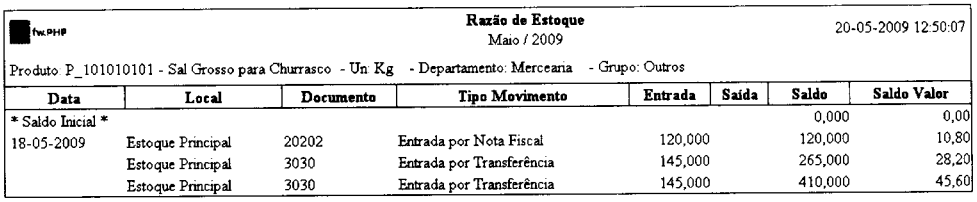
Figura 8.15

Razão de estoque

O relatório de razão de estoque ou kardex de produtos apresenta uma ficha com a movimentação mensal de cada um dos produtos selecionados no mês e ano estabelecidos pelo usuário. Diferentemente do relatório de extrato de movimentação, o razão de estoque contém o saldo anterior do produto no mês anterior ao que está sendo impresso, além disso o razão calcula o saldo, tanto em quantidade quanto em valor, a cada movimento no período.

Por ser um relatório muito específico, não podemos utilizar os recursos existentes no framework para a geração automática de relatórios. Em vez disso construiremos todo o relatório dentro da classe.

A Figura 8.16 indica como deve ser o formato do relatório razão de estoque.



fw.PHP — **Razão de Estoque** — Maio / 2009 — 20-05-2009 12:50:07

Produto P_101010101 - Sal Grosso para Churrasco - Un: Kg - Departamento: Mercearia - Grupo: Outros

Data	Local	Documento	Tipo Movimento	Entrada	Saída	Saldo	Saldo Valor
* Saldo Inicial *						0,000	0,00
18-05-2009	Estoque Principal	20202	Entrada por Nota Fiscal	120,000		120,000	10,80
	Estoque Principal	3030	Entrada por Transferência	145,000		265,000	28,20
	Estoque Principal	3030	Entrada por Transferência	145,000		410,000	45,60

Figura 8.16

Cada produto é exibido em uma ficha separada. Todas as fichas contêm o logo da empresa, o mês e ano de referência, bem como a data de emissão do relatório.

Os seguintes filtros são disponibilizados para o relatório:

- Produto inicial
- Produto final
- Mês de referência
- Ano de referência

Campos do relatório

Cabeçalho

- Código do produto
- Descrição do produto
- Unidade de medida
- Departamento
- Grupo do produto

Detalhe

- Data do movimento
- Local
- Documento
- Tipo de movimento
- Quantidade de entrada
- Quantidade de saída
- Saldo em quantidade
- Saldo em valor

Não há totalizações no relatório, pois o saldo é calculado a cada movimento.

No início da ficha de cada produto o sistema apresenta o saldo inicial, ou seja, o saldo no dia anterior ao do primeiro dia do mês que está sendo impresso.

A classe deve conter alguns atributos para a correta execução do processo:

1. $_html

   Instância da classe **tag**, contém os marcadores html do relatório. É a partir desse atributo que o relatório será enviado para o browser.

2. $_relatorio

   Instância da classe **tag**, contém as fichas dos produtos. Será inserida no atributo $_html antes do envio do relatório para o browser. Cada produto será uma tabela (marcador html TABLE), a qual estará inserida em $_relatorio.

3. $_data_inicial

   Instância da classe **data**, contém a data do dia inicial do período no qual o razão de estoque será emitido, ou seja, caso o mês e o ano de referências sejam Maio e 2009, respectivamente, teremos que $_data_inicial conterá 01-05-2009 como valor. Esse atributo será utilizado na busca dos saldos iniciais, bem como na montagem do filtro de busca dos movimentos.

4. $_saldo_inicial

   Esse atributo conterá os saldos iniciais de todos os produtos que comporão o relatório. Seu formato será o de uma lista, e cada elemento da lista conterá o saldo em quantidade e o saldo em valor de um produto. A chave de indexação da lista será o código do produto. O exemplo a seguir mostra como deve ser a lista:

```
Array(
     '000001'=>Array(0,0),
     '000002'=>Array(10.5,123.67)
     ...
)
```

1. $_produto_atual

   Esse atributo conterá o código do produto que está sendo processado, sendo inicialmente seu valor marcado como nulo (null).

2. $_data_atual

   Contém a última data processada. Esse atributo será utilizado para que a impressão da mesma data no mesmo produto não seja repetida.

3. $_saldo_atual

   Atributo no formato de lista, conterá o saldo atual do produto que está sendo processado. Esse atributo será inicializado a cada produto com o saldo inicial dele tanto em quantidade quanto em valor.

4. $_meses

   Lista dos meses do ano no formato definido no framework para uma lista de valores de um campo como o atributo 'comportamento form' marcado como 'select'.

O método construtor da classe *__construct()* cria os campos conforme a definição de cada um deles. Como esses campos já foram listados em outros relatórios criados anteriormente, não é mais necessário repetir suas definições. Esse método deve, ainda, separar os campos que são filtros do relatório e definir o conteúdo dos campos para mês e ano de referência. Para o mês de referência teremos o atributo $_meses. O mês corrente (date("m")) será marcado como o valor pré-selecionado no campo (fazemos pelo método virtual *setValor()* do campo relacionado). O campo para ano de referência será preenchido com os últimos seis anos, em ordem decrescente, iniciando com o ano atual.

Não marcamos nenhum campo como participante ou não do relatório. Esse controle não é necessário, uma vez que não utilizaremos os recursos do framework para a geração automática do relatório.

O método *processaAcao()* deve apenas executar o método *geraFormularioFiltroRelatorio()* o qual exibe o formulário de filtragem e em seguida processa o relatório.

O método *montaSELECT()* construirá o comando SQL necessário para a busca dos movimentos no período informado e para os produtos selecionados (se for o caso), buscando também os dados complementares nas tabelas de produtos (produto), locais de estoque (localestoque), departamentos (departamento), grupos de produtos (grupos) e tipos de movimentos (tipomovimento).

O método *imprimirRelatorioEspecial()* será totalmente reescrito, porque não podemos utilizar a classe **relatorio** do framework para geração desse relatório específico.

Nesse método teremos todo o processamento do relatório, começando pela determinação do atributo de data inicial ($_data_inicial), seguido da busca dos saldos iniciais dos produtos e da montagem do cabeçalho da página html (é aqui que instanciamos a classe **tag** no atributo $_html). Passamos então para a geração da ficha de cada produto e finalizamos com o envio da página html gerada para o browser.

A geração dos saldos iniciais é realizada no método *getSaldoIncial()*, o qual soma todos os movimentos de entrada e subtrai os de saída de cada produto, considerando os movimentos anteriores ao do período informado para a geração do razão de estoque. O resultado dessa busca é armazenado no atributo $_saldo_inicial e será utilizado durante a impressão das fichas dos produtos.

O método *processaRelatorio()* é utilizado para a composição do saldo do produto, tanto quantidade quanto valor, e também para o controle da impressão da data do movimento.

O método *processaQuebra()* é responsável por gerar a ficha do produto, inserindo o cabeçalho da ficha, bem como as informações do produto e ainda o saldo inicial. É nesse método, quando um novo produto é iniciado, que o saldo acumulado do produto é inicializado com o saldo inicial recuperado do banco de dados, o qual está armazenado no atributo $_saldo_inicial.

A listagem seguinte mostra o conteúdo completo da classe **razaoestoque**.

Lista 17: classe_razaoestoque.inc

```
<?php
/**
 * Classe Razão de Estoque
 */

class razaoestoque extends base {
    protected $_relatorio = null;
    protected $_html = null;
    protected $_data_inicial = null;
    protected $_saldo_inicial = Array();
    protected $_data_atual = null;
    protected $_produto_atual = null;
    protected $_saldo_atual = Array(0,0);
    protected $_meses = Array(
                            Array('valor'=>1,'label'=>"Janeiro"),
                            Array('valor'=>2,'label'=>"Fevereiro"),
                            Array('valor'=>3,'label'=>"Março"),
                            Array('valor'=>4,'label'=>"Abril"),
                            Array('valor'=>5,'label'=>"Maio"),
                            Array('valor'=>6,'label'=>"Junho"),
                            Array('valor'=>7,'label'=>"Julho"),
                            Array('valor'=>8,'label'=>"Agosto"),
                            Array('valor'=>9,'label'=>"Setembro"),
                            Array('valor'=>10,'label'=>"Outubro"),
                            Array('valor'=>11,'label'=>"Novembro"),
                            Array('valor'=>12,'label'=>"Dezembro")
                        );

    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'historicomovimento';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("PRODUTO_CODIGO","Produto Inicial",
                20,null,null));
        $this->addCampo(new string("PRODUTO_CODIGO_F","Produto Final",
                20,null,null));
        $this->addCampo(new inteiro("MES","Mês",2,null,null));
        $this->addCampo(new inteiro("ANO","Ano",4,null,null));
        $this->addCampo(new string("PRODUTO_DESC","Descrição",60,null,null));
        $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
        $this->addCampo(new string("DEPTO_DESC","Departamento",40,null,null));
        $this->addCampo(new string("GRUPO_DESC","Grupo",40,null,null));
        $this->addCampo(new data("HISTORICO_DATA","Data",
                10,null,null,false,false,false,null,null,null,false,$_conn));
        $this->addCampo(new string("LOCAL_DESC","Local",30,null,null));
        $this->addCampo(new string("TIPOMOV_DESC","Tipo de movimento",
                30,null,null));
        $this->addCampo(new string("TIPOMOV_TIPO","E/S",1,null,null));
        $this->addCampo(new string("HISTORICO_DOCUMENTO","Documento",
                20,null,null));
```

```
        $this->addCampo(new float("HISTORICO_QUANTIDADE","Quantidade",
                10,null,null));
        $this->addCampo(new dinheiro("HISTORICO_VALOR_TOTAL","Total",
                12,null,null));
        $this->addCampo(new float("SALDO_QUANTIDADE","Quantidade",10,null,null));
        $this->addCampo(new dinheiro("SALDO_VALOR","Total",12,null,null));
        // Fitros
        $this->getCampo("LOCAL_DESC")->setFiltro(false);
        $this->getCampo("PRODUTO_DESC")->setFiltro(false);
        $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
        $this->getCampo("TIPOMOV_DESC")->setFiltro(false);
        $this->getCampo("TIPOMOV_TIPO")->setFiltro(false);
        $this->getCampo("HISTORICO_DOCUMENTO")->setFiltro(false);
        $this->getCampo("HISTORICO_DATA")->setFiltro(false);
        $this->getCampo("HISTORICO_QUANTIDADE")->setFiltro(false);
        $this->getCampo("HISTORICO_VALOR_TOTAL")->setFiltro(false);
        $this->getCampo("SALDO_QUANTIDADE")->setFiltro(false);
        $this->getCampo("SALDO_VALOR")->setFiltro(false);
        $this->getCampo("DEPTO_DESC")->setFiltro(false);
        $this->getCampo("GRUPO_DESC")->setFiltro(false);
        //
        $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                        "estoque/classes/classe_produto.inc");
        $this->getCampo("PRODUTO_CODIGO_F")->setComportamento_form('ajax');
        $this->getCampo("PRODUTO_CODIGO_F")->setClasse("produto");
        $this->getCampo("PRODUTO_CODIGO_F")->setArquivo_Classe(
                        "estoque/classes/classe_produto.inc");

        $this->getCampo("MES")->setComportamento_form('select');
        $this->getCampo("ANO")->setComportamento_form('select');
        $this->getCampo("MES")->setValor_fixo($this->_meses);
        $this->getCampo("MES")->setValor(date("m"));
        for($_ano=date("Y");$_ano>=date("Y")-5;$_ano--) {
            $_anos[] = Array('valor'=>$_ano,'label'=>$_ano);
        }
        $this->getCampo("ANO")->setValor_fixo($_anos);
    }

    /**
     * Sempre executamos o módulo de relatório especial
     *
     */
    public function processaAcao() {
        $this->geraFormularioFiltroRelatorio();
    }

    /**
     * Gera os saldos iniciais dos produtos
     *
     */
    protected function getSaldoInicial() {
        $_strSQL = "SELECT PRODUTO_CODIGO," .
                   "SUM(HISTORICO_QUANTIDADE*" .
                   "(CASE TIPOMOV_TIPO WHEN 'E' THEN 1 ELSE -1 END)) " .
                   "AS QUANTIDADE," .
                   "SUM(HISTORICO_VALOR_TOTAL*" .
                   "(CASE TIPOMOV_TIPO WHEN 'E' THEN 1 ELSE -1 END)) AS VALOR " .
                   "FROM {$this->_nome_tabela} " .
                   "WHERE HISTORICO_DATA < {$this->_data_inicial->toBD()}";
        if($_POST["PRODUTO_CODIGO"]!="") {
            $_strSQL .= " AND PRODUTO_CODIGO>='{$_POST["PRODUTO_CODIGO"]}'";
        }
```

```
        if($_POST["PRODUTO_CODIGO_F"]!="") {
            $_strSQL .= " AND PRODUTO_CODIGO<='{$_POST["PRODUTO_CODIGO_F"]}'";
        }
        $_strSQL .= " GROUP BY PRODUTO_CODIGO";
        $this->_saldo_inicial = Array();
        if($this->_conn->executaSQL($_strSQL)!==false&&
           $this->_conn->getNumRows()>0) {
            while(($_dados=$this->_conn->proximo())!==false) {
                $this->_saldo_inicial[$_dados['PRODUTO_CODIGO']] =
                        Array(
                            $_dados['QUANTIDADE'],
                            $_dados['VALOR']
                        );
            }
        }
    }


    /**
     * Uma classe de relatórios deve sempre montar seu select
     *
     */
    public function montaSELECT($_campos,$_where=null,
                                $_orderby=null,$_limit=null,$_extras=null) {
        $this->getCampo("PRODUTO_CODIGO")->setTitulo("Produto");
        $this->getCampo("HISTORICO_DATA")->setTitulo("Data");
        $_strSQL= "SELECT H.*,P.PRODUTO_DESC,P.UNIDADE_CODIGO," .
            "L.LOCAL_DESC,T.TIPOMOV_DESC,D.DEPTO_DESC,G.GRUPO_DESC " .
            "FROM {$this->_nome_tabela} H " .
            "LEFT JOIN PRODUTO P ON H.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
            "LEFT JOIN DEPARTAMENTO D ON P.DEPTO_CODIGO=D.DEPTO_CODIGO " .
            "LEFT JOIN GRUPO G ON P.GRUPO_CODIGO=G.GRUPO_CODIGO " .
            "LEFT JOIN LOCALESTOQUE L ON H.LOCAL_CODIGO=L.LOCAL_CODIGO " .
            "LEFT JOIN TIPOMOVIMENTO T ON H.TIPOMOV_CODIGO=T.TIPOMOV_CODIGO ";
        // Filtro
        $_where = Array();
        $this->getCampo("HISTORICO_DATA")->setValor(
                $this->_data_inicial->getValor());
        $_where[] = "H.HISTORICO_DATA>=" .
                "{$this->getCampo("HISTORICO_DATA")->toBD()}";
        $this->getCampo("HISTORICO_DATA")->setValor(
                date("d-m-Y",mktime(0,0,0,$_POST['MES']+1,0,$_POST['ANO'])));
        $_where[] = "H.HISTORICO_DATA<=" .
                "{$this->getCampo("HISTORICO_DATA")->toBD()}";
        if($_POST["PRODUTO_CODIGO"]!="") {
            $_where[] = "H.PRODUTO_CODIGO>='{$_POST["PRODUTO_CODIGO"]}'";
        }
        if($_POST["PRODUTO_CODIGO_F"]!="") {
            $_where[] = "H.PRODUTO_CODIGO<='{$_POST["PRODUTO_CODIGO_F"]}'";
        }
        $_strSQL .= " WHERE " . implode(" AND ",$_where) . " ";
        $_strSQL .= "ORDER BY H.PRODUTO_CODIGO,H.HISTORICO_DATA," .
                    "H.HISTORICO_SEQUENCIA";
        return $_strSQL;
    }

    /**
     * Gera o cabeçalho inicial do sistema
     *
     */
    protected function addCabecalho() {
        $this->_html = new tag(new tipotag("HTML"));
        $this->_html->addSubTag(new tag(new tipotag("HEAD")));
        $this->_html->getLastSubTag()->addSubTag(
```

```
                new tag(new tipotag("TITLE"),null,"Sistema WEB"));
        $this->_html->addSubTag(
                new tag(new tipotag("BODY"),
                        Array(new atributo("WIDTH","1010"))));
}

/**
 * Impressão do relatório
 *
 */
public function imprimirRelatorioEspecial() {
     // Armazena a Data inicial de referência
     $this->_data_inicial = $this->getCampo("HISTORICO_DATA");
     $this->_data_inicial->setValor(
               date("d-m-Y",mktime(0,0,0,$_POST['MES'],1,$_POST['ANO'])));
     // Gera os saldos iniciais dos produtos
     $this->getSaldoInicial();
     $this->addCabecalho();
     // Imprime o Relatório
     $this->_relatorio = new tag(new tipotag("DIV"));
     if($this->_conn->executaSQL($this->montaSELECT(""))!==false&&
        $this->_conn->getNumRows()>0) {
          while($this->proximo()) {
               $this->processaQuebra();
               $this->processaRelatorio();
               $this->_relatorio->getLastSubTag()->addSubTag(
                    new tag(new tipotag("TR")));
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;" .
                                                "padding-left:5px;")),
                         $this->getCampo("HISTORICO_DATA")->toPrint()));
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;")),
                         $this->getCampo("LOCAL_DESC")->toPrint()));
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;")),
                         $this->getCampo("HISTORICO_DOCUMENTO")->toPrint()));
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;")),
                         $this->getCampo("TIPOMOV_DESC")->toPrint()));
               $_entrada = ($this->getCampo("TIPOMOV_TIPO")->getValor()=='E'
                         ? $this->getCampo("HISTORICO_QUANTIDADE")->toPrint()
                         : "");
               $_saida   = ($this->getCampo("TIPOMOV_TIPO")->getValor()=='S'
                         ? $this->getCampo("HISTORICO_QUANTIDADE")->toPrint()
                         : "");
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;" .
                                                "text-align:right;")),
                         $_entrada));
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;" .
                                                "text-align:right;")),
                         $_saida));
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;" .
                                                "text-align:right;")),
```

```
                    $this->getCampo("SALDO_QUANTIDADE")->toPrint()));
               $this->_relatorio->getLastSubTag()->getLastSubTag()->
                    addSubTag(new tag(new tipotag("TD"),
                         Array(new atributo("STYLE","border:1px none;" .
                                                    "text-align:right;")),
                         $this->getCampo("SALDO_VALOR")->toPrint()));
          }
     } else {
          $this->_relatorio->setValor(" * Nenhum Registro *");
     }
     $this->addbotaoimpressao();
     $this->_html->getLastSubTag()->addSubTag($this->_relatorio);
     echo utf8_encode($this->_html->toHTML());
}

/**
 * Adiciona o botão de impressão do relatório
 *
 */
protected function addbotaoimpressao() {
     $this->_relatorio->addSubTag(new tag(new tipotag("DIV"),
          Array(new atributo("ID","DIVPRINT"),
               new atributo("STYLE","padding:10px;text-align:right;"))));
     $this->_relatorio->getLastSubTag()->addSubTag(
          new tag(new tipotag("IMG",false),
               Array(new atributo("SRC","framework/imagens/print.png"),
                 new atributo("BORDER",0),
                 new atributo("ID","BTN_PRINT"),
                 new atributo("TITLE","Imprimir"),
                 new atributo("STYLE","cursor:pointer"),
                 new atributo("ONCLICK",
                    "this.parentNode.style.display='none';window.print();" .
                    "var obj=this;setTimeout(function(){" .
                    "obj.parentNode.style.display='block';},3000);")
     )));
}

/**
 * Devemos pré-processar o relatório antes de ser gerado o html
 *
 */
public function processaRelatorio() {
     // Vamos acumular os valores
     if($this->getCampo("TIPOMOV_TIPO")->getValor()=='E') {
          $this->_saldo_atual[0] +=
                    $this->getCampo('HISTORICO_QUANTIDADE')->getValor();
          $this->_saldo_atual[1] +=
                    $this->getCampo('HISTORICO_VALOR_TOTAL')->getValor();
     } else {
          $this->_saldo_atual[0] -=
                    $this->getCampo('HISTORICO_QUANTIDADE')->getValor();
          $this->_saldo_atual[1] -=
                    $this->getCampo('HISTORICO_VALOR_TOTAL')->getValor();
     }
     $this->getCampo("SALDO_QUANTIDADE")->setValor($this->_saldo_atual[0]);
     $this->getCampo("SALDO_VALOR")->setValor($this->_saldo_atual[1]);
     if($this->_data_atual==$this->getCampo("HISTORICO_DATA")->getValor()) {
          $this->getCampo("HISTORICO_DATA")->setValor("");
     } else {
          $this->_data_atual = $this->getCampo("HISTORICO_DATA")->getValor();
     }
}

public function processaQuebra($_fim=false) {
```

```
        if($this->_produto_atual!=$this->getCampo("PRODUTO_CODIGO")->getValor())
{
            if($this->_produto_atual!==null) {
                $this->_relatorio->addSubTag(new tag(new tipotag("BR",false)));
                $this->_relatorio->addSubTag(new tag(new tipotag("BR",false)));
            }
            $this->_relatorio->addSubTag(new tag(new tipotag("TABLE"),
                Array(new atributo("STYLE","border:1px solid black;" .
                                   "border-collapse:collapse;padding:3px;"),
                   new atributo("CELLSPACING",0),
                   new atributo("BORDER",0),
                   new atributo("WIDTH",980))));
            // Cabeçalho
            $this->_relatorio->getLastSubTag()->addSubTag(
                new tag(new tipotag("TR")));
            $this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
                new tag(new tipotag("TD"),
                     Array(new atributo("COLSPAN",2),
                       new atributo("STYLE","border:1px none;" .
                                          "padding-left:5px;")))));
            $this->_relatorio->getLastSubTag()->getLastSubTag()->
                getLastSubTag()->addSubTag(new tag(new tipotag("IMG",false),
                     Array(new atributo("SRC","framework/imagens/logo.png"),
                       new atributo("BORDER",0),
                       new atributo("ALIGN","BOTTOM"),
                       new atributo("WIDTH",60))));
            $this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
                new tag(new tipotag("TD"),
                          Array(new atributo("COLSPAN",4),
                            new atributo("STYLE","border:1px none;" .
                                         "text-align:center;")),
                          "<b>Razão de Estoque</b>" .
                          "<br>{$this->_meses[$_POST['MES']-1]['label']} / " .
                          "{$_POST['ANO']}"));
            $this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
                new tag(new tipotag("TD"),
                     Array(new atributo("COLSPAN",2),
                       new atributo("STYLE","border:1px none;" .
                                         "text-align:right;padding-right:5px;")),
                     date("d-m-Y H:i:s ")));
            // Produto
            $this->_relatorio->getLastSubTag()->addSubTag(
                new tag(new tipotag("TR")));
            $this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
                new tag(new tipotag("TD"),
                     Array(new atributo("COLSPAN",8),
                       new atributo("STYLE","border:1px none;color:Navy;" .
                                         "height:35px;padding-left:5px;")),
                     "Produto: {$this->getCampo("PRODUTO_CODIGO")->toHTML()}" .
                     " - {$this->getCampo("PRODUTO_DESC")->toHTML()}" .
                     "  - " .
                     "Un: {$this->getCampo("UNIDADE_CODIGO")->toHTML()}" .
                     "    - " .
                     "Departamento: {$this->getCampo("DEPTO_DESC")->toHTML()}" .
                     "    - " .
                     "Grupo: {$this->getCampo("GRUPO_DESC")->toHTML()}"));
            // cabeçalho do razão
            $this->_relatorio->getLastSubTag()->addSubTag(
                new tag(new tipotag("TR")));
            $this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
                new tag(new tipotag("TD"),
                     Array(new atributo("STYLE","border:1px solid black;" .
                                         "font-weight:bold;text-align:center;")),
                     "Data"));
```

```
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px solid black;" .
                                "font-weight:bold;text-align:center;")),
          "Local"));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px solid black;" .
                                "font-weight:bold;text-align:center;")),
          "Documento"));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px solid black;" .
                                "font-weight:bold;text-align:center;")),
          "Tipo Movimento"));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px solid black;" .
                                "font-weight:bold;text-align:center;")),
          "Entrada"));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px solid black;" .
                                "font-weight:bold;text-align:center;")),
          "Saída"));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px solid black;" .
                                "font-weight:bold;text-align:center;")),
          "Saldo"));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px solid black;" .
                                "font-weight:bold;text-align:center;")),
          "Saldo Valor"));
// Saldo Inicial do Produto
$this->_saldo_atual =
          $this->_saldo_inicial[
               $this->getCampo("PRODUTO_CODIGO")->getValor()
          ];
$this->_relatorio->getLastSubTag()->addSubTag(
     new tag(new tipotag("TR")));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("COLSPAN",3),
            new atributo("STYLE","border:1px none;" .
                                  "padding-left:5px;")),
          " * Saldo Inicial *"));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("COLSPAN",3),
            new atributo("STYLE","border:1px none;")))));
$this->getCampo("SALDO_QUANTIDADE")->setValor(
                        $this->_saldo_atual[0]);
$this->getCampo("SALDO_VALOR")->setValor($this->_saldo_atual[1]);
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px none;" .
                                "text-align:right;")),
          $this->getCampo("SALDO_QUANTIDADE")->toPrint()));
$this->_relatorio->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TD"),
          Array(new atributo("STYLE","border:1px none;" .
                                "text-align:right;")),
```

```
                    $this->getCampo("SALDO_VALOR")->toPrint()));
            $this->_produto_atual =
                                $this->getCampo("PRODUTO_CODIGO")->getValor();
            $this->_data_atual = null;

        }
    }

}
?>
```

Agora só resta criar o programa de emissão do razão de estoque, o qual chamaremos de *rel_razaoestoque.php5*, incluí-lo no menu do sistema, dentro de relatórios no módulo de estoque.

Lista 18: rel_razaoestoque.php5

```
<?php
/**
 * Relatório: Extrato de movimentação
 *
 */
$_classe = Array(
            "arquivo"=>"estoque/classes/classe_razaoestoque.inc",
            "nome"=>"razaoestoque");
$_FTitulo = "Relatório de Razão de Estoque";
return include_once("../instanciaclasse.php5");
?>
```

As Figuras 8.17 e 8.18 mostram o formulário de filtragem do razão de estoque e um exemplo do relatório respectivamente.

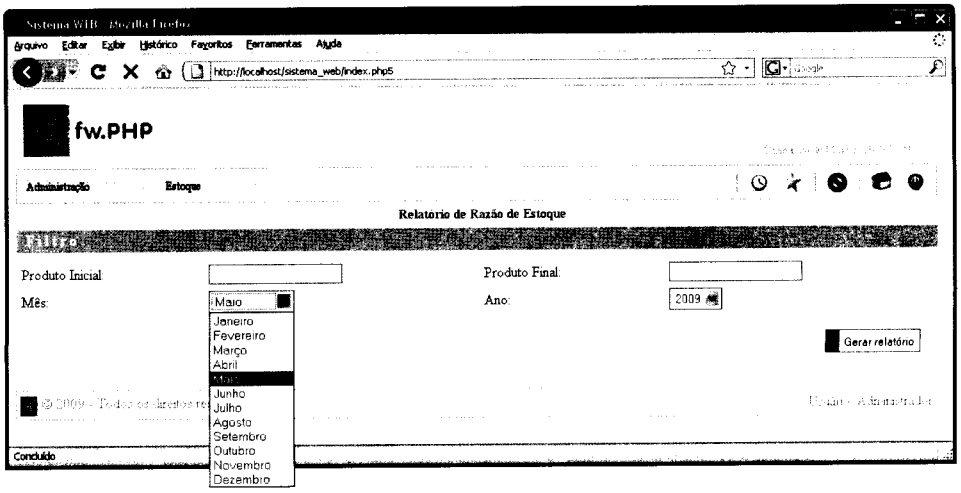
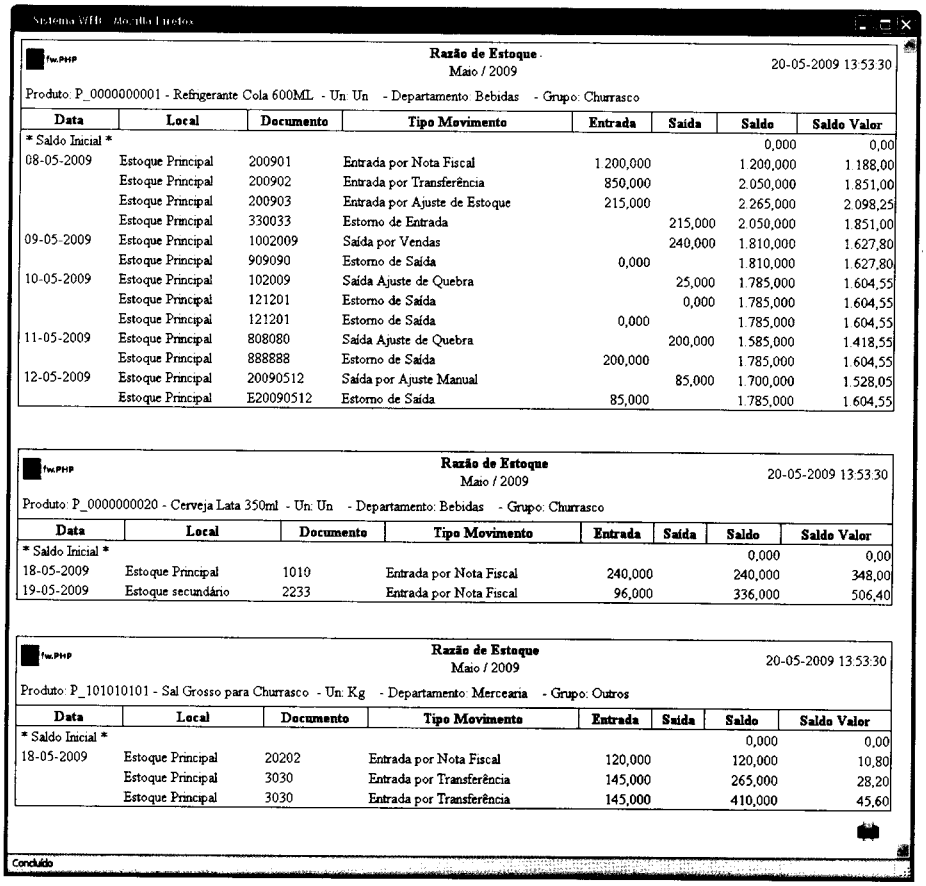


Figura 8.17



Razão de Estoque
Maio / 2009
20-05-2009 13:53:30

Produto: P_0000000001 - Refrigerante Cola 600ML - Un: Un - Departamento: Bebidas - Grupo: Churrasco

Data	Local	Documento	Tipo Movimento	Entrada	Saída	Saldo	Saldo Valor
* Saldo Inicial *						0,000	0,00
08-05-2009	Estoque Principal	200901	Entrada por Nota Fiscal	1.200,000		1.200,000	1.188,00
	Estoque Principal	200902	Entrada por Transferência	850,000		2.050,000	1.851,00
	Estoque Principal	200903	Entrada por Ajuste de Estoque	215,000		2.265,000	2.098,25
	Estoque Principal	330033	Estorno de Entrada		215,000	2.050,000	1.851,00
09-05-2009	Estoque Principal	1002009	Saída por Vendas		240,000	1.810,000	1.627,80
	Estoque Principal	909090	Estorno de Saída	0,000		1.810,000	1.627,80
10-05-2009	Estoque Principal	102009	Saída Ajuste de Quebra		25,000	1.785,000	1.604,55
	Estoque Principal	121201	Estorno de Saída		0,000	1.785,000	1.604,55
	Estoque Principal	121201	Estorno de Saída	0,000		1.785,000	1.604,55
11-05-2009	Estoque Principal	808080	Saída Ajuste de Quebra		200,000	1.585,000	1.418,55
	Estoque Principal	888888	Estorno de Saída	200,000		1.785,000	1.604,55
12-05-2009	Estoque Principal	20090512	Saída por Ajuste Manual		85,000	1.700,000	1.528,05
	Estoque Principal	E20090512	Estorno de Saída	85,000		1.785,000	1.604,55

Razão de Estoque
Maio / 2009
20-05-2009 13:53:30

Produto: P_0000000020 - Cerveja Lata 350ml - Un: Un - Departamento: Bebidas - Grupo: Churrasco

Data	Local	Documento	Tipo Movimento	Entrada	Saída	Saldo	Saldo Valor
* Saldo Inicial *						0,000	0,00
18-05-2009	Estoque Principal	1010	Entrada por Nota Fiscal	240,000		240,000	348,00
19-05-2009	Estoque secundário	2233	Entrada por Nota Fiscal	96,000		336,000	506,40

Razão de Estoque
Maio / 2009
20-05-2009 13:53:30

Produto: P_101010101 - Sal Grosso para Churrasco - Un: Kg - Departamento: Mercearia - Grupo: Outros

Data	Local	Documento	Tipo Movimento	Entrada	Saída	Saldo	Saldo Valor
* Saldo Inicial *						0,000	0,00
18-05-2009	Estoque Principal	20202	Entrada por Nota Fiscal	120,000		120,000	10,80
	Estoque Principal	3030	Entrada por Transferência	145,000		265,000	28,20
	Estoque Principal	3030	Entrada por Transferência	145,000		410,000	45,60

Concluído

Figura 8.18

Curva ABC de estoque

O último relatório mostra aos usuários a classificação dos estoques pelo método da curva ABC que se baseia no diagrama de Pareto (desenvolvido por Vilfredo Pareto no século XIX) o qual classifica os itens conforme sua participação no total analisado. O princípio 80-20 (ou lei de Pareto) afirma que 20% dos itens são responsáveis por 80% do total analisado (no caso por 80% dos estoques em valor).

O objetivo com esse relatório é mostrar ao tomador de decisões os itens mais importantes do estoque, aqueles que representam 80% do valor total dos estoques da empresa. Esses itens devem ser considerados como de alta prioridade, requerendo muita atenção do gestor. O relatório exibe também os itens que exigem atenção mediana, em geral representando cerca de 15% do valor total. Por último temos os itens que representam cerca de 5% do valor total dos estoques, geralmente em grande número (estima-se

que 50% dos itens façam parte dessa classe), exigem pouca atenção do gestor (mas é necessário algum gerenciamento de qualquer forma).

A tabela seguinte mostra a distribuição esperada em uma curvas ABC.

Curva	% sobre o Valor Total	% de itens
A	80%	20%
B	15%	30%
C	5%	50%

Existem várias formas para montagem do cálculo da curva ABC. Alguns utilizam o estoque atual, outros o estoque médio nos últimos 12 meses e há ainda os que utilizam a média de consumo em determinado período (em geral 12 meses).

Nós vamos utilizar o estoque atual dos itens, podendo valorá-los pelo custo médio ou pelo custo atual dos produtos.

Para gerar o relatório, em primeiro lugar é preciso conhecer o total dos estoques conforme o critério de valorização definido. Depois precisamos apenas listar os itens em ordem descrecente de valor e conforme o percentual de participação acumulado for evoluindo, uma curva é definida para o item.

Os percentuais 80,15 e 5 são referências, não sendo de forma nenhuma valores rígidos, ou seja, esses limites podem variar conforme os percentuais calculados. O critério utilizado de corte é a proximidade entre o último item abaixo do percentual estabelecido comparada com a proximidade do próximo item que ultrapassa esse percentual. Por exemplo, vamos supor que o acumulado até o momento é de 79,55% e o próximo item leva o acumulado para 80,10%. Ora, 80,10 está mais próximo de 80 que 79,55, logo consideramos que o item que leva o acumulado para 80,10 ainda pertence à curva 'A'.

Não teremos um filtro para relatório, apenas oferecemos ao usuário a possibilidade de gerar a curva ABC pelo custo médio dos produtos ou pelo custo atual.

Não haverá nenhum totalizador para esse relatório.

Os seguintes campos vão compor o relatório:

- Código do produto
- Descrição do produto
- Unidade de medida
- Estoque atual
- Custo (médio ou atual)
- Valor do estoque
- Percentual de participação no total

- Percentual acumulado
- Posição do produto em ordem crescente
- Curva (A, B ou C)

O campo de valor do estoque é calculado na geração do relatório, sendo uma instância da classe **dinheiro**. Sua fórmula é:

```
VALOR_TOTAL = CUSTO * PRODUTO_ESTOQUE
```

Em que o CUSTO pode ser PRODUTO_CUSTOMEDIO ou então PRODUTO_CUSTOATUAL conforme a escolha do usuário.

O campo de participação do produto no total é calculado, sendo uma instância da classe **float**. Sua fórmula é:

```
PERC = (VALOR_TOTAL / TOTAL_ESTOQUES)*100
```

O total de estoque será calculado previamente pela classe.

O campo Percentual acumulado é também calculado e é uma instância da classe **float**. Esse campo vai de 0% a 100%.

O campo de posição do produto na lista é calculado conforme a lista avança. Ele é uma instância da classe **inteiro**.

O campo que informa a curva à qual o produto pertence é uma instância da classe **string**. Seu valor é determinado conforme o percentual acumulado até o produto atual (juntamente com a regra de proximidade mostrada anteriormente). Esse campo pode assumir os valores A, B ou C.

A classe para emissão do relatório deve criar os campos no método construtor da classe, separando os campos que fazem parte do formulário de filtro (neste caso apenas a determinação do tipo de custo) e quais são parte do relatório.

O método *processaAcao()* executa o procedimento-padrão de chamar o método *geraFormularioFiltroRelatorio()* da classe **base**.

Uma vez que precisamos calcular o total do estoque antes da emissão do relatório, devemos alterar o método *imprimirRelatorioEspecial()* para que antes da impressão, executada pelo mesmo método na classe **base**, os totais sejam calculados. O cálculo do total é bem simples. Precisamos apenas executar um comando SQL de soma e armazenar o resultado em um atributo da classe.

```
/**
 * Precisamos pré-processar os dados antes
 * de imprimir
 *
 */
public function imprimirRelatorioEspecial() {
    $_custo = ($_POST['CUSTO']=='CM'
```

```
                    ? 'PRODUTO_CUSTOMEDIO'
                    : 'PRODUTO_CUSTOATUAL');
    $this->getCampo("CUSTO")->SetTitulo($_POST['CUSTO']=='CM'
                    ? 'Custo Médio'
                    : 'Custo Atual');
    $_strSQL = "SELECT SUM(PRODUTO_ESTOQUE*{$_custo}) AS TOTAL " .
               "FROM {$this->_nome_tabela} " .
               "WHERE PRODUTO_ESTOQUE>0 ";
    if($this->_conn->executaSQL($_strSQL)!==false) {
        $_dados = $this->_conn->proximo();
        $this->_total = $_dados['TOTAL'];
    }
    parent::imprimirRelatorioEspecial();
}
```

O método *montaSELECT()* é bem simples, retornando o comando SQL necessário para a busca dos produtos. O detalhe importante é sua classificação em ordem decrescente de valor de estoque. Lembrando que a valorização pode ser tanto por custo médio quanto por custo total.

A classificação do produto como A,B ou C é feita no método *processaRelatorio()*, no qual atribuímos os valores para os campos de percentual, percentual acumulado, posição do item na lista e Tipo de Curva.

A lista a seguir mostra a classe **curvaabc**.

Lista 19: classe_curvaabc.inc

```
<?php
/**
 * Classe Curva ABC de Estoque
 */
class curvaabc extends base {
    protected $_total = 0;
    protected $_ordem = 0;
    protected $_perc = 0;

    public function __construct(BancoDados $_conn) {
        parent::__construct($_conn);
        $this->_nome_tabela = 'produto';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("PRODUTO_CODIGO","Produto",20,null,null));
        // Campos somente do relatório
        $this->addCampo(new string("PRODUTO_DESC","Descrição",60,null,null));
        $this->addCampo(new string("UNIDADE_CODIGO","Un",2,null,null));
        $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque",12,null,null));
        $this->addCampo(new dinheiro("CUSTO","Custo",12,null,null));
        $this->addCampo(new dinheiro("VALOR_TOTAL","Valor Total",12,null,null));
        $this->addCampo(new float("PERC","Perc.(%)",5,null,null));
        $this->addCampo(new float("ACUM","Acumulado(%)",5,null,null));
        $this->addCampo(new inteiro('POSIC','Posição',3,null,null));
        $this->addCampo(new string("CURVA","Curva",1,null,null));
        // Acertar filtro
        $this->getCampo("PRODUTO_CODIGO")->setFiltro(false);
        $this->getCampo("PRODUTO_DESC")->setFiltro(false);
        $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
        $this->getCampo("PRODUTO_ESTOQUE")->setFiltro(false);
        $this->getCampo("VALOR_TOTAL")->setFiltro(false);
        $this->getCampo("PERC")->setFiltro(false);
```

```
        $this->getCampo("POSIC")->setFiltro(false);
        $this->getCampo("CURVA")->setFiltro(false);
        $this->getCampo("ACUM")->setFiltro(false);
        $this->getCampo("PERC")->setCasas_Decimais(2);
        $this->getCampo("CUSTO")->setComportamento_Form('radio');
        $this->getCampo("CUSTO")->setValor_Fixo(
                                Array(
                                    Array('valor'=>'CM',
                                        'label'=>'Custo Médio',
                                        'marcar'=>true),
                                    Array('valor'=>'CA',
                                        'label'=>'Custo Atual')
                                ));
    }

    /**
     * Sempre executamos o módulo de relatório especial
     *
     */
    public function processaAcao() {
        // Sempre mostra o filtro do relatório nada mais...
        $this->geraFormularioFiltroRelatorio();
    }

    /**
     * Precisamos pré-processar os dados antes
     * de imprimir
     *
     */
    public function imprimirRelatorioEspecial() {
        $_custo = ($_POST['CUSTO']=='CM'
                            ? 'PRODUTO_CUSTOMEDIO'
                            : 'PRODUTO_CUSTOATUAL');
        $this->getCampo("CUSTO")->SetTitulo($_POST['CUSTO']=='CM'
                            ? 'Custo Médio'
                            : 'Custo Atual');
        $_strSQL = "SELECT SUM(PRODUTO_ESTOQUE*{$_custo}) AS TOTAL" .
                   " FROM {$this->_nome_tabela} " .
                   "WHERE PRODUTO_ESTOQUE>0 ";
        if($this->_conn->executaSQL($_strSQL)!==false) {
            $_dados = $this->_conn->proximo();
            $this->_total = $_dados['TOTAL'];
        }
        parent::imprimirRelatorioEspecial();
    }


    /**
     * Uma classe de relatórios deve sempre montar seu select
     *
     */
    public function montaSELECT($_campos,$_where=null,
                                $_orderby=null,$_limit=null,$_extras=null) {
        $_custo = ($_POST['CUSTO']=='CM'
                            ? 'PRODUTO_CUSTOMEDIO'
                            : 'PRODUTO_CUSTOATUAL');
        $_strSQL= "SELECT " .
                  "PRODUTO_CODIGO, PRODUTO_DESC," .
                  "PRODUTO_ESTOQUE,{$_custo} AS CUSTO," .
                  "UNIDADE_CODIGO," .
                  "(PRODUTO_ESTOQUE*{$_custo}) AS VALOR_TOTAL " .
                  "FROM {$this->_nome_tabela} " .
                  "WHERE PRODUTO_ESTOQUE>0 " .
                  "ORDER BY (PRODUTO_ESTOQUE*{$_custo}) DESC";
```

```
        return $_strSQL;
    }

    /**
     * Devemos pré-processar o relatório antes de ser gerado o html
     *
     */
    public function processaRelatorio() {
        // Vamos acumular os valores
        $this->getCampo("POSIC")->setValor(++$this->_ordem);
        $_perc = ($this->getCampo("VALOR_TOTAL")->getValor()/$this->_total)*100;
        $this->getCampo("PERC")->setValor($_perc);
        if($this->_perc<=80) {
            $_curva = "A";
            if(($this->_perc+$_perc)>80) {
                if(($this->_perc+$_perc-80)-(80-$this->_perc)>0) {
                    $_curva = "B";
                }
            }
        } elseif($this->_perc<=95) {
            $_curva = "B";
            if(($this->_perc+$_perc)>95) {
                if(($this->_perc+$_perc-95)-(95-$this->_perc)>0) {
                    $_curva = "C";
                }
            }
        } else {
            $_curva = "C";
        }
        $this->getCampo("CURVA")->setValor($_curva);
        $this->_perc += $_perc;
        $this->getCampo("ACUM")->setValor($this->_perc);
    }
}
?>
```

Pronto, com a classe construída precisamos apenas criar o programa de geração do relatório e incluí-lo no menu do sistema.

Lista 20: rel_curvaabc.php5

```
<?php
/**
 * Relatório: Curva ABC de Estoque
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_curvaabc.inc",
              "nome"=>"curvaabc");
$_FTitulo = "Relatório Curva ABC de Estoque";
return include_once("../instanciaclasse.php5");
?>
```

A Figura 8.19 exibe o formulário de definição do tipo de custo que será utilizado na geração do relatório.

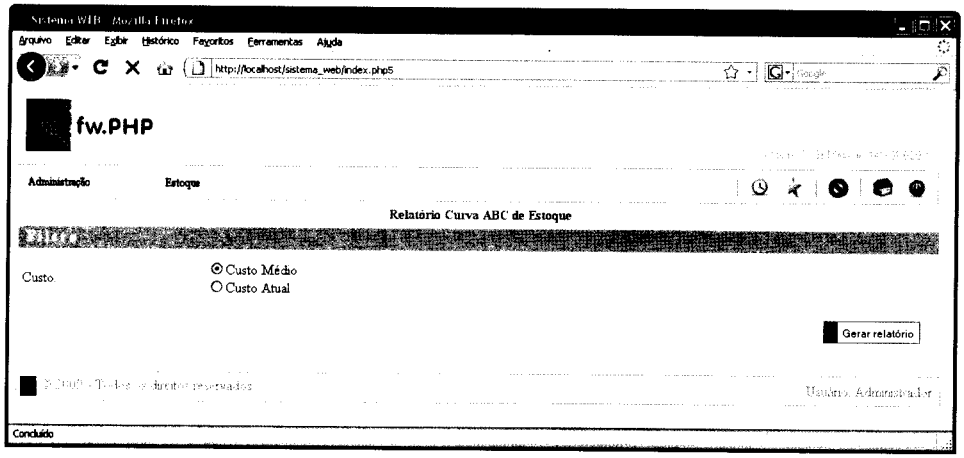
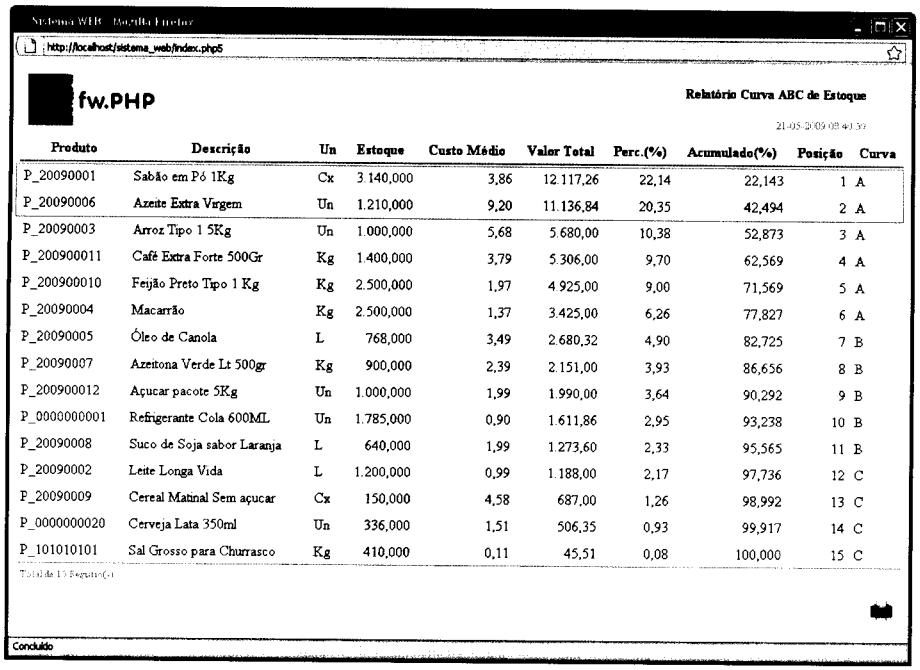


Figura 8.19

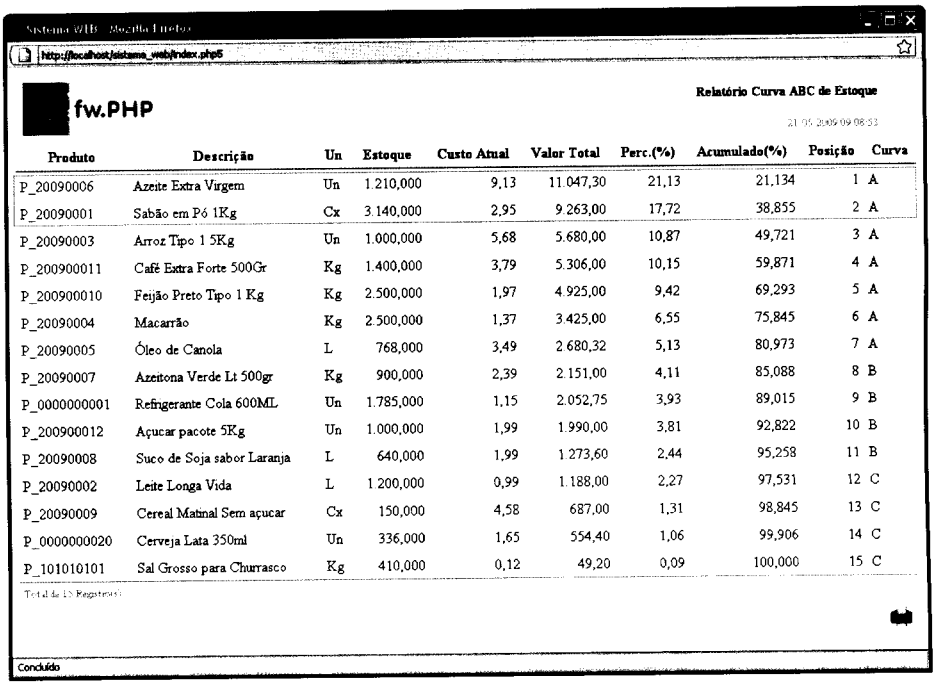
A escolha do tipo de custo pode alterar a classificação da lista, porque o custo médio reflete um custo histórico do produto, absorvendo as variações ao longo de sua vida no estoque e o custo atual é uma fotografia do momento, pois representa o último custo de entrada registrado para o produto.

A diferença entre esses dois custos é mostrada nas Figuras 8.20 e 8.21. Note a troca de posição de alguns itens.



Produto	Descrição	Un	Estoque	Custo Médio	Valor Total	Perc.(%)	Acumulado(%)	Posição	Curva
P_20090001	Sabão em Pó 1Kg	Cx	3.140,000	3,86	12.117,26	22,14	22,143	1	A
P_20090006	Azeite Extra Virgem	Un	1.210,000	9,20	11.136,84	20,35	42,494	2	A
P_20090003	Arroz Tipo 1 5Kg	Un	1.000,000	5,68	5.680,00	10,38	52,873	3	A
P_200900011	Café Extra Forte 500Gr	Kg	1.400,000	3,79	5.306,00	9,70	62,569	4	A
P_200900010	Feijão Preto Tipo 1 Kg	Kg	2.500,000	1,97	4.925,00	9,00	71,569	5	A
P_20090004	Macarrão	Kg	2.500,000	1,37	3.425,00	6,26	77,827	6	A
P_20090005	Óleo de Canola	L	768,000	3,49	2.680,32	4,90	82,725	7	B
P_20090007	Azeitona Verde Lt 500gr	Kg	900,000	2,39	2.151,00	3,93	86,656	8	B
P_200900012	Açucar pacote 5Kg	Un	1.000,000	1,99	1.990,00	3,64	90,292	9	B
P_0000000001	Refrigerante Cola 600ML	Un	1.785,000	0,90	1.611,86	2,95	93,238	10	B
P_20090008	Suco de Soja sabor Laranja	L	640,000	1,99	1.273,60	2,33	95,565	11	B
P_20090002	Leite Longa Vida	L	1.200,000	0,99	1.188,00	2,17	97,736	12	C
P_20090009	Cereal Matinal Sem açucar	Cx	150,000	4,58	687,00	1,26	98,992	13	C
P_0000000020	Cerveja Lata 350ml	Un	336,000	1,51	506,35	0,93	99,917	14	C
P_101010101	Sal Grosso para Churrasco	Kg	410,000	0,11	45,51	0,08	100,000	15	C

Figura 8.20



fw.PHP

Relatório Curva ABC de Estoque

21/05/2009 09:08:53

Produto	Descrição	Un	Estoque	Custo Atual	Valor Total	Perc.(%)	Acumulado(%)	Posição	Curva
P_20090006	Azeite Extra Virgem	Un	1.210,000	9,13	11.047,30	21,13	21,134	1	A
P_20090001	Sabão em Pó 1Kg	Cx	3.140,000	2,95	9.263,00	17,72	38,855	2	A
P_20090003	Arroz Tipo 1 5Kg	Un	1.000,000	5,68	5.680,00	10,87	49,721	3	A
P_200900011	Café Extra Forte 500Gr	Kg	1.400,000	3,79	5.306,00	10,15	59,871	4	A
P_200900010	Feijão Preto Tipo 1 Kg	Kg	2.500,000	1,97	4.925,00	9,42	69,293	5	A
P_20090004	Macarrão	Kg	2.500,000	1,37	3.425,00	6,55	75,845	6	A
P_20090005	Óleo de Canola	L	768,000	3,49	2.680,32	5,13	80,973	7	A
P_20090007	Azeitona Verde Lt 500gr	Kg	900,000	2,39	2.151,00	4,11	85,088	8	B
P_0000000001	Refrigerante Cola 600ML	Un	1.785,000	1,15	2.052,75	3,93	89,015	9	B
P_200900012	Açucar pacote 5Kg	Un	1.000,000	1,99	1.990,00	3,81	92,822	10	B
P_20090008	Suco de Soja sabor Laranja	L	640,000	1,99	1.273,60	2,44	95,258	11	B
P_20090002	Leite Longa Vida	L	1.200,000	0,99	1.188,00	2,27	97,531	12	C
P_20090009	Cereal Matinal Sem açucar	Cx	150,000	4,58	687,00	1,31	98,845	13	C
P_0000000020	Cerveja Lata 350ml	Un	336,000	1,65	554,40	1,06	99,906	14	C
P_101010101	Sal Grosso para Churrasco	Kg	410,000	0,12	49,20	0,09	100,000	15	C

Total de 15 Registros

Concluído

Figura 8.21

Anotações

Capítulo 9

Inventário

A premissa de qualquer sistema de controle de estoque é que o estoque virtual esteja sempre igual ao estoque físico, porém é fato que existem vários acontecimentos que, via de regra, não são detectados pelos gestores do estoque. Entre esses acontecimentos podemos ter:

- Furto
- Erros operacionais
- Quebras não identificadas
- Digitação de dados incorretos
- Falta de apontamento de movimentos de estoque

Estes problemas e outros relacionados ao não-lançamento dos movimentos (ou o lançamento incorreto) fazem com que o estoque físico fique diferente do estoque registrado no sistema.

A falta de confiabilidade ou de acuracidade dos estoques afeta toda a empresa, pois com essa informação equivocada podemos ter problemas no planejamento de compra de produtos, atrasos na produção e principalmente a falta de estoque para venda aos clientes finais.

A correção deste tipo de situação deve ser realizada da seguinte forma:

- Inventário
- Diagnóstico das causas
- Implementação de medidas preventivas

O inventário tem a função de tornar os estoques virtuais e físicos idênticos, eliminando as distorções existentes. Esta é uma medida corretiva, ou seja, se nada for feito para resolver os problemas existentes, teremos novas divergências no futuro. Mas, você pode estar se perguntando, não é natural a existência do inventário? Sim, devemos realizar um inventário de tempos em tempos. Mas não é de forma nenhuma natural que sejam encontradas grandes divergências, mais de 10% dos itens com divergência ou itens com mais de 15% isoladamente. O problema com as divergências é que acabamos

perdendo a confiança no sistema, o que nos leva a construir ferramentas extra-oficiais para corrigir os problemas existentes.

Para evitar que o estoque torne-se pouco confiável, devemos descobrir as causas das diferenças (humanas, processos, tecnológicos) e implementar rotinas para eliminar ou pelo menos reduzir a ocorrência desses erros.

Não vamos tratar deste assunto (diagnóstico de causas e implementação de medidas corretivas), mas do módulo de inventário, construindo as ferramentas necessárias para sua efetiva execução.

O inventário é um processo de contagem física dos estoques, seja de um produto ou de toda a empresa. Os produtos devem ser contados por local de armazenamento, pois só assim teremos informações suficientes para detectar erros no processo, tanto do estoque como um todo quanto dos estoques localizados (podemos ter o produto com o estoque correto, mas as distribuições locais com erros, alguns com falta outros com sobra).

O processo de inventário (somente a parte sistêmica) divide-se nas seguintes etapas:

- Cadastro do inventário
- Inclusão dos itens que serão inventariados
- Congelamento do inventário
- Geração das fichas de inventário
- Entrada da contagem
- Análise do inventário
- Recontagem (implica em voltar ao processo de geração das fichas)
- Consolidação do inventário

Cadastro de inventário

A rotina de cadastro de inventário é bem simples. Não há nada diferente do que já foi mostrado nos capítulos anteriores. É preciso criar a classe e disponibilizar os campos que serão tratados na inclusão do inventário, sua alteração e, quando possível, sua exclusão.

Os seguintes campos devem ser definidos na classe:

1. INVENTARIO_CODIGO

   Instância da classe **string**, contém o código do inventário. Esse campo pode ter seu valor atribuído pelo usuário (limitado a dez caracteres) ou ter o valor definido de forma automática pelo sistema, neste caso considera-se que o valor será sempre numérico.

2. INVENTARIO_DESC

   Instância da classe **string**, contém a descrição do inventário.

3. INVENTARIO_DATA_PREVISTA

   Instância da classe **data**, esse campo contém a data prevista de realização do inventário, sendo apenas um campo informativo.

4. INVENTARIO_DATA_REAL

   Instância da classe **data**, contém a data real de execução do inventário, mais precisamente a data em que o inventário foi consolidado no sistema. Esse campo não é exibido no formulário de manutenção do inventário.

5. INVENTARIO_CONGELADO

   Instância da classe **string**, contém o indicador de que o inventário foi congelado ou não, sendo o valor-padrão 'N'. Esse campo não é exibido no formulário de manutenção, tendo seu conteúdo alterado no processo de congelamento de inventário.

6. INVENTARIO_CONTAGEM

   Instância da classe **inteiro**, contém o número da contagem atual do inventário. O valor inicial do campo é 1 (um), o qual é alterado sempre que for necessária uma recontagem do estoque. Esse campo não é exibido no formulário de manutenção. O processo de análise de inventário altera o seu valor sempre que o usuário optar pela recontagem.

7. INVENTARIO_CONSOLIDADO

   Instância da classe **string**, indica se o inventário já foi consolidado. Ele não é exibido no formulário de manutenção, sendo o seu valor-padrão igual a 'N' e o seu conteúdo é alterado pelo processo de análise de inventário quando o usuário optar por consolidar o inventário.

8. LOCAL_CODIGO

   Instância da classe **string**, contém o código do local que será inventariado ou a constante 'TODOS', indicando que o inventário trata todos os locais existentes.

Neste momento, vamos criar a classe **inventario** apenas com os métodos básicos necessário para o cadastro do inventário. No restante deste capítulo incluiremos os métodos complementares.

Lista 1: classe_inventario.inc

```
<?php
/**
 * Classe inventario
 */
class inventario extends base {
     public function __construct(BancoDados $_conn) {
```

```
        parent::__construct($_conn);
        $this->_nome_tabela = 'inventario';
        $this->_class_path = 'estoque';
        $this->addCampo(new string("INVENTARIO_CODIGO","Código do Inventário",
                10,null,null,true,false,true,1,null,null,true,$_conn));
        $this->addCampo(new string("INVENTARIO_DESC","Descrição",
                30,null,null,true,true,true,5,null,null,false,$_conn));
        $this->addCampo(new data("INVENTARIO_DATA_PREVISTA","Data Prevista",
                10,null,null,true,true,true,'NOW',null,null,false,$_conn));
        $this->addCampo(new data("INVENTARIO_DATA_REAL","Data Realizado",
                10,null,null,false,false,true,'NOW',null,null,false,$_conn));
        $this->addCampo(new string("INVENTARIO_CONGELADO","Congelado",
                1,null,null,false,false,true,null,null,null,false,$_conn));
        $this->addCampo(new inteiro("INVENTARIO_CONTAGEM","Contagem atual",
                1,null,null,false,false,false,1,null,null,false));
        $this->addCampo(new string("INVENTARIO_CONSOLIDADO","Fechado",
                1,null,null,false,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("LOCAL_CODIGO","Local",
                10,null,null,true,false,true,1,null,null,false,$_conn));
        $this->addCampo(new string("USUARIO_LOGIN","Usuário",
                20,null,null,false,false,false,null,null,null,false,$_conn));
        $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",
                14,null,null,false,false,false,null,null,null,false,$_conn));
        $this->getCampo("INVENTARIO_CODIGO")->setComportamento_form('proximo');
        $this->getCampo("LOCAL_CODIGO")->setComportamento_form('select');
    }

    /**
     * Retorna o comando SQL necessário para
     * a busca do próximo código
     *
     * @return string
     */
    public function montaSQLProximoCodigo() {
        return "SELECT MAX(INVENTARIO_CODIGO) AS CODIGO " .
               "FROM {$this->_nome_tabela}";
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function incluir() {
        $this->setUsuarioeData();
        return parent::incluir();
    }

    /**
     * Ajustar os campos Usuário e data da
     * última alteração antes de efetivar a transação
     *
     * @return mixed
     */
    public function Alterar() {
        $this->setUsuarioeData();
        return parent::alterar();
    }

}
?>
```

O programa para executar a manutenção do inventário é, pelo menos por enquanto, tão simples como qualquer outro programa já descrito anteriormente.

Para separar o módulo de inventário, todos os programas terão como sufixo a constante 'inv_'. Logo, o cadastro do inventário será *inv_cadastroinventario.php5*.

Lista 2: inv_cadastroinventario.php5

```
<?php
/**
 * Cadastro de Inventário
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_inventario.inc",
              "nome"=>"inventario");
$_FTitulo = "Cadastro de Inventários";
return include_once("../instanciaclasse.php5");
?>
```

Este programa será alterado no próximo item para lançamento dos itens do inventário.

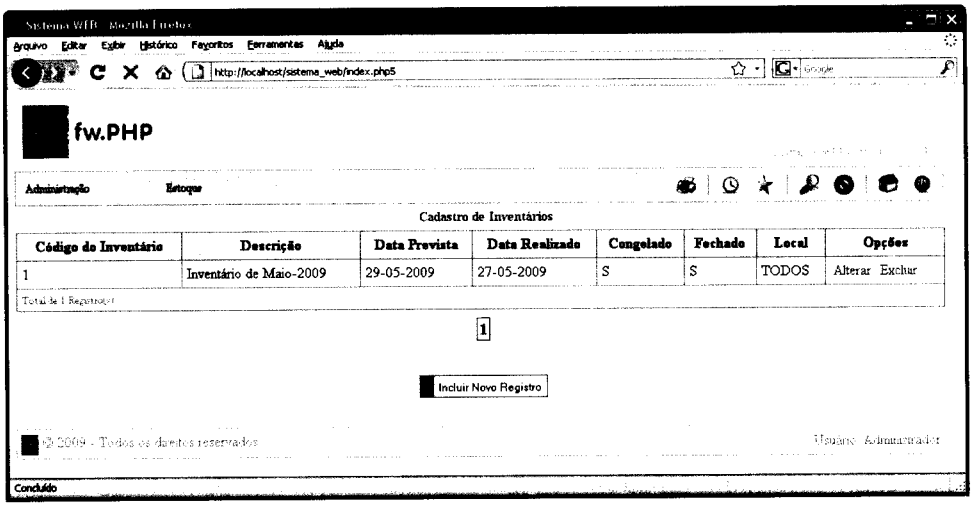


Figura 9.1

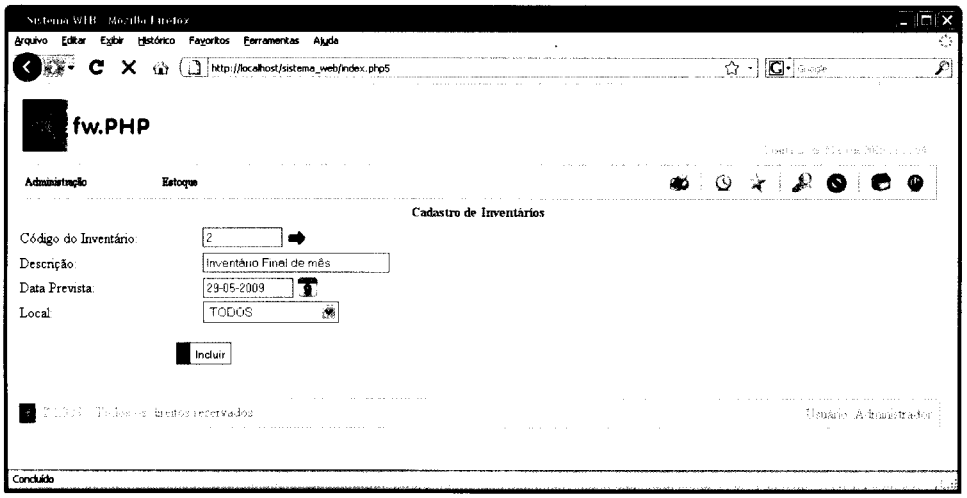


Figura 9.2

Itens do inventário

Uma vez que o cadastro do inventário foi realizado, podemos incluir os itens que farão parte do inventário. Os itens devem ser lançados um a um, conforme a necessidade do usuário (uma facilidade que podemos incluir futuramente no sistema é a inclusão de todos os itens de um departamento ou de um grupo de produtos).

É importante notar que o lançamento de itens está vinculado a um inventário específico, logo precisamos de ferramentas para informar ao programa o inventário que está sendo gerenciado.

Os seguintes campos devem ser definidos na classe de itens de inventário:

1. INVENTARIO_CODIGO

   Instância da classe **string**, contém o código do inventário. Esse campo é enviado pela classe **inventario** no momento da chamada ao processo de inclusão de itens.

2. PRODUTO_CODIGO

   Instância da classe **string**, contém o código do produto que será inventariado. Esse campo terá o atributo 'comportamento form' alterado para 'ajax'.

3. PRODUTO_DESC

   Instância da classe **string**, contém o descritivo do produto, sendo apenas um campo informativo, utilizado na exibição da lista de itens do inventário e na geração do relatório.

4. LOCAL_CODIGO

   Instância da classe **string**, contém o código do local a ser inventariado. Seu preenchimento é automático, a classe gera os registros necessários para inventariar todos os locais definidos no inventário (que pode ser todos os locais ou um local específico). Esde campo não é exibido no formulário de manutenção.

5. LOCAL_DESC

   Instância da classe **string**, contém o descritivo do local, sendo apenas um campo informativo, utilizado na exibição da lista de itens do inventário e na geração do relatório. Esse campo não é exibido no formulário de manutenção.

6. PRODUTO_ESTOQUE

   Instância da classe **float**, contém o estoque do produto e o local no momento do congelamento do estoque. Esse campo será preenchido no processo de congelamento do inventário, não sendo exibido no formulário de manutenção. Ele será comparado no momento da análise do inventário e da geração do movimento de ajuste.

7. PRODUTO_CUSTOMEDIO

   Instância da classe **dinheiro**, contém o custo médio do produto e o local. Esse campo terá seu conteúdo definido no momento do congelamento do estoque e será utilizado no processo de ajuste de estoque para a valorização do movimento. Ele não será exibido no formulário de manutenção.

8. INVENTARIO_ESTOQUE

   Instância da classe **float**, contém o estoque inventariado do produto, quer seja da contagem considerada válida ou da última contagem realizada (quando a consolidação for executada no modo forçado). Seu valor será preenchido no processo de análise de inventário, tanto no modo de recontagem quanto no modo de consolidação do inventário. Esse campo não será exibido no formulário de manutenção.

Requisitos

1. Os inventários congelados ou consolidados (INVENTARIO_CONGELADO='S' ou INVENTARIO_CONSOLIDADO='S') não podem ter os itens modificados, ou seja, não devemos permitir a inclusão de novos itens, alteração de itens ou mesmo a exclusão. Esta regra deve ser implementada no método *listar()*, o que indica que precisamos sobrescrever o método original da classe **base**.
2. Um item do inventário só pode ser excluído, uma vez que não faz *sentido sua* alteração (teríamos somente o produto para alterar, mas esse campo é chave primária da tabela).
3. Deve-se gerar um registro na tabela **inventarioproduto** para cada relação produto e local.

4. A exclusão de um produto e local implica na exclusão de todos os registros relacionados ao produto.

   Assim como foi mostrado no item anterior, mostraremos agora a construção básica da classe **inventarioproduto**, deixando para os próximos tópicos a construção dos demais métodos necessários ao correto funcionamento do módulo.

Lista 3: classe_inventarioproduto.inc

```
<?php
/**
 * Classe itens do inventario
 */
class inventarioproduto extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'inventarioproduto';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("INVENTARIO_CODIGO","Inventário",
                    10,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("PRODUTO_CODIGO","Produto",
                    20,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("PRODUTO_DESC","Descrição produto",
                    60,null,null,false,false,true,null,null,null,false,$_conn));
          $this->addCampo(new string("LOCAL_CODIGO","Local",
                    10,null,null,false,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("LOCAL_DESC","Descrição local",
                    30,null,null,false,false,true,null,null,null,false,$_conn));
          $this->addCampo(new float("PRODUTO_ESTOQUE","Estoque Atual",
                    12,null,null,false,true,true,0.001,null,null,false));
          $this->addCampo(new dinheiro("PRODUTO_CUSTOMEDIO","Custo médio",
                    12,null,null,false,true,false,0,null,null,false));
          $this->addCampo(new float("INVENTARIO_ESTOQUE","Inventariado",
                    12,null,null,false,false,true,0.001,null,null,false));
          $this->addCampo(new string("USUARIO_LOGIN","Usuário",
                    20,null,null,false,false,false,null,null,null,false,$_conn));
          $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",
                    14,null,null,false,false,false,null,null,null,false,$_conn));
          $this->addCampo(new string("ITENS","",
                    1,null,null,true,false,false,null,null,null,false,$_conn));
          $this->addCampo(new string("INVENTARIO","",
                    10,null,null,true,false,false,null,null,null,false,$_conn));
          $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('ajax');
          $this->getCampo("PRODUTO_CODIGO")->setClasse("produto");
          $this->getCampo("PRODUTO_CODIGO")->setArquivo_Classe(
                    "estoque/classes/classe_produto.inc");
          $this->getCampo("INVENTARIO")->setComportamento_form('hidden');
          $this->getCampo("ITENS")->setComportamento_form('hidden');
          $this->getCampo("ITENS")->setValor_Fixo('S');
     }


     /**
      * Apenas os itens do inventário selecionado
      *
      * @return string
      */
     public function FiltroFixo() {
          return "(INVENTARIO_CODIGO='{$_GET['INVENTARIO_CODIGO']}') ";
     }
```

```
/**
 * Ajusta o comportamento dos campos do formulário
 *
 */
protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
     if(stripos($_metodo,'getformulario')!==false) {
          switch($_nome) {
               case 'INVENTARIO_CODIGO':
                              $_campo->addAtributo(new atributo("DISABLED"));
                              break;
          }
     }
}

/**
 * Reescrito
 *
 */
public function montaSELECT($_campos,$_where=null,
                    $_orderby=null,$_limit=null,$_extras=null) {
     if(func_num_args()==6&&
       (func_get_arg(5)=='listar'||func_get_arg(5)=='relatorio')) {
          return "SELECT INVENTARIO_CODIGO,I.PRODUTO_CODIGO,P.PRODUTO_DESC," .
            "I.LOCAL_CODIGO,L.LOCAL_DESC," .
            "I.PRODUTO_ESTOQUE,I.PRODUTO_CUSTOMEDIO,I.INVENTARIO_ESTOQUE " .
            "FROM {$this->_nome_tabela} I " .
            "LEFT JOIN PRODUTO P ON I.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
            "LEFT JOIN LOCALESTOQUE L ON I.LOCAL_CODIGO=L.LOCAL_CODIGO " .
            ($_where!==null ? "WHERE {$_where}" : "") .
            ($_orderby!==null ? " ORDER BY {$_orderby}" : "") .
            ($_extras!==null ? " {$_extras}" : "") .
            ($_limit!==null ? " LIMIT {$_limit}" : "");
     } else {
          return parent::montaSELECT($_campos,$_where,
                                     $_orderby,$_limit,$_extras);
     }
}


/**
 * Conforme o parâmetro de geração do inventário
 * deve-se gerar o mesmo produto para todos os locais
 * definidos no gerador do inventário
 * Ajustar os campos Usuário e data da
 * última alteração antes de efetivar a transação
 *
 * @return mixed
 */
public function incluir() {
     $this->getCampo("INVENTARIO_CODIGO")->setValor($_POST['INVENTARIO']);
     $this->getCampo("INVENTARIO_CODIGO")->setIncluir(true);
     $this->getCampo("LOCAL_CODIGO")->setIncluir(true);
     $this->getCampo("INVENTARIO")->setIncluir(false);
     $this->getCampo("ITENS")->setIncluir(false);
     $_GET['INVENTARIO_CODIGO'] =
                              $this->getCampo("INVENTARIO_CODIGO")->getValor();
     $_GET['ITENS'] = 'S';
     $_POST['SCRIPT_NAME'] .=
                    "?ITENS=S&INVENTARIO_CODIGO={$_GET['INVENTARIO_CODIGO']}";
     $this->setUsuarioeData();
     $_strSQL = "SELECT L.LOCAL_CODIGO FROM LOCALESTOQUE L " .
                "LEFT JOIN INVENTARIO I " .
                "ON (I.LOCAL_CODIGO='TODOS' OR " .
                "L.LOCAL_CODIGO=I.LOCAL_CODIGO) " .
```

```
                "WHERE I.INVENTARIO_CODIGO='{$_POST['INVENTARIO']}'";
        $_locais = null;
        if($this->_conn->executaSQL($_strSQL)!==false&&
           $this->_conn->getNumRows()>0) {
             while($_dados=$this->_conn->proximo()) {
                  $_locais[] = $_dados['LOCAL_CODIGO'];
             }
        }
        if($_locais===null) {
             return false;
        } else {
             $this->_conn->startTransaction();
             foreach($_locais as $_local) {
                  $this->getCampo("LOCAL_CODIGO")->setValor($_local);
                  if(($_res=parent::incluir())===false) {
                       $this->_conn->ROLLBACK();
                       return $_res;
                  }
             }
             $this->_conn->commit();
             return $_res;
        }
   }

   /**
    * Devemos excluir todos os produtos do mesmo inventário
    *
    */
   public function Excluir() {
        $this->getCampo("LOCAL_CODIGO")->setPk(false);
        $_GET['INVENTARIO_CODIGO'] =
                             $this->getCampo("INVENTARIO_CODIGO")->getValor();
        $_GET['ITENS'] = 'S';
        $_POST['SCRIPT_NAME'] .=
                        "?ITENS=S&INVENTARIO_CODIGO={$_GET['INVENTARIO_CODIGO']}";
        return parent::excluir();
   }

   public function getFormulario($_funcao='INC') {
        $this->getCampo("INVENTARIO_CODIGO")->setValor(
                                                  $_GET['INVENTARIO_CODIGO']);
        $this->getCampo("INVENTARIO")->setValor_Fixo($_GET['INVENTARIO_CODIGO']);
        return parent::getFormulario($_funcao);
   }

   /**
    * Gera as opções na lista de registros
    * Na forma padrão, gera os links para Alterar e Excluir
    *
    * @param string $_pk
    * @param tag $_tab
    */
   protected function getOpcoesLista($_pk,tag & $_tab,tipospadrao $_tipos) {
        if($this->_exibe_opcoes===true) {
             $_lnkexc = new tag($_tipos->getTipo("A"),
                            Array(new atributo("HREF","javascript:void(0);"),
                                 new atributo("ONCLICK",
                                 "ObjProcAjax.run('{$_SERVER['PHP_SELF']}" .
                                 "?ITENS=S&ACAO=EXC{$_pk}','CORPO');")),
                            " Excluir ");
             $_tab->getLastSubTag()->addSubTag(new tag($_tipos->getTipo('TD'),
                       Array(new atributo("STYLE","border:1px solid #a0a0a0;")),
                       $_lnkexc->toHTML()));
        }
```

```
    }

    /**
     * Insere o botão de inclusão na lista de registros
     *
     * @param tag $_div
     * @param tipospadrao $_tipos
     */
    protected function getBotaoIncluir(tag & $_div,tipospadrao $_tipos) {
        if($this->_exibe_opcoes===true||$this->_exibe_botao_incluir===true) {
            $_div->addSubTag(new tag($_tipos->getTipo("BR")));
            $_div->addSubTag(new tag($_tipos->getTipo("P"),
                        Array(new atributo("ALIGN","CENTER"))));
            $_div->getLastSubTag()->addSubTag(
                new tag($_tipos->getTipo('BUTTON'),
                    Array(new atributo("ONCLICK",
                        "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?ITENS=S&" .
                        "ACAO=INC&INVENTARIO_CODIGO=" .
                        "{$_GET['INVENTARIO_CODIGO']}'," .
                        "'CORPO');")),
                    " Incluir Novo Registro "));
        }
    }

    /**
     * Se o Inventário estiver congelado ou consolidado
     * não será possível incluir ou excluir itens
     */
    public function listar() {
        $_inv = new inventario($this->_conn);
        if($_inv->inventarioliberado($_GET['INVENTARIO_CODIGO'])===false) {
            $this->_exibe_opcoes = false;
            $this->_exibe_botao_incluir = false;
        }
        return parent::listar();
    }
}
?>
```

Para utilizar essa classe precisamos alterar a classe **inventario** para que seja gerado um link para exibição dos itens do inventário, uma vez que os itens sempre estão relacionados a um inventário específico.

Precisamos ainda alterar o programa *inv_cadastroinventario.php5* para que ele trate tanto do cadastro de inventário quanto do cadastro de itens do inventário. Essa adaptação é necessária para contornar a questão de permissão do sistema, pois para acessar qualquer programa, ele deve estar na lista de programas autorizados ao usuário e no seu menu de acesso.

Como não vamos inserir o programa de itens no menu, uma vez que ele sempre depende do inventário, precisamos de uma forma para que a classe seja instanciada normalmente. Temos duas opções, a primeira é alterar o framework (que é um pouco complicado) ou então realizar a adaptação proposta (que provavelmente é a melhor opção).

O primeiro passo é alterar a classe **inventario**. Vamos modificar o método *getOpcoesLista()* e incluir o link nos itens do inventário.

```
/**
 * Gera as opções na lista de registros
 * Na forma padrão, gera os links para Alterar e Excluir
 *
 * @param string $_pk
 * @param tag $_tab
 */
protected function getOpcoesLista($_pk,tag & $_tab,tipospadrao $_tipos) {
    $_lnkitens = new tag($_tipos->getTipo("A"),
                         Array(new atributo("HREF","javascript:void(0);"),
                            new atributo("ONCLICK",
                                  "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?" .
                                  "ITENS=S{$_pk}','CORPO');")),
                         " Itens ");
    $_lnkalt = new tag($_tipos->getTipo("A"),
                    Array(new atributo("HREF","javascript:void(0);"),
                       new atributo("ONCLICK",
                            "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?" .
                            "ACAO=ALT{$_pk}','CORPO');")),
                         " Alterar ");
        $_lnkexc = new tag($_tipos->getTipo("A"),
                         Array(new atributo("HREF","javascript:void(0);"),
                            new atributo("ONCLICK",
                              "ObjProcAjax.run('{$_SERVER['PHP_SELF']}?" .
                              "ACAO=EXC{$_pk}','CORPO');")),
                         " Excluir ");
        $_tab->getLastSubTag()->addSubTag(new tag($_tipos->getTipo('TD'),
                         Array(new atributo("STYLE","border:1px solid #a0a0a0;")),
                         $_lnkitens->toHTML() . $_lnkalt->toHTML() .
                         $_lnkexc->toHTML()));
    }
```

Com isso teremos um novo formato para a página de manutenção de inventário.

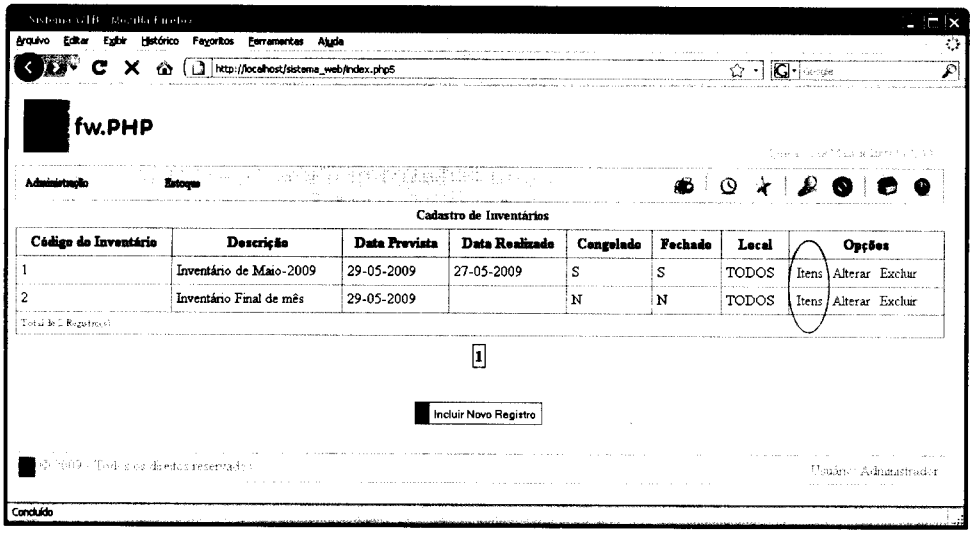


Figura 9.3

O próximo passo é alterar o programa *inv_cadastroinventario.php5*, no qual vamos tratar condicionalmente a definição da classe e do título da página. A variável de controle será $_GET['ITENS'] ou $_POST['ITENS']. Caso qualquer uma contenha o valor 'S', o programa entende que se trata da manutenção de itens.

Lista 4: inv_cadastroinventario.php5

```
<?php
/**
 * Cadastro de Inventário
 *
 */
if($_GET['ITENS']=='S'||$_POST['ITENS']=='S') {
     $_classe = Array(
                    "arquivo"=>"estoque/classes/classe_inventarioproduto.inc",
                    "nome"=>"inventarioproduto");
     $_FTitulo = "Cadastro de Inventário - Itens";
} else {
     $_classe = Array(
                    "arquivo"=>"estoque/classes/classe_inventario.inc",
                    "nome"=>"inventario");
     $_FTitulo = "Cadastro de Inventários";
}
return include_once("../instanciaclasse.php5");
?>
```

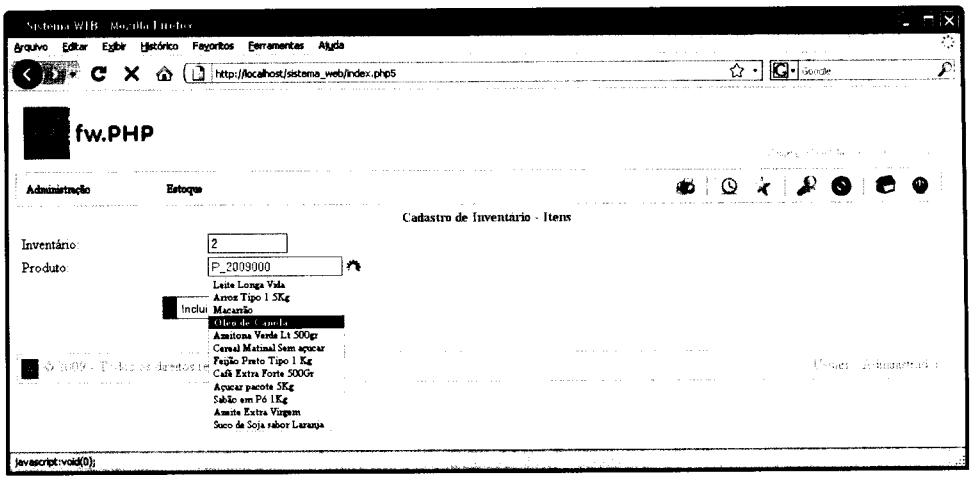


Figura 9.4

Note que o sistema gerou duas entradas na tabela de itens de inventário (inventarioproduto), uma vez que temos dois locais cadastrados no sistema e o inventário foi gerado para todos os locais.

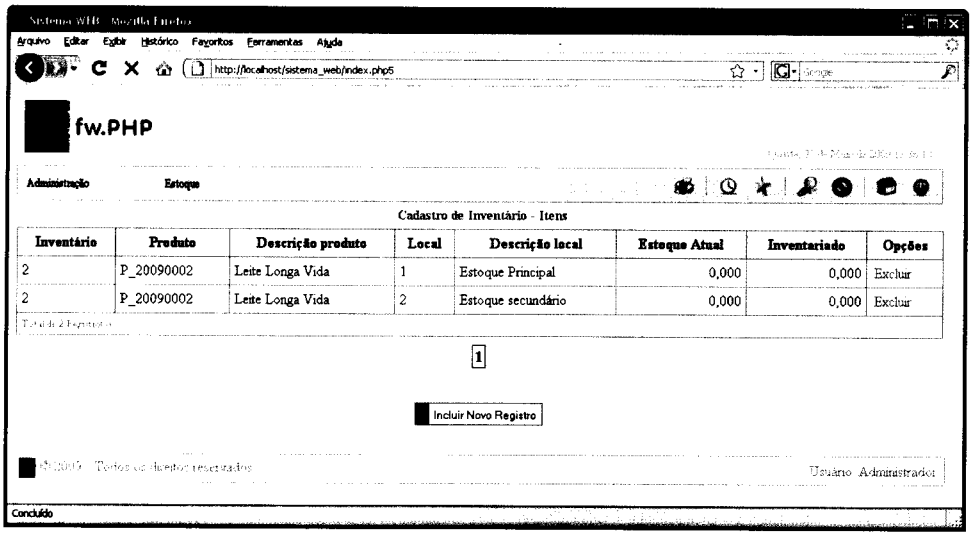


Figura 9.5

Congelamento

Uma vez que o inventário e seus itens foram cadastrados, devemos, antes de realizar a contagem do estoque, executar o processo de congelamento. O processo grava os estoques dos produtos e locais participantes do inventário, bem como o custo médio relacionado. Sem a execução desse processo não é possível ir adiante com o inventário, ou seja, não há como emitir as fichas de contagem nem efetuar a entrada das contagens ou analisar o inventário.

O processo de congelamento deve ser irreversível e essa informação deve ser passada ao usuário durante a execução do processo.

A classe é bem simples, precisamos apenas de um campo definido, que será utilizado para a seleção do inventário a ser congelado, e de dois métodos de processamento. O primeiro método é somente para uma exibição especializada do formulário de filtragem. Vamos sobrescrever o método *processaAcao()*, o qual gera o formulário de seleção e inclui a biblioteca javascript específica do congelamento.

O segundo método que precisamos desenvolver será responsável pelo processo de congelamento, ou seja, pela geração do estoque e do custo médio de cada produto e local. O método deve instanciar as classes **inventario** e **inventarioproduto**, em ambas as classes Serpa executado o método *congelar()*.

Esse método deve ser desenvolvido para cada classe. Na classe **inventario** o método executa a alteração do campo INVENTARIO_CONGELADO para 'S' e na classe **inventarioproduto** o método grava as informações de custo médio e estoque atual contidas na tabela estoquelocal.

inventario

```
/**
 * Executa a operação de congelamento de inventário
 *
 * @param string $_inv
 * @return mixed
 */
public function congelar($_inv) {
     $this->getCampo("INVENTARIO_DESC")->setAlterar(false);
     $this->getCampo("INVENTARIO_DATA_PREVISTA")->setAlterar(false);
     $this->getCampo("INVENTARIO_DATA_REAL")->setAlterar(false);
     $this->getCampo("INVENTARIO_CONGELADO")->setAlterar(true);
     $this->getCampo("INVENTARIO_CONGELADO")->setValor('S');
     $this->getCampo("INVENTARIO_CODIGO")->setValor($_inv);
     return $this->alterar();
}
```

inventarioproduto

```
/**
 * Executa o congelamento do inventário selecionado
 * 1 - atualiza a tabela de inventarioproduto com o estoque e custo médio
 *
 * @param string $_inv
 * @return mixed
 */
public function congelar($_inv) {
     $_strSQL = "SELECT PRODUTO_CODIGO,LOCAL_CODIGO," .
                "PRODUTO_ESTOQUE,PRODUTO_CUSTOMEDIO FROM " .
                "ESTOQUELOCAL WHERE PRODUTO_CODIGO " .
                "IN(SELECT PRODUTO_CODIGO FROM {$this->_nome_tabela} " .
                "WHERE INVENTARIO_CODIGO='{$_inv}')";
     if($this->_conn->executaSQL($_strSQL)!==false&&
        $this->_conn->getNumRows()>0) {
          while(($_dados=$this->_conn->proximo())!==false) {
               $_produtos[$_dados['PRODUTO_CODIGO']][$_dados['LOCAL_CODIGO']] =
                         Array($_dados["PRODUTO_ESTOQUE"],
                              $_dados["PRODUTO_CUSTOMEDIO"]);
          }
          $this->getCampo("INVENTARIO_CODIGO")->setValor($_inv);
          foreach($_produtos as $_prd=>$_local) {
               $this->getCampo("PRODUTO_CODIGO")->setValor($_prd);
               foreach($_local as $_lcl=>$_info) {
                    $this->getCampo("LOCAL_CODIGO")->setValor($_lcl);
                    $this->getCampo("PRODUTO_ESTOQUE")->setValor(
                                        number_format($_info[0],3,",",""));
                    $this->getCampo("PRODUTO_CUSTOMEDIO")->setValor(
                                        number_format($_info[1],2,",",""));
                    $this->setUsuarioeData();
                    if($this->alterar()===false) {
                         return false;
                    }
               }
          }
          return true;
     } else {
          return false;
     }
}
```

A classe javascript de congelamento deve realizar apenas duas tarefas. Primeiramente deve mostrar uma mensagem informando que o processo é irreversível, em segundo lugar deve disponibilizar o método para execução do congelamento.

Lista 5: congelamento.js

```
// Rotinas de congelamento
function congelar() {};
congelar.prototype = {
run: function() {
              var inv = document.getElementById('INVENTARIO_CODIGO');
              var cod = inv.options[inv.selectedIndex].value;
              var lbl = inv.options[inv.selectedIndex].text;
              var txt = '<span style="color:red;">Atenção, este processo' +
                        ' é irreversivel</span><br><br>';
              txt += 'Deseja realmente congelar o inventário<br>' + cod +
                     ' - ' + lbl + ' ?<br><br><br>';
              txt += '<div id="OPCOES_CON"><form name="Congelar" ' +
                     'onsubmit="return false;">   ';
              txt += '<input type="button" name="OK" id="OK" ' +
                     'value="   Congelar   " ';
              txt += 'onclick="objCongelamento.congelar('+
                     cod + ');">      ';
              txt += '<input type="button" name="NOK" id="OK" value="Cancelar" ';
              txt += 'onclick="hideDialog();" style="color:red;">' +
                     '</form></div><br><br>';
              showDialog('Congelamento de inventário',txt,'prompt');
         },
congelar: function(cod) {
              var params = 'classe=congelamento&metodo=congelar' +
                           '&arquivo_classe=estoque/classes/classe_congelamento.inc' +
                           '&inventario='+cod;
              ObjProcAjax.runPost('executa_busca_ajax.php5',
                                  'OPCOES_CON',params);
         }
};
var objCongelamento = new congelar();
```

Lista 6: classe_congelamento.inc

```
<?php
/**
 * Classe inventario
 */

class congelamento extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'inventario';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("INVENTARIO_CODIGO","Inventário",
                    10,null,null,true,false,true,1,null,null,false,$_conn));
          $this->getCampo("INVENTARIO_CODIGO")->setComportamento_form('select');
     }

     /**
      * Sempre executamos o módulo de relatório especial
      *
      */
     public function processaAcao() {
```

```
        // Sempre mostra o filtro do relatório nada mais...
        $_inv = new inventario($this->_conn);
        $this->getCampo("INVENTARIO_CODIGO")->setValor_fixo($_inv-
>buscaLiberados());
        $_form = new formulario("CONGELAR_{$this->_nome_tabela}",
                                "Javascript:void(0);");
        $_form->deleteAtributo(4);
        $_form->addAtributo(new atributo("ONSUBMIT",
                "objCongelamento.run(retornaFiltro(this,false));" .
                "return false;"));
        $_tab = $this->getTabelaFiltro(false);
        $_tab->getLastSubTag()->addSubTag(new tag(new tipotag("TD"),
                Array(new atributo("COLSPAN",4),
                  new atributo('STYLE','text-align:right;padding-right:20px;'),
                  new atributo("ID","BTNOK"))));
        $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                new formInputSubmit("OK"," Congelar Inventário "));
        $_form->addSubTag($_tab);
        $_sc = new tag(new tipotag("SCRIPT"),null,
                    "LoadJS('estoque/javascript/congelamento.js');");
        echo utf8_encode($this->getInfoPagina()->toHTML() .
                         $this->inibeBotoes()->toHTML() .
                         $_form->toHTML() .
                         $_sc->toHTML());
    }

    public function congelar() {
        $_cinv = $_POST['inventario'];
        $_inv = new inventario($this->_conn);
        if($_inv->Buscar(Array('INVENTARIO_CODIGO'=>$_cinv))===false) {
            echo utf8_encode("<span style='color:red;'>" .
                             "Inventário {$_cinv} Não Encontrado</span>");
        } else {
            $_inv->proximo();
            if($_inv->getCampo("INVENTARIO_CONGELADO")=='S') {
                echo utf8_encode("<span style='color:red;'>" .
                             "Inventário {$_cinv} Já está congelado</span>");
            } else {
                $this->_conn->startTransaction();
                $_itens = new inventarioproduto($this->_conn);
                if($_itens->congelar($_cinv)===false) {
                    $this->_conn->ROLLBACK();
                    echo utf8_encode("<span style='color:red;'>" .
                                     "Não foi possível executar o congelamento " .
                                     "dos itens do Inventário {$_cinv}</span>");
                } else {
                    if($_inv->congelar($_cinv)===false) {
                        $this->_conn->ROLLBACK();
                        echo utf8_encode("<span style='color:red;'>" .
                                         "Não foi possível executar o congelamento " .
                                         "do Inventário {$_cinv}</span>");
                    } else {
                        $this->_conn->commit();
                        echo utf8_encode("<span style='color:Navy;'>" .
                                         "Inventário {$_cinv} congelado com " .
                                         "sucesso.</span>");
                    }
                }
            }
        }
    }
}
?>
```

Lista 7: inv_congelarinventario.php5

```
<?php
/**
 * Congelamento de inventário
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_congelamento.inc",
              "nome"=>"congelamento");
$_FTitulo = "Congelamento de Inventário";
return include_once("../instanciaclasse.php5");
?>
```

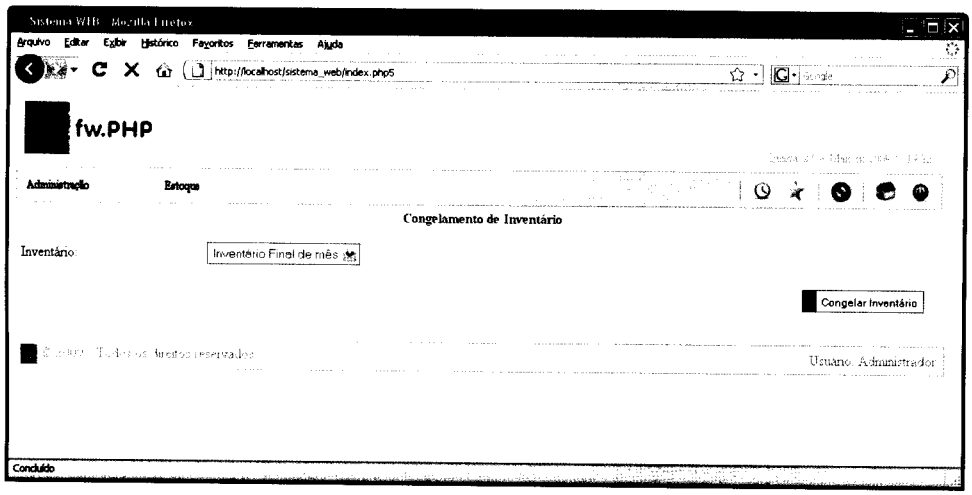


Figura 9.6

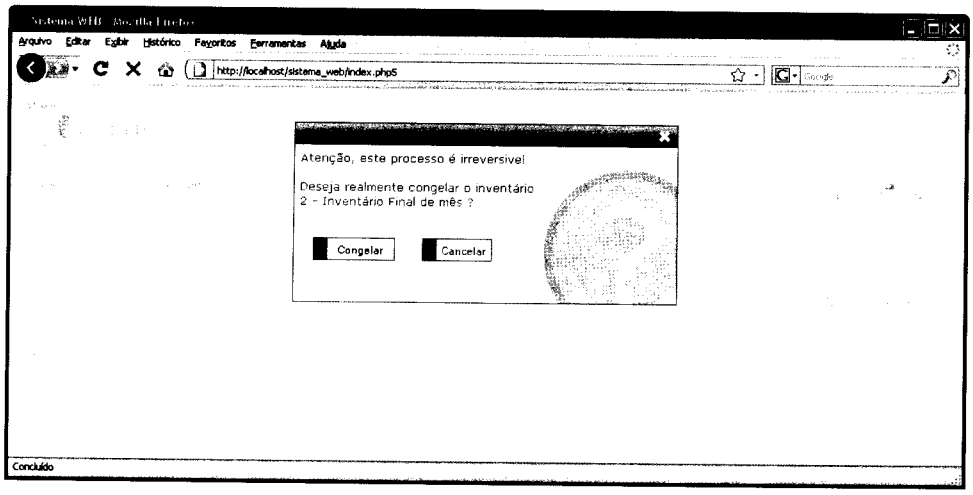


Figura 9.7

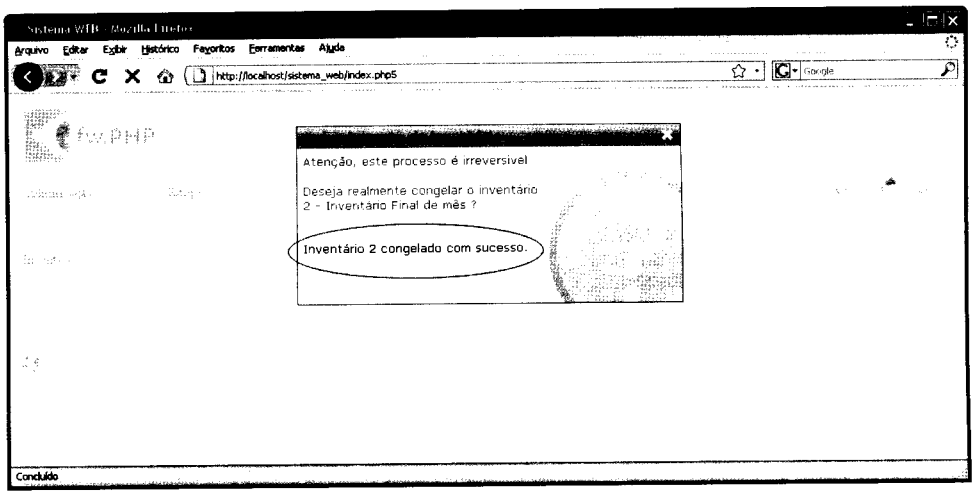


Figura 9.8

Fichas de inventário

Para executar a contagem dos estoques, é necessário que sejam emitidas as fichas de inventário. Elas contêm as informações dos produtos e locais, bem como o número da contagem que está sendo executada. Só os inventários congelados e não consolidados podem ser listados para emissão das fichas.

O usuário deve ter a opção de listar todos os produtos participantes do inventário ou somente os produtos que apresentaram alguma divergência (se for a primeira contagem, obrigatoriamente todas as fichas serão impressas). O sistema deve emitir uma ficha para cada relação produto e local, ou seja, uma ficha para cada registro na tabela inventarioproduto.

Para imprimir as fichas precisamos redefinir os métodos de impressão do framework, uma vez que o formato é específico para essa funcionalidade.

O usuário terá como opções de seleção:

- Inventário para impressão
- Indicador de impressão somente das divergências

Para listar os inventários disponíveis para emissão das fichas, precisamos de um novo método na classe **inventario**. Esse método será responsável por buscar os inventários congelados e não consolidados e gerar a lista de inventários no formato aceito pelo framework para campos com atributo 'comportamento form' marcado como 'select'.

```
/**
 * Retorna a Lista de inventários disponíveis
 * Ou seja Congelado=S Consolidado=N
 *
 * @return array
 */
public function buscaLiberadosFicha() {
     if($this->_conn->executaSQL(
          $this->montaSELECT("INVENTARIO_CODIGO,INVENTARIO_DESC",
                        "(INVENTARIO_CONGELADO='S' AND " .
                        "INVENTARIO_CONSOLIDADO='N')",
                        "INVENTARIO_CODIGO DESC"))===false) {
          return Array();
     } else {
          $_inv = Array();
          while(($_dados=$this->_conn->proximo())!==false) {
               $_inv[] = Array(
                              "valor"=>$_dados['INVENTARIO_CODIGO'],
                              "label"=>$_dados['INVENTARIO_DESC']);
          }
          return $_inv;
     }
}
```

Cada ficha traz as seguintes informações:

- Código do inventário
- Descrição do inventário
- Número da contagem atual
- Código do produto
- Descrição do produto
- Unidade de medida
- Código do local
- Descrição do local
- Espaço para quantidade inventariada

O modelo seguinte mostra como deve ser uma ficha de inventário (você encontra no mercado vários modelos diferentes).

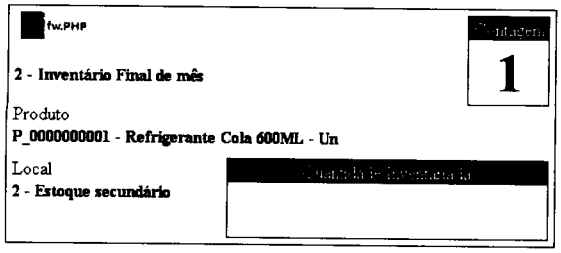


Figura 9.9

Lista 8: classe_fichainventario.inc

```
<?php
/**
 * Classe Fichas de inventario
 */

class fichainventario extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'inventarioproduto';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("INVENTARIO_CODIGO","Inventário",
                    10,null,null,true,false,true,1,null,null,false,$_conn));
          $this->addCampo(new string("MODO","Somente Divergentes",
                    1,null,null,false,false,true,1,null,null,false,$_conn));
          $this->addCampo(new inteiro("INVENTARIO_CONTAGEM","Contagem atual",
                    1,null,null,false,false,false,1,null,null,false));
          $this->addCampo(new string("PRODUTO","Produto",
                    60,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("UNIDADE_CODIGO","Unidade",
                    2,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("LOCAL","Local",
                    40,null,null,false,false,true,1,null,null,true,$_conn));
          $this->getCampo("INVENTARIO_CODIGO")->setComportamento_form('select');
          $this->getCampo("MODO")->setComportamento_form('checkbox');
          $this->getCampo("MODO")->setValor_fixo('D');
          $this->getCampo("MODO")->setMarcar(false);
          $this->getCampo("INVENTARIO_CONTAGEM")->setFiltro(false);
          $this->getCampo("LOCAL")->setFiltro(false);
          $this->getCampo("PRODUTO")->setFiltro(false);
          $this->getCampo("UNIDADE_CODIGO")->setFiltro(false);
     }

     /**
      * Sempre executamos o módulo de relatório especial
      *
      */
     public function processaAcao() {
          // Sempre mostra o filtro do relatório nada mais...
          $_inv = new inventario($this->_conn);
          $this->getCampo("INVENTARIO_CODIGO")->setValor_fixo(
                                                  $_inv->buscaLiberadosFicha());
          $this->geraFormularioFiltroRelatorio();
     }

     /**
      * Uma classe de relatórios deve sempre montar seu select
      *
      */
     public function montaSELECT($_campos,$_where=null,
                                 $_orderby=null,$_limit=null,$_extras=null) {
          $_strSQL= "SELECT I.INVENTARIO_CODIGO||' - '||I.INVENTARIO_DESC AS " .
                    "INVENTARIO_CODIGO,I.INVENTARIO_CONTAGEM," .
                    "P.PRODUTO_CODIGO||' - '||P.PRODUTO_DESC AS " .
                    "PRODUTO,P.UNIDADE_CODIGO," .
                    "L.LOCAL_CODIGO||' - '||L.LOCAL_DESC AS LOCAL " .
                    "FROM {$this->_nome_tabela} IP " .
                    "LEFT JOIN INVENTARIO I ON " .
                    "IP.INVENTARIO_CODIGO=I.INVENTARIO_CODIGO " .
                    "LEFT JOIN PRODUTO P ON IP.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
                    "LEFT JOIN LOCALESTOQUE L ON IP.LOCAL_CODIGO=L.LOCAL_CODIGO ";
          // Filtro
```

```
        $_strSQL .= " WHERE " .
                    "IP.INVENTARIO_CODIGO='{$_POST['INVENTARIO_CODIGO']}'";
        if($_POST['MODO']=='D') {
            $_strSQL .= " AND (IP.INVENTARIO_ESTOQUE<>IP.PRODUTO_ESTOQUE OR " .
                        "OR IP.INVENTARIO_ESTOQUE ISNULL)";
        }
        $_strSQL .=    " ORDER BY IP.PRODUTO_CODIGO";
        return $_strSQL;
    }


    /**
     * Gera o cabeçalho inicial do sistema
     *
     */
    protected function addCabecalho() {
        $this->_html = new tag(new tipotag("HTML"));
        $this->_html->addSubTag(new tag(new tipotag("HEAD")));
        $this->_html->getLastSubTag()->addSubTag(new tag(
                                new tipotag("TITLE"),null,"Sistema WEB"));
        $this->_html->addSubTag(new tag(new tipotag("BODY"),
                                Array(new atributo("WIDTH","1010"))));
    }

    /**
     * Impressão do relatório
     *
     */
    public function imprimirRelatorioEspecial() {
        $this->addCabecalho();
        // Imprime o Relatório
        $_tabela    = new tag(new tipotag("TABLE"),
                              Array(new atributo("BORDER",0),
                                    new atributo("CELLPADDING",0),
                                    new atributo("CELLSPACING",0)));
        $_colunas   = null;
        if($this->_conn->executaSQL($this->montaSELECT(""))!==false&&
           $this->_conn->getNumRows()>0) {
            while($this->proximo()) {
                if($_colunas>=3||$_colunas===null) {
                    if($_colunas!==null) {
                        $_tabela->addSubTag(new tag(new tipotag("TR")));
                        $_tabela->getLastSubTag()->addSubTag(new tag(
                                  new tipotag("TD"),
                                  Array(new atributo("HEIGHT","20")),
                                  " "));
                    }
                    $_tabela->addSubTag(new tag(new tipotag("TR")));
                    $_colunas = 1;
                }
                ++$_colunas;
                $_tabela->getLastSubTag()->addSubTag(new tag(
                          new tipotag("TD"),
                          Array(new atributo("WIDTH","520"))));
                $_tab_int = new tag(new tipotag("TABLE"),
                              Array(new atributo("BORDER",0),
                                    new atributo("CELLSPACING",0),
                                    new atributo("CELLPADDING",0),
                                    new atributo("STYLE","width:100%;" .
                                    "border:1px solid black;padding:5px;")));
                // Cabeçalho
                $_tab_int->addSubTag(new tag(new tipotag("TR")));
                $_tab_int->getLastSubTag()->addSubTag(new tag(
                          new tipotag("TD"),
```

```
          Array(new atributo("STYLE",
                    "vertical-align:top;padding-left:5px;"))));
$_tab_int->getLastSubTag()->getLastSubTag()->addSubTag(
          new tag(new tipotag("IMG",false),
          Array(new atributo("SRC",
                    "framework/imagens/logo.png"),
                new atributo("BORDER",0),
                new atributo("ALIGN","BOTTOM"),
                new atributo("WIDTH",60))));
$_tab_int->getLastSubTag()->addSubTag(new tag(
          new tipotag("TD"),
          Array(new atributo("ROWSPAN",2),
               new atributo("STYLE",
                    "border:1px solid black;width:15%;"))));
$_tab_int->getLastSubTag()->getLastSubTag()->addSubTag(
          new tag(new tipotag("DIV"),
          Array(new atributo("STYLE",
               "text-align:center;background-color:black;" .
               "color:white;")),
          "Contagem"));
$_tab_int->getLastSubTag()->getLastSubTag()->addSubTag(
          new tag(new tipotag("DIV"),
          Array(new atributo("STYLE","font-size:48px;" .
               "text-align:center;font-weight:bold;")),
          $this->getCampo("INVENTARIO_CONTAGEM")->toPrint()));
$_tab_int->addSubTag(new tag(new tipotag("TR")));
$_tab_int->getLastSubTag()->addSubTag(new tag(
          new tipotag("TD"),
          Array(new atributo("STYLE",
               "vertical-align:top;width:85%;" .
               "text-align:left;font-size:14px;" .
               "font-weight:bold;")),
          $this->getCampo("INVENTARIO_CODIGO")->toPrint()));
$_tab_int->getLastSubTag()->addSubTag(new tag(
          new tipotag("TD")));
// Produto + Local
$_tab_int->addSubTag(new tag(new tipotag("TR")));
$_tab_int->getLastSubTag()->addSubTag(new tag(
          new tipotag("TD"),
          Array(new atributo("COLSPAN",2),
               new atributo("STYLE","padding-bottom:10px;")),
          "Produto<br>"));
$_tab_int->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("SPAN"),
               Array(new atributo("STYLE",
                         "font-size:14px;font-weight:bold;")),
               $this->getCampo("PRODUTO")->toPrint() . " - " .
               $this->getCampo("UNIDADE_CODIGO")->toPrint()));
// Quantidade
$_tab_int->addSubTag(new tag(new tipotag("TR")));
$_tab_int->getLastSubTag()->addSubTag(new tag(
     new tipotag("TD"),Array(new atributo("COLSPAN",2))));
$_tab_int->getLastSubTag()->getLastSubTag()->addSubTag(
     new tag(new tipotag("TABLE"),
          Array(new atributo("WIDTH","100%"),
               new atributo("CELLSPACING",0),
               new atributo("CELLPADDING",0))));
$_tab_int->getLastSubTag()->getLastSubTag()->
     getLastSubTag()->addSubTag(new tag(new tipotag("TR")));
$_tab_int->getLastSubTag()->getLastSubTag()->
     getLastSubTag()->getLastSubTag()->getLastSubTag()->addSubTag(
               new tag(new tipotag("TD"),
                    Array(new atributo("WIDTH","40%"),
                         new atributo("VALIGN","TOP")),
```


```
                                        "Local<br>"));
                    $_tab_int->getLastSubTag()->getLastSubTag()->
                         getLastSubTag()->getLastSubTag()->
                         getLastSubTag()->addSubTag(new tag(
                                   new tipotag("SPAN"),
                                        Array(new atributo("STYLE",
                                             "font-size:14px;font-weight:bold;")),
                                        $this->getCampo("LOCAL")->toPrint()));
                    $_tab_int->getLastSubTag()->getLastSubTag()->
                         getLastSubTag()->getLastSubTag()->addSubTag(
                                   new tag(new tipotag("TD"),
                                        Array(new atributo("ALIGN","RIGHT"),
                                          new atributo("STYLE",
                                             "border:1px solid black;" .
                                             "vertical-align:top;width:60%;" .
                                             "height:70px;"))));
                    $_tab_int->getLastSubTag()->getLastSubTag()->
                         getLastSubTag()->getLastSubTag()->
                         getLastSubTag()->addSubTag(new tag(
                                   new tipotag("DIV"),
                                   Array(new atributo("STYLE",
                                        "text-align:center;vertical-align:top;" .
                                        "background-color:black;color:white;")),
                                   "Quantidade Inventariada"));
                    $_tabela->getLastSubTag()->getLastSubTag()->addSubTag(
                                   $_tab_int);
                    if($_colunas==2) {
                         $_tabela->getLastSubTag()->addSubTag(new tag(
                              new tipotag("TD"),Array(new atributo("WIDTH","20"))));
                    }
               }
          } else {
               $this->_html->setValor(" * Nenhum Registro *");
          }
          $this->_html->getLastSubTag()->addSubTag($_tabela);
          $this->addbotaoimpressao();
          echo utf8_encode($this->_html->toHTML());
     }

     /**
      * Adiciona o botão de impressão do relatório
      *
      */
     protected function addbotaoimpressao() {
          $this->_html->addSubTag(new tag(new tipotag("DIV"),
               Array(new atributo("ID","DIVPRINT"),
                    new atributo("STYLE","padding:10px;text-align:right;"))));
          $this->_html->getLastSubTag()->addSubTag(new tag(
               new tipotag("IMG",false),
               Array(new atributo("SRC","framework/imagens/print.png"),
                    new atributo("BORDER",0),
                    new atributo("ID","BTN_PRINT"),
                    new atributo("TITLE","Imprimir"),
                    new atributo("STYLE","cursor:pointer"),
                    new atributo("ONCLICK",
                       "this.parentNode.style.display='none';window.print();" .
                       "var obj=this;setTimeout(function(){" .
                       "obj.parentNode.style.display='block';},3000);")
          )));
     }
}
?>
```

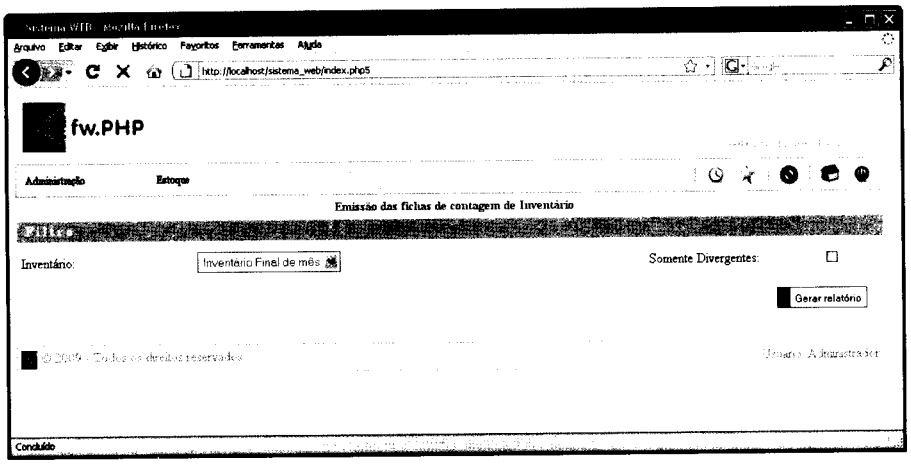


Figura 9.10

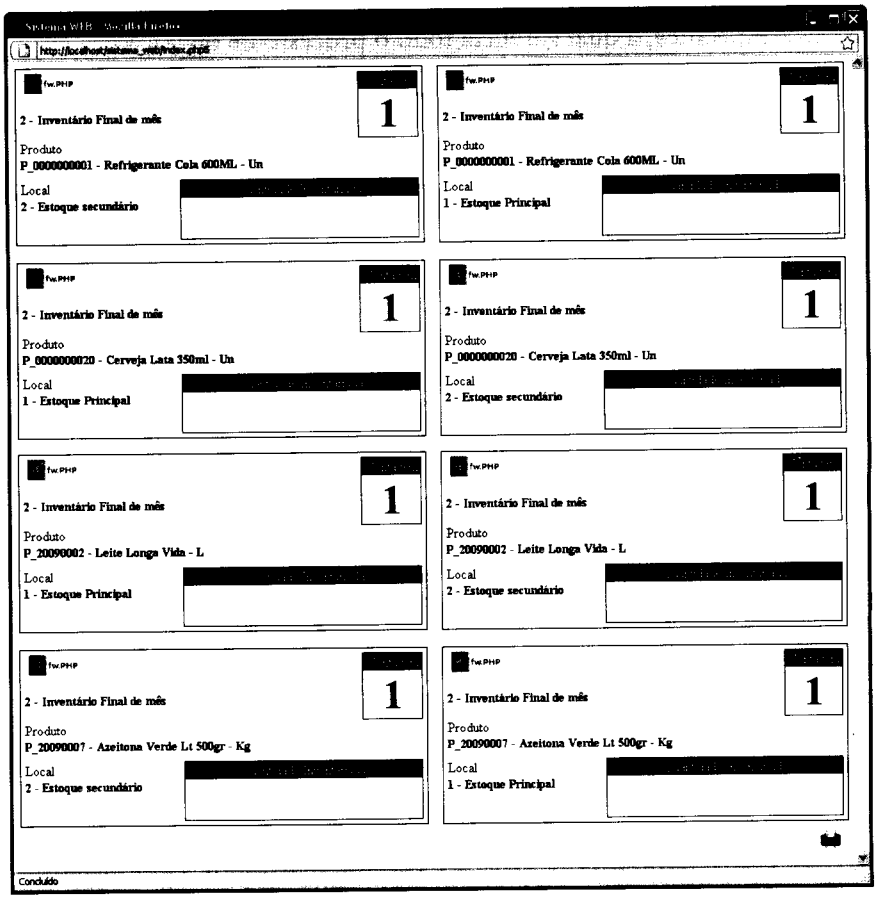


Figura 9.11

Lista 9: inv_fichainventario.php5

```
<?php
/**
 * Ficha de inventário
 *
 */
$_classe = Array(
              "arquivo"=>"estoque/classes/classe_fichainventario.inc",
              "nome"=>"fichainventario");
$_FTitulo = "Emissão das fichas de contagem de Inventário";
return include_once("../instanciaclasse.php5");
?>
```

Entrada de contagem

Com as fichas de inventário emitidas, o processo de contagem pode ser iniciado e uma vez que os estoques são inventariados, é necessário informar ao sistema as quantidades contadas de cada produto e local. Para o lançamento dessas contagens precisamos criar uma classe específica, a qual gerencia todo o processo de digitação.

Vamos precisar, ainda, de uma classe javascript para o controle do processo, uma vez que teremos algumas situações para controlar:

- Ao selecionar o inventário, é preciso exibir a lista de produtos disponíveis e ainda não contados.
- Ao selecionar o produto, deve-se exibir a lista de locais disponíveis e ainda não contados.
- Ao selecionar o local, deve-se habilitar o campo de digitação da quantidade contada.
- Após incluir uma contagem, os dados dela devem ser incluídos na lista de contagem existente no formulário.

A página de contagem só executa a opção de inclusão de contagem, ou seja, não vamos exibir uma lista de contagem nem mostrar opções para alteração ou exclusão da contagem. Para isso devemos forçar a função a ser executada como 'INC' e, uma vez que o formulário é específico para esse processo, redefinir o método *getFormulario()* para que seja gerado o formulário no formato desejado.

Os seguintes campos devem ser definidos na classe:

1. INVENTARIO_CODIGO

   Instância da classe **string**, contém a lista de inventários disponíveis para entrada da contagem. Este é um campo de seleção, ou seja, o atributo 'comportamento form' deve ser alterado para 'select'.

2. CONTAGEM_NUMERO

   Instância da classe **inteiro**, contém o número da contagem que está sendo registrada. O valor desse campo é obtido diretamente da classe **inventario**. Ele é informativo no formulário de inclusão da contagem, porém é utilizado no momento da gravação.

3. PRODUTO_CODIGO

   Instância da classe **string**, contém o código do produto que foi contado. Esse campo tem seu atributo 'comportamento form' definido para 'select', e a lista é formada pelos produtos que estão na tabela inventarioproduto e que não foram contados ainda.

4. LOCAL_CODIGO

   Instância da classe **string**, contém o código do local que foi contado. Esse campo tem seu atributo 'comportamento form' definido para 'select', e a lista é formada pelos locais relacionados ao produto e que estão na tabela inventarioproduto e não foram contados ainda.

5. CONTAGEM_ESTOQUE

   Instância da classe **float**, contém a quantidade contada para a relação produto, local e número da contagem.

O método *gravarContagem()* recebe os dados da contagem atual e gera um novo registro na tabela contagem.

Lista 10: classe_contagem.inc

```
<?php
/**
 * Classe contagem do inventario
 */
class contagem extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'contagem';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("INVENTARIO_CODIGO","Inventário",
                    10,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new inteiro("CONTAGEM_NUMERO","Número da contagem",
                    5,null,null,true,false,true,1,null,null,true));
          $this->addCampo(new string("PRODUTO_CODIGO","Produto",
                    20,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new string("LOCAL_CODIGO","Local",
                    10,null,null,true,false,true,1,null,null,true,$_conn));
          $this->addCampo(new float("CONTAGEM_ESTOQUE","Quantidade Contada",
                    12,null,null,true,false,true,0,null,null,false));
          $this->addCampo(new string("USUARIO_LOGIN","Usuário",
                    20,null,null,false,false,false,null,null,null,false,$_conn));
          $this->addCampo(new datahora("DATA_ULTIMA_ALTERACAO","Data",
                    14,null,null,false,false,false,null,null,null,false,$_conn));
          $this->getCampo("INVENTARIO_CODIGO")->setComportamento_form('select');
          $this->getCampo("PRODUTO_CODIGO")->setComportamento_form('select');
```

```
        $this->getCampo("LOCAL_CODIGO")->setComportamento_form('select');
    }


    /**
     * Ajusta o comportamento dos campos do formulário
     *
     */
    protected function processaCampoFormulario($_nome,tag &$_campo,$_metodo) {
        if(stripos($_metodo,'getformulario')!==false) {
            switch($_nome) {
                case 'CONTAGEM_ESTOQUE':
                case 'CONTAGEM_NUMERO':
                                $_campo->addAtributo(new atributo("DISABLED"));
                                break;
                case 'INVENTARIO_CODIGO':
                                $_campo->addAtributo(new atributo("ONCHANGE",
                                    "objContagem.habilita();" .
                                    "objContagem.buscaContagem();" .
                                    "objContagem.buscaProdutos();"));
                                break;
                case 'PRODUTO_CODIGO':
                                $_campo->addAtributo(new atributo("ONCHANGE",
                                    "objContagem.buscaLocais();"));
                                break;
                case 'LOCAL_CODIGO':
                                $_campo->addAtributo(new atributo("ONCHANGE",
                                    "objContagem.habilita();"));
                                break;
            }
        }
    }


    /**
     * Não é possível excluir uma contagem
     *
     */
    public function Excluir() {
        return false;
    }

    /**
     * Não é possível alterar uma contagem
     *
     * @return unknown
     */
    public function Alterar() {
        return false;
    }

    /**
     * Monta o formulário específico de entrada de contagem
     *
     * @param string $_funcao
     * @return string
     */
    public function getFormulario($_funcao='INC') {
        $_inv = new inventario($this->_conn);
        $_opc = Array(Array('label'=>'Selecione um Inventário','valor'=>'-'));
        $this->getCampo("INVENTARIO_CODIGO")->setValor_Fixo(
                                Array_merge($_opc,$_inv->buscaLiberadosFicha()));
        $_form = new formulario("FRM_{$this->_nome_tabela}","#");
        $_form->deleteAtributo(4);
```

```
$_form->addAtributo(new atributo("ONSUBMIT",
                        "objContagem.gravar();return false;"));
$_form->addSubTag(new tag(new tipotag("SCRIPT"),null,
                        "objForm.construct();"));
$_tab  = new tag(new tipotag("TABLE"),
                    Array(new atributo("BORDER",0),
                        new atributo("CELLPADDING",2),
                        new atributo("CELLSPACING",0)));
$_tr = new tipotag("TR");
$_td = new tipotag("TD");
$_num_colunas = 1;
$_cols = $_num_colunas+1;
$_foco = null;
$_lhidden = Array();
foreach($this->filtrarCampos() as $_campo) {
     if($_cols>$_num_colunas) {
          $_tab->addSubTag(new tag($_tr));
          $_cols = 1;
     }
     ++$_cols;
     $_tab->getLastSubTag()->addSubTag(new tag($_td,
          Array(new atributo("STYLE",
                        "color:Navy;width:200px;padding-left:5px;")),
          "{$_campo->getTitulo()}:"));
     $_tab->getLastSubTag()->addSubTag(new tag($_td));
     if($_foco===null) {
          $_foco = $_campo->getNome();
     }
     if(strtolower($_campo->getComportamento_form())=='radio'&&
       is_array($_campo->getValor_fixo())) {
          foreach($_campo->getValor_Fixo() as $_opcao) {
               $_cpoform = new formInputRadio($_campo->getNome(),
                              $_opcao['valor'],$_opcao['label'],null,null,
                         ($_opcao['marcar']===true||
                          $_campo->getValor()==$_opcao['valor']));
               $this->processaCampoFormulario($_campo->getNome(),
                              $_cpoform,__METHOD__);
               $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                              $_cpoform);
               $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                              new tag(new tipotag('BR',false)));
          }
     } else {
          $_cpoform = $_campo->toForm();
          $this->processaCampoFormulario($_campo->getNome(),
                              $_cpoform,__METHOD__);
          $_tab->getLastSubTag()->getLastSubTag()->addSubTag($_cpoform);
     }
}
$_tab->addSubTag(new tag($_tr));
$_tab->getLastSubTag()->addSubTag(new tag($_td,
          Array(new atributo("COLSPAN",$_num_colunas*2),
               new atributo('ALIGN','CENTER'),
               new atributo("ID","BTNOK"))));
$_tab->getLastSubTag()->getLastSubTag()->addSubTag(
          new formInputSubmit("OK"," Gravar Contagem"));
$_form->addSubTag($_tab);
if($_foco!==null) {
     $_form->addSubTag(new tag(
          new tipotag("SCRIPT"),null,
          "document.getElementById('{$_foco}').focus();"));
}
$_tab = new tag(new tipotag('TABLE'),
               Array(new atributo("BORDER",0),
```

```
                        new atributo("CELLPADDING",1),
                        new atributo("STYLE","border:1px solid #a0a0a0;" .
                                             "border-collapse:collapse;"),
                        new atributo("WIDTH","100%"),
                        new atributo("ID","TAB_CONTAGEM")));
        $_bd  = new tag(new tipotag("TBODY"));
        $_tr  = new tag($_tr);
        $_tr->addSubTag(new tag(new tipotag('TH'),
             Array(new atributo("STYLE","border:1px solid #a0a0a0;" .
                                        "background-color:#f0f0f0;")),
             "Inventário"));
        $_tr->addSubTag(new tag(new tipotag('TH'),
             Array(new atributo("STYLE","border:1px solid #a0a0a0;" .
                                        "background-color:#f0f0f0;")),
             "Contagem"));
        $_tr->addSubTag(new tag(new tipotag('TH'),
             Array(new atributo("STYLE","border:1px solid #a0a0a0;" .
                                        "background-color:#f0f0f0;")),
             "Produto"));
        $_tr->addSubTag(new tag(new tipotag('TH'),
             Array(new atributo("STYLE","border:1px solid #a0a0a0;" .
                                        "background-color:#f0f0f0;")),
             "Local"));
        $_tr->addSubTag(new tag(new tipotag('TH'),
             Array(new atributo("STYLE","border:1px solid #a0a0a0;" .
                                        "background-color:#f0f0f0;")),
             "Quantidade"));
        $_tr->addSubTag(new tag(new tipotag('TH'),
             Array(new atributo("STYLE","border:1px solid #a0a0a0;" .
                                        "background-color:#f0f0f0;")),
             "Situação"));
        $_bd->addSubTag($_tr);
        $_tab->addSubTag($_bd);
        return utf8_encode($_form->toHTML() . $_tab->toHTML());
    }

    /**
     * Processa a ação, que no caso é sempre Inc
     *
     * @param string $_acao
     */
    public function processaAcao($_acao='INC') {
        $_sc = new tag(new tipotag("SCRIPT"),null,
                       "LoadJS('estoque/javascript/contagem.js');");
        echo utf8_encode($this->getInfoPagina()->toHTML() .
                       $this->inibeBotoes()->toHTML());
        parent::processaAcao();
        echo $_sc->toHTML();
    }

    /**
     * Grava a contagem
     *
     */
    public function GravarContagem() {
        // inventário, contagem, produto, local, quantidade...
        foreach($this->filtrarCampos("incluir") as $_nome=>$_campo) {
            $_campo->setValor($_POST[$_nome]);
        }
        $this->SetUsuarioeData();
        if(($_res=$this->incluir())!==false) {
            echo "Contagem Gravada";
        } else {
            echo utf8_encode("Erro na gravação:" .
```

```
                                "{$_res} {$this->_conn->getUltimoErro()}");
        }
    }
}
?>
```

A classe javascript para auxílio no processo de contagem disponibiliza métodos para atender às seguintes necessidades do processo:

- Busca de produtos disponíveis relacionados ao inventário;
- Busca de locais disponíveis para contagem;
- Habilitação/inibição da digitação da quantidade inventariada;
- Gravação da contagem;
- Inclusão dos dados da contagem na lista de contagens executadas.

Lista 11: contagem.js

```
// rotinas javascript para contagem
// inventário

function contagem() {};
contagem.prototype = {
buscaContagem: function() {
                        var params='classe=inventario&metodo=buscaContagem'  +
                                  '&arquivo_classe=estoque/classes/' +
                                  'classe_inventario.inc';
                        params += '&INVENTARIO_CODIGO='+
                                   objSelect.checked('INVENTARIO_CODIGO');
                        ObjProcAjax.runPost('executa_busca_ajax.php5',
                                            'CONTAGEM_NUMERO',params);
                    },
buscaProdutos: function() {
                        this.habilita();
                        objSelect.clearAll('PRODUTO_CODIGO');
                        objSelect.clearAll('LOCAL_CODIGO');
                        if(objSelect.checked('INVENTARIO_CODIGO')!=='-') {
                             var params='classe=inventarioproduto&'  +
                                  'metodo=buscaprodutos&arquivo_classe=' +
                                  'estoque/classes/classe_inventarioproduto.inc';
                             params += '&INVENTARIO_CODIGO='+
                                       objSelect.checked('INVENTARIO_CODIGO');
                             ObjProcAjax.runPost('executa_busca_ajax.php5',
                                                 null,params,
                                                 objContagem.addProdutos);
                        }
                    },
addProdutos:  function(txt) {
                        var arr = txt.evalJSON();
                        objSelect.addOption('PRODUTO_CODIGO','-',
                                            'Selecione o produto');
                        for(var i=0;i<arr.length;i++) {
                             objSelect.addOption('PRODUTO_CODIGO',
                                                 arr[i].PRODUTO_CODIGO,
                                                 arr[i].PRODUTO_DESC);
                        }
                    },
buscaLocais:  function() {
```

```
                    this.habilita();
                    objSelect.clearAll('LOCAL_CODIGO');
                    if(objSelect.checked('PRODUTO_CODIGO')!=='-') {
                        var params='classe=inventarioproduto&' +
                            'metodo=buscalocais&arquivo_classe=' +
                            'estoque/classes/classe_inventarioproduto.inc';
                        params += '&INVENTARIO_CODIGO='+
                                objSelect.checked('INVENTARIO_CODIGO') +
                                '&PRODUTO_CODIGO='+
                                objSelect.checked('PRODUTO_CODIGO');
                        ObjProcAjax.runPost('executa_busca_ajax.php5',
                                        null,params,objContagem.addLocais);
                    }
                },
addLocais:      function(txt) {
                    var arr = txt.evalJSON();
                    objSelect.addOption('LOCAL_CODIGO','-',
                                    'Selecione o Local');
                    if(arr.length===0) {
                        objContagem.buscaProdutos();
                    } else {
                        for(var i=0;i<arr.length;i++) {
                            objSelect.addOption('LOCAL_CODIGO',
                                            arr[i].LOCAL_CODIGO,
                                            arr[i].LOCAL_DESC);
                        }
                    }
                },
habilita:   function() {
                    var i = objSelect.checked('INVENTARIO_CODIGO');
                    if(i==='-') {
                        objSelect.clearAll('PRODUTO_CODIGO');
                    }
                    var p = objSelect.checked('PRODUTO_CODIGO');
                    if(p==='-') {
                        objSelect.clearAll('LOCAL_CODIGO');
                    }
                    var l = objSelect.checked('LOCAL_CODIGO');
                    var h = (i==='-'||p==='-'||l==='-') ? true : false;
                    document.getElementById('CONTAGEM_ESTOQUE').disabled = h;
                    if(h===false) {
                        document.getElementById('CONTAGEM_ESTOQUE').focus();
                    }
                },
gravar:         function() {
                    var i = objSelect.checked('INVENTARIO_CODIGO');
                    var p = objSelect.checked('PRODUTO_CODIGO');
                    var l = objSelect.checked('LOCAL_CODIGO');
                    var q = document.getElementById('CONTAGEM_ESTOQUE').value;
                    if(i==='-'||p==='-'||l==='-'||q===''||q===0) {
                        alert('Erro no processamento...'  +
                            'Informe os dados do inventário/contagem');
                        document.getElementById('INVENTARIO_CODIGO').focus();
                    } else {
                        if(objForm.estaOk()===true) {
                            // Gravar os dados da Contagem
                            var c =
                            document.getElementById('CONTAGEM_NUMERO').value;
                            var params='classe=contagem&' +
                                'metodo=gravarContagem&arquivo_classe=' +
                                'estoque/classes/classe_contagem.inc';
                            params += '&INVENTARIO_CODIGO='+i+
                                    '&PRODUTO_CODIGO='+p;
                            params += '&LOCAL_CODIGO='+l+
```

```
                                        '&CONTAGEM_ESTOQUE='+q+
                                        '&CONTAGEM_NUMERO='+c;
                                ObjProcAjax.runPost('executa_busca_ajax.php5',
                                                    null,params,
                                                    objContagem.posgravar);
                            }
                        }
                    },
addColuna:          function(tr,v) {
                        var td  = document.createElement("TD");
                        td.appendChild(document.createTextNode(v));
                        td.style.textAlign = 'center';
                        tr.appendChild(td);
                    },
posgravar:          function(txt) {
                        var tb = document.getElementById('TAB_CONTAGEM');
                        var tbody = tb.getElementsByTagName("TBODY")[0];
                        var row = document.createElement("TR");
                        objContagem.addColuna(row,
                            objSelect.texto('INVENTARIO_CODIGO'));
                        objContagem.addColuna(row,
                            document.getElementById('CONTAGEM_NUMERO').value);
                        objContagem.addColuna(row,
                            objSelect.texto('PRODUTO_CODIGO'));
                        objContagem.addColuna(row,
                            objSelect.texto('LOCAL_CODIGO'));
                        objContagem.addColuna(row,
                            document.getElementById('CONTAGEM_ESTOQUE').value);
                        objContagem.addColuna(row,txt);
                        tbody.appendChild(row);
                        objContagem.buscaLocais();
                        document.getElementById('CONTAGEM_ESTOQUE').value = '';
                    }
};

var objContagem = new contagem();
var objSelect   = new Select();
```

Para completar o processo, devemos criar os métodos *buscaContagem()* na classe **inventario**, *bvscaProdutos()* e *buscaLocais()* na classe **inventarioproduto**.

inventario

```
/**
 * Busca o número da contagem atual do inventário informado
 *
 */
public function buscaContagem() {
    $this->Buscar(Array("INVENTARIO_CODIGO"=>$_POST['INVENTARIO_CODIGO']));
    $this->proximo();
    echo $this->getCampo("INVENTARIO_CONTAGEM")->toHTML();
}
```

inventarioproduto

```
/**
 * Busca os produtos do inventário
 * retornando os dados no formato JSON
 *
 * @return string JSON
```

```
 */
public function buscaprodutos() {
     $_strSQL = "SELECT DISTINCT IP.PRODUTO_CODIGO,P.PRODUTO_DESC FROM " .
          "{$this->_nome_tabela} IP " .
          "LEFT JOIN INVENTARIO I ON IP.INVENTARIO_CODIGO=I.INVENTARIO_CODIGO " .
          "LEFT JOIN PRODUTO P ON IP.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
          "WHERE NOT EXISTS (" .
          "    SELECT 1 FROM CONTAGEM C " .
          "    WHERE C.PRODUTO_CODIGO=IP.PRODUTO_CODIGO AND " .
          "    C.LOCAL_CODIGO=IP.LOCAL_CODIGO AND " .
          "    C.INVENTARIO_CODIGO=IP.INVENTARIO_CODIGO AND " .
          "    C.CONTAGEM_NUMERO=I.INVENTARIO_CONTAGEM) " .
          "AND IP.INVENTARIO_CODIGO='{$_POST['INVENTARIO_CODIGO']}'";
     $_arr = Array();
     if($this->_conn->executaSQL($_strSQL)!==false&&
        $this->_conn->getNumRows()>0) {
          while(($_dados=$this->_conn->proximo())!==false) {
               $_arr[] = array_map(utf8_encode,$_dados);
          }
     }
     echo json_encode($_arr);
}

/**
 * Busca os Locais do produto selecionado do inventário
 * retornando os dados no formato JSON
 *
 * @return string JSON
 */
public function buscaLocais() {
     $_strSQL = "SELECT DISTINCT IP.LOCAL_CODIGO,L.LOCAL_DESC FROM " .
          "{$this->_nome_tabela} IP " .
          "LEFT JOIN INVENTARIO I ON IP.INVENTARIO_CODIGO=I.INVENTARIO_CODIGO " .
          "LEFT JOIN LOCALESTOQUE L ON IP.LOCAL_CODIGO=L.LOCAL_CODIGO " .
               "WHERE NOT EXISTS (" .
               "    SELECT 1 FROM CONTAGEM C " .
               "    WHERE C.PRODUTO_CODIGO=IP.PRODUTO_CODIGO AND " .
               "    C.LOCAL_CODIGO=IP.LOCAL_CODIGO AND " .
               "    C.INVENTARIO_CODIGO=IP.INVENTARIO_CODIGO AND " .
               "    C.CONTAGEM_NUMERO=I.INVENTARIO_CONTAGEM) " .
               "AND IP.INVENTARIO_CODIGO='{$_POST['INVENTARIO_CODIGO']}' " .
               "AND IP.PRODUTO_CODIGO='{$_POST['PRODUTO_CODIGO']}'";
     $_arr = Array();
     if($this->_conn->executaSQL($_strSQL)!==false&&
        $this->_conn->getNumRows()>0) {
          while(($_dados=$this->_conn->proximo())!==false) {
               $_arr[] = array_map(utf8_encode,$_dados);
          }
     }
     echo json_encode($_arr);
}
```

Lista 12: inv_contagem.php5

```
<?php
/**
 * Contagem de inventário
 *
 */
$_classe = Array(
               "arquivo"=>"estoque/classes/classe_contagem.inc",
               "nome"=>"contagem");
```

```
$_FTitulo = "Contagem de Inventário";
$_GET['ACAO'] = 'INC';
return include_once("../instanciaclasse.php5");
?>
```

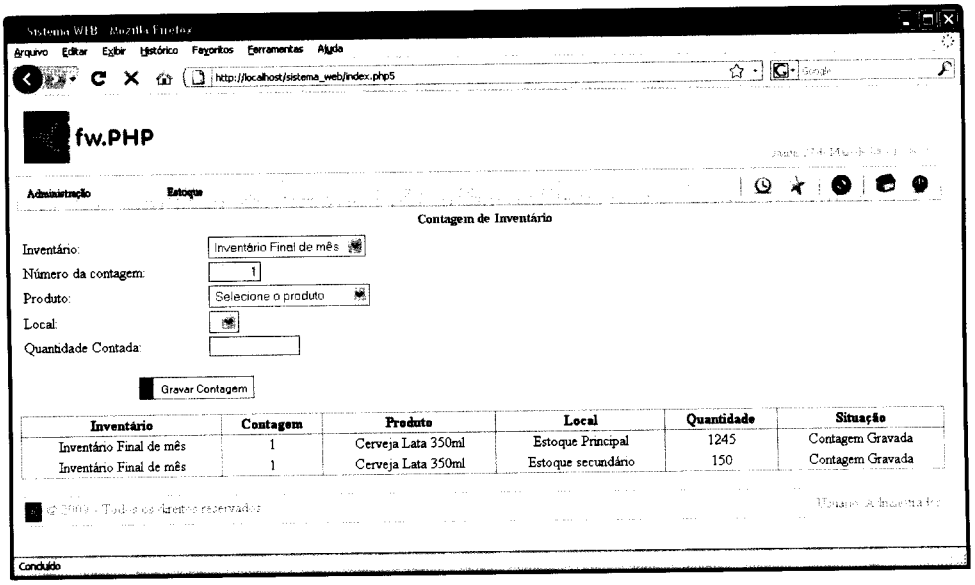


Figura 9.12

Análise de inventário

Após a digitação das fichas de contagem devemos executar o processo de análise do inventário. É nesse processo que validamos a contagem realizada. Seu resultado é a lista de produtos e locais considerados corretos (mesmo que o estoque esteja divergente) ou com divergências. O critério adotado para a validação da contagem é:

Se primeira contagem:

Estoque congelado = Estoque contado

PRODUTO_ESTOQUE = CONTAGEM_ESTOQUE

Caso contrário:

Estoque contado da contagem = Estoque contado da contagem anterior

$CONTAGEM\_ESTOQUE_n = CONTAGEM\_ESTOQUE_{(n-1)}$

Caso exista uma ou mais divergências, o sistema disponibiliza duas opções ao usuário:

- Confirmar as contagens corretas e gerar uma nova contagem (recontagem) para os produtos e locais divergentes.

- Consolidar o inventário mesmo com as divergências encontradas, definindo a última contagem como a base para atualização dos estoques (as validações anteriores permanecem).

Se não houver nenhuma divergência, o sistema disponibiliza apenas a opção de consolidação do inventário.

A classe **analiseinventario** possui apenas um atributo, que será utilizado para seleção do inventário para análise.

A classe possui também um método (além do construtor), para que seja gerado o formulário com as opções necessárias à execução do processo de análise do inventário.

Somente os inventários congelados e não consolidados são disponibilizados para seleção:

```
INVENTARIO_CONGELADO='S' AND INVENTARIO_CONSOLIDADO='N'
```

Lista 13: classe_analiseinventario.inc

```
<?php
/**
 * Classe analise do inventario
 */

class analiseinventario extends base {
     public function __construct(BancoDados $_conn) {
          parent::__construct($_conn);
          $this->_nome_tabela = 'inventario';
          $this->_class_path = 'estoque';
          $this->addCampo(new string("INVENTARIO_CODIGO","Inventário",
                    10,null,null,true,false,true,1,null,null,false,$_conn));
          $this->getCampo("INVENTARIO_CODIGO")->setComportamento_form('select');
     }

     /**
      * Exibição de uma tela específica para a análise de inventário
      *
      */
     public function processaAcao() {
          // Sempre mostra o filtro do relatório nada mais...
          $_inv = new inventario($this->_conn);
          $this->getCampo("INVENTARIO_CODIGO")->setValor_fixo(
                                   $_inv->buscaLiberadosFicha());
          $_form = new formulario("ANALISE_{$this->_nome_tabela}",
                                   "Javascript:void(0);");
          $_form->deleteAtributo(4);
          $_form->addAtributo(new atributo("ONSUBMIT",
                    "objAnalise.run(retornaFiltro(this,false));return false;"));
          $_tab = $this->getTabelaFiltro(false);
          $_tab->getLastSubTag()->addSubTag(new tag(new tipotag("TD"),
                    Array(new atributo("COLSPAN",4),
                      new atributo('STYLE','text-align:right;padding-right:20px;'),
                      new atributo("ID","BTNOK"))));
          $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
                    new formInputSubmit("OK"," Analisar o Inventário "));
          $_form->addSubTag($_tab);
          $_div = new tag(new tipotag("DIV"),
                    Array(new atributo("ID","RESULTADO_ANALISE")));
```

```
        $_sc = new tag(new tipotag("SCRIPT"),null,
                "LoadJS('estoque/javascript/analise.js');");
        echo utf8_encode($this->getInfoPagina()->toHTML() .
                        $this->inibeBotoes()->toHTML() .
                        $_form->toHTML() . $_sc->toHTML() . $_div->toHTML());
    }

}
?>
```

Note que precisamos de uma classe javascript para auxiliar no processo de análise. Essa classe deve disponibilizar métodos para solicitar ao servidor web a análise do inventário selecionado, bem como para processar a recontagem ou a consolidação do inventário, conforme as opções disponíveis e a escolha do usuário.

Lista 14: analise.js

```
// rotinas javascript para análise do inventário
function analise() {};
analise.prototype = {
run:    function() {
            var i = objSelect.checked('INVENTARIO_CODIGO');
            if(i==='-') {
                alert('Selecione o Inventário desejado para análise');
                document.getElementById('INVENTARIO_CODIGO').focus();
            } else {
                var params='classe=inventario&metodo=analisar' +
                    '&arquivo_classe=estoque/classes/classe_inventario.inc';
                params += '&INVENTARIO_CODIGO='+i;
                ObjProcAjax.runPost('executa_busca_ajax.php5',
                                    'RESULTADO_ANALISE',params);
            }
        },
recontagem:
        function(iv) {
                var params='classe=inventario&metodo=recontar' +
                    '&arquivo_classe=estoque/classes/classe_inventario.inc';
                params += '&INVENTARIO_CODIGO='+iv;
                params += '&ITENS='+ contagem.toJSON();
                ObjProcAjax.runPost('executa_busca_ajax.php5',
                                    'RESULTADO_ANALISE',params);
        },
consolidar:
        function(iv,di) {
            var tme = objSelect.checked('TIPO_MOV_E');
            var tms = objSelect.checked('TIPO_MOV_S');
            if(tme==='-'||tms==='-') {
                alert('Favor selecionar os tipos de movimento ' +
                    'de entradas e saída para ajuste no estoque');
                return false;
            } else {
                var params='classe=inventario&metodo=consolidar'  +
                    '&arquivo_classe=estoque/classes/classe_inventario.inc';
                params += '&INVENTARIO_CODIGO='+iv;
                params += '&ITENS='+ contagem.toJSON();
                params += '&DIVERGENCIAS='+di;
                params += '&TIPO_MOV_E=' + tme;
                params += '&TIPO_MOV_S=' + tms;
                ObjProcAjax.runPost('executa_busca_ajax.php5',
                                    'RESULTADO_ANALISE',params);
```

```
            }
        }
};

var objAnalise = new analise();
var objSelect   = new Select();
```

O processamento da análise do inventário selecionado será executado pela chamada ao método *analisar()* da classe **inventario**. Esse método faz uso do método *buscarContagens()* da classe **inventarioproduto**. De posse da lista devolvida pelo método *buscarContagens()*, o método *analisar()* valida as contagens realizadas, comparando os estoques conforme a regra definida. No final da análise teremos uma lista com os produtos considerados corretos e das divergências encontradas. Com base nas divergências o sistema gera as opções para o usuário.

inventario

```
/**
 * Realiza a Análise do inventário
 * O resultado é uma lista dos itens inventariados
 * e a sua situação que pode ser: "OK" ou "Necessita recontagem"
 * No final da lista exibe as opções
 * - Confirmar o inventário (com divergências ou não)
 * - Gerar recontagem
 *
 * Buscar Itens + Contagens de cada produto+local (contagem atual)
 * Buscar a contagem N-1 (se n>1)
 * Regra basica Contagem N = Contagem N-1
 * Se N=1 => Contagem 1 = Estoque atual
 * Caso contrário gera uma divergência
 *
 */
public function analisar() {
     $this->Buscar(Array("INVENTARIO_CODIGO"=>$_POST['INVENTARIO_CODIGO']));
     $this->proximo();
     if($this->getCampo("INVENTARIO_CONSOLIDADO")->getValor()=='S') {
          $_div = new tag(new tipotag("DIV"),
                          Array(new atributo("STYLE","padding-bottom:5px;")),
                          "Inventário já está consolidado.");
          echo utf8_encode($_div->toHTML());
          return;
     }
     $_ip = new inventarioproduto($this->_conn);
     $_itens = $_ip->buscarContagens($_POST['INVENTARIO_CODIGO'],
                          $this->getCampo("INVENTARIO_CONTAGEM")->getValor());
     // Montar a lista e decidir o que fazer...
     $_divergencias = 0;
     $_trt = new tipotag("TR");
     $_tdt = new tipotag("TD");
     $_tab = new tag(new tipotag('TABLE'),
               Array(new atributo("BORDER",0),
                    new atributo("CELLPADDING",1),
                    new atributo("STYLE",
                         "border:1px solid #a0a0a0;border-collapse:collapse;"),
                    new atributo("WIDTH","100%"),
                    new atributo("ID","TAB_CONTAGEM")));
     $_bd  = new tag(new tipotag("TBODY"));
     $_tr  = new tag($_trt);
     $_tr->addSubTag(new tag(new tipotag('TH'),
```

```
          Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
          "Produto"));
$_tr->addSubTag(new tag(new tipotag('TH'),
          Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
          "Local"));
$_tr->addSubTag(new tag(new tipotag('TH'),
          Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
          "Estoque Congelado"));
if($this->getCampo("INVENTARIO_CONTAGEM")->getValor()>1) {
     $_tr->addSubTag(new tag(new tipotag('TH'),
          Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
          "Contagem " .
          ($this->getCampo("INVENTARIO_CONTAGEM")->getValor()-1)));
}
$_tr->addSubTag(new tag(new tipotag('TH'),
          Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
          "Contagem {$this->getCampo("INVENTARIO_CONTAGEM")->getValor()}"));
$_tr->addSubTag(new tag(new tipotag('TH'),
          Array(new atributo("STYLE",
                    "border:1px solid #a0a0a0;background-color:#f0f0f0;")),
          "Situação"));
$_bd->addSubTag($_tr);
$_var = "var contagem=[];";
foreach($_itens as $_item) {
     $_tr = new tag($_trt);
     $_tr->addSubTag(new tag($_tdt,null,
               $_item['PRODUTO_CODIGO'] . "-" . $_item["PRODUTO_DESC"]));
     $_tr->addSubTag(new tag($_tdt,null,
               $_item['LOCAL_CODIGO'] . "-" . $_item["LOCAL_DESC"]));
     $_tr->addSubTag(new tag($_tdt,
               Array(new atributo("STYLE",
                         "padding-right:5px;text-align:RIGHT")),
               number_format($_item['PRODUTO_ESTOQUE'],3,",","."")));
     if($this->getCampo("INVENTARIO_CONTAGEM")->getValor()>1) {
          $_tr->addSubTag(new tag($_tdt,
               Array(new atributo("STYLE",
                         "padding-right:5px;text-align:RIGHT")),
               ($_item['CONTAGEM_ANTERIOR']===null
                    ? "-"
                    : number_format($_item['CONTAGEM_ANTERIOR'],3,",","."))));
     }
     $_tr->addSubTag(new tag($_tdt,
               Array(new atributo("STYLE",
                         "padding-right:5px;text-align:RIGHT")),
               ($_item['CONTAGEM_ESTOQUE']===null
                    ? "-"
                    : number_format($_item['CONTAGEM_ESTOQUE'],3,",","."))));
     $_parametro_1 = $_item['CONTAGEM_ESTOQUE'];
     $_parametro_2 = ($this->getCampo("INVENTARIO_CONTAGEM")->getValor()>1
                         ? $_item['CONTAGEM_ANTERIOR']
                         : $_item['PRODUTO_ESTOQUE']);
     $_var .= "contagem[contagem.length]=[" .
               "'{$_item['PRODUTO_CODIGO']}','{$_item['LOCAL_CODIGO']}'," .
               number_format($_item['CONTAGEM_ESTOQUE'],3,".","") . ",";
     if($_parametro_1===null)  {
          ++$_divergencias;
          $_tr->addSubTag(new tag($_tdt,
               Array(new atributo("STYLE","color:red")),"Não contado"));
          $_var .= "'nok'";
```

```
        } elseif($_parametro_1!=$_parametro_2) {
            ++$_divergencias;
            $_tr->addSubTag(new tag($_tdt,
                Array(new atributo("STYLE","color:red")),
                "Quantidades Divergentes"));
            $_var .= "'nok'";
        } else {
            $_tr->addSubTag(new tag($_tdt,
                Array(new atributo("STYLE","color:blue")),"OK"));
            $_var .= "'ok'";
        }
        $_var .= "];";
        $_bd->addSubTag($_tr);
}
$_tab->addSubTag($_bd);
$_divi = new tag(new tipotag("DIV"),
                Array(new atributo("STYLE","padding-bottom:5px;")),
                "Análise do Inventário {$_POST['INVENTARIO_CODIGO']}-" .
                "{$this->getCampo("INVENTARIO_DESC")->toHTML()} , " .
                "Contagem {$this->getCampo("INVENTARIO_CONTAGEM")->toHTML()}");
$_divd = new tag(new tipotag("DIV"),
                Array(new atributo("STYLE",
                                "padding-top:5px;padding-bottom:10px;")),
                "{$_divergencias} divergencia(s) encontrada(s)");
$_sc = new tag(new tipotag("SCRIPT"),null,$_var);
echo utf8_encode($_divi->toHTML() . $_tab->toHTML() .
                    $_divd->toHTML() . $_sc->toHTML());
// Opcões
$_form = new formulario("OPCOES_{$this->_nome_tabela}","#");
$_form->deleteAtributo(4);
$_form->addAtributo(new atributo("ONSUBMIT","return false;"));
$_tab  = new tag(new tipotag("TABLE"),
          Array(new atributo("BORDER",0),
              new atributo("CELLPADDING",2),
              new atributo("CELLSPACING",0),
              new atributo("STYLE",
                   "border:1px solid Navy;padding:15px;width:100%;" .
                   "border-collapse:collapse;")));
$_tr = new tipotag("TR");
$_td = new tipotag("TD");
// Recontagem??
$_tab->addSubTag(new tag($_tr));
$_tab->getLastSubTag()->addSubTag(new tag($_td,
         Array(new atributo("COLSPAN",2),
              new atributo("STYLE","color:white;background-color:Navy;" .
                              "border: 1px solid Navy;vertical-align:top;")),
         "Opções para este Inventário"));
$_tab->addSubTag(new tag($_tr));
if($_divergencias>0) {
     $_tab->getLastSubTag()->addSubTag(new tag($_td,
          Array(new atributo("ID","BTNOK"),
               new atributo("STYLE",
               "padding:20px;border: 1px solid Navy;vertical-align:top;"))));
     $_ev = new Eventos();
     $_ev->setOnclick("objAnalise.recontagem(" .
                         "'{$_POST['INVENTARIO_CODIGO']}')");
     $_tab->getLastSubTag()->getLastSubTag()->addSubTag(
          new formInputSubmit("OK"," Gerar Recontagem Número " .
                    ($this->getCampo("INVENTARIO_CONTAGEM")->getValor()+1),
                    null,$_ev));
     $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
          new tipotag("BR",false)));
     $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
          new tipotag("BR",false)));
```

```
    $_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
         new tipotag("SPAN"),
         Array(new atributo("STYLE","color:blue")),
         " Esta opção confirma a contagem dos itens<br> considerados " .
         "corretos(marcados como OK)<br>" .
         " e gera uma nova contagem para os itens divergentes.<br>" .
         " Os Ajustes de estoque não são gerados."));
}
$_tab->getLastSubTag()->addSubTag(new tag($_td,
         Array(new atributo("ID","BTNOK"),
              new atributo("STYLE",
              "padding:20px;border: 1px solid Navy;vertical-align:top;"))));
$_ev = new Eventos();
$_ev->setOnclick("objAnalise.consolidar(" .
                  "'{$_POST['INVENTARIO_CODIGO']}','{$_divergencias}')");
$_tab->getLastSubTag()->getLastSubTag()->addSubTag(
         new formInputSubmit("OK"," Confirmar o Inventário ",null,$_ev));
$_tm = new tipomovimento($this->_conn);
$_ent = array_merge(Array(
                        Array(
                             "valor"=>"-",
                             "label"=>"Selecione o Tipo de Movimento")
                        ),$_tm->retornaLista('E'));
$_sai = array_merge(Array(
                        Array(
                             "valor"=>"-",
                             "label"=>"Selecione o Tipo de Movimento")
                        ),$_tm->retornaLista('S'));
$_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
         new tipotag("BR",false)));
$_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
         new tipotag("DIV"),
         Array(new atributo("STYLE",
                   "padding-top:5px;color:Navy;font-size:12px;")),
         "Tipo Movimento Entrada: "));
$_tab->getLastSubTag()->getLastSubTag()->getLastSubTag()->addSubTag(
         new formSelect("TIPO_MOV_E",false,$_ent));
$_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
         new tipotag("DIV"),
         Array(new atributo("STYLE",
                        "padding-top:5px;color:Navy;font-size:12px;")),
         "Tipo Movimento Saída: "));
$_tab->getLastSubTag()->getLastSubTag()->getLastSubTag()->addSubTag(
         new formSelect("TIPO_MOV_S",false,$_sai));
$_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
         new tipotag("BR",false)));
$_tab->getLastSubTag()->getLastSubTag()->addSubTag(new tag(
         new tipotag("SPAN"),
         Array(new atributo("STYLE","color:" .
                        ($_divergencias==0
                             ? "blue"
                             : "red"))),
         ($_divergencias==0
              ?    " Confirma o Inventário Atual gerando os ajustes " .
                   "de estoque<br>" .
                   " Necessários, para isso deve-se escolher os tipos " .
                   "de movimentos<br>" .
                   " para ajustes de entrada e de saída."
              :    " Confirma o Inventário no modo forçado, ignorando<br>" .
                   " as divergências encontradas. O sistema considera a<br>" .
                   " última contagem como base para o ajuste de estoque." .
                   "<br> Produtos não contados serão considerados com " .
                   "saldo zero.<br>" .
                   " *** Esta não é a opção mais indicada ***"
```

```
            )));

    $_form->addSubTag($_tab);
    echo utf8_encode($_form->toHTML());
}
```

inventarioproduto

```
/**
 * Busca as contagens do estoque
 *
 * @param string $_inventario
 * @param integer $_contagem
 * @return array
 */
public function buscarContagens($_inventario,$_contagem) {
    $_strSQL = "SELECT I.PRODUTO_CODIGO,P.PRODUTO_DESC," .
               "I.LOCAL_CODIGO,L.LOCAL_DESC," .
               "I.PRODUTO_ESTOQUE,C.CONTAGEM_ESTOQUE";
    if($_contagem>1) {
        $_strSQL .= ",CC.CONTAGEM_ESTOQUE AS CONTAGEM_ANTERIOR";
    }
    $_strSQL .= " FROM {$this->_nome_tabela} I " .
               "LEFT JOIN CONTAGEM C ON I.INVENTARIO_CODIGO=C.INVENTARIO_CODIGO " .
               "AND I.PRODUTO_CODIGO=C.PRODUTO_CODIGO AND " .
               "I.LOCAL_CODIGO=C.LOCAL_CODIGO AND " .
               "C.CONTAGEM_NUMERO={$_contagem} " .
               "LEFT JOIN PRODUTO P ON I.PRODUTO_CODIGO=P.PRODUTO_CODIGO " .
               "LEFT JOIN LOCALESTOQUE L ON I.LOCAL_CODIGO=L.LOCAL_CODIGO ";
    if($_contagem>1) {
        $_strSQL .= "LEFT JOIN CONTAGEM CC " .
                    "ON I.INVENTARIO_CODIGO=CC.INVENTARIO_CODIGO AND " .
                    "I.PRODUTO_CODIGO=CC.PRODUTO_CODIGO AND " .
                    "I.LOCAL_CODIGO=CC.LOCAL_CODIGO AND " .
                    "CC.CONTAGEM_NUMERO=" . ($_contagem-1) . " ";
    }
    $_strSQL .=    "WHERE I.INVENTARIO_CODIGO='{$_inventario}' AND " .
                   "I.INVENTARIO_ESTOQUE IS NULL";
    $_res = Array();
    if($this->_conn->executaSQL($_strSQL)!==false&&
       $this->_conn->getNumRows()>0) {
        while($_dados=$this->_conn->proximo()) {
            $_res[] = $_dados;
        }
    }
    return $_res;
}
```

A Figura 9.13 mostra o resultado da análise de um inventário.

Caso a opção do usuário seja a de executar a recontagem, o sistema grava os inventários considerados corretos marcando o campo INVENTARIO_ESTOQUE com a quantidade contada, limpa o campo INVENTARIO_ESTOQUE (atribui o valor nulo) para os produtos com divergência e em seguida incrementa o campo INVENTARIO_CONTAGEM. Com isso será possível a emissão das fichas de contagem para os produtos e locais divergentes.

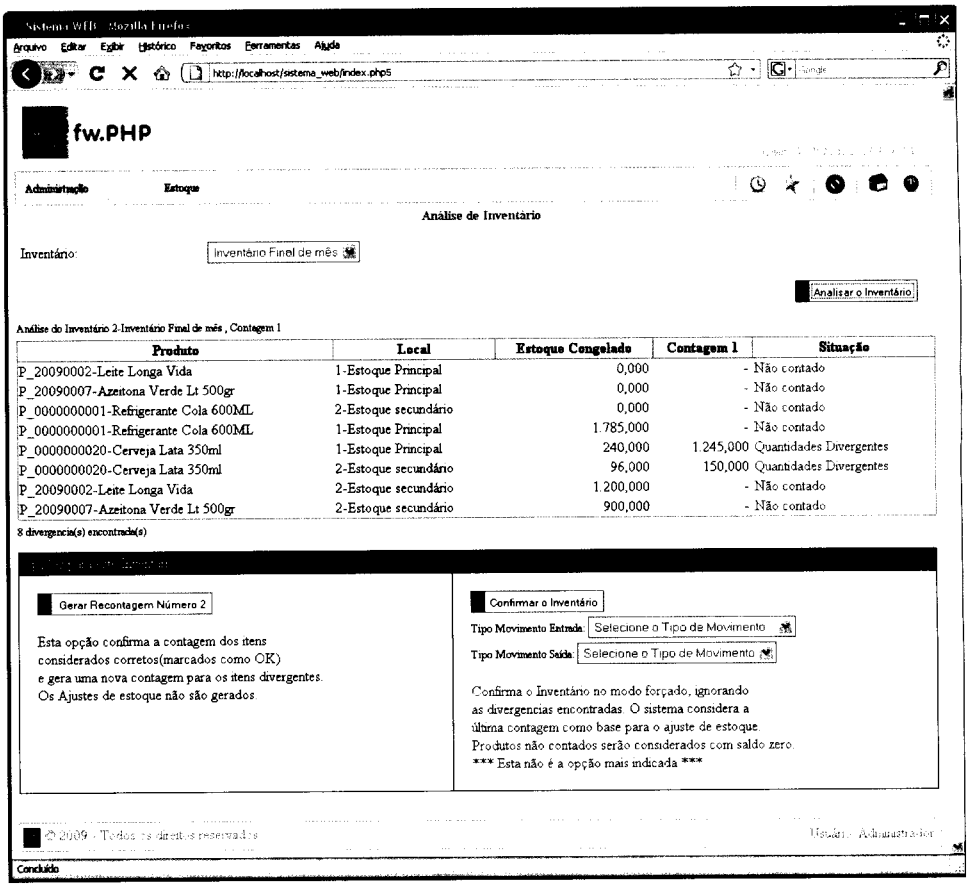


Figura 9.13

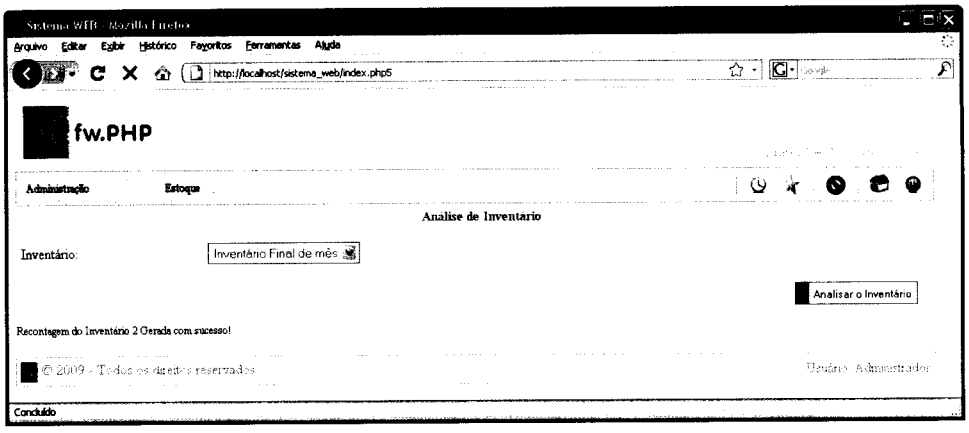


Figura 9.14

Uma vez que a recontagem foi gerada, o usuário pode imprimir novas fichas de contagem, mas desta vez somente com os produtos e locais inconsistentes.

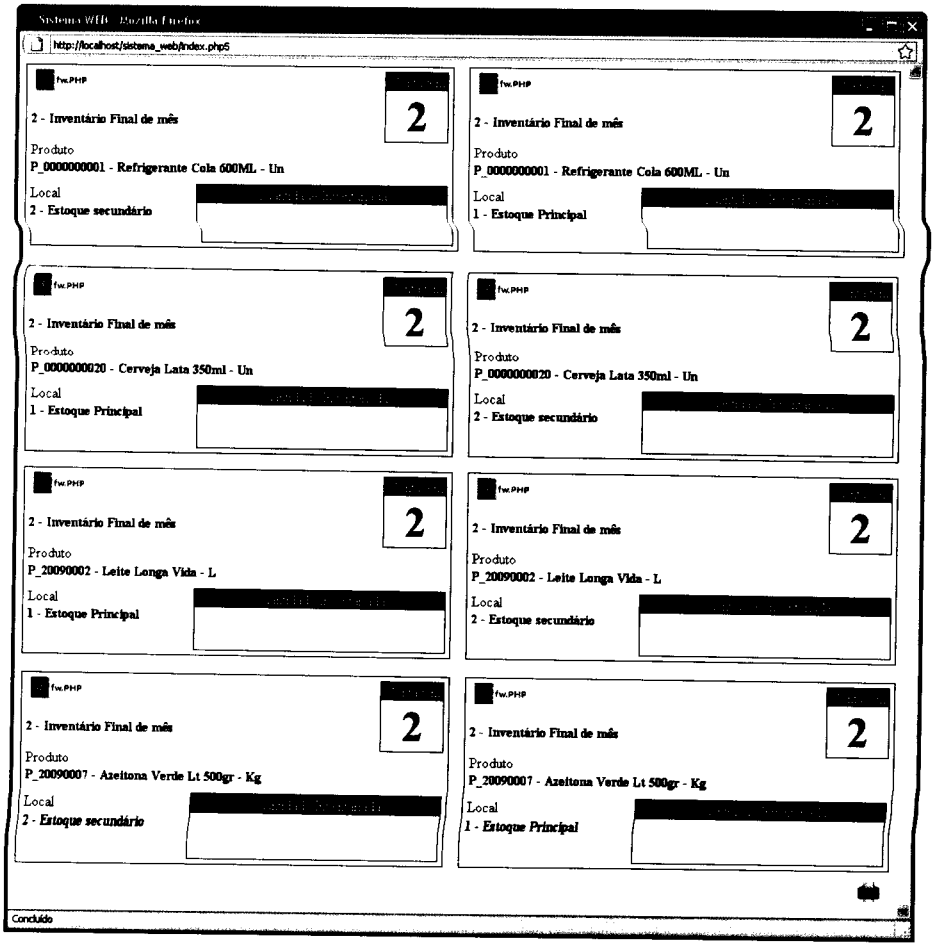


Figura 9.15

Quando não houver mais divergências, ou caso o usuário opte por consolidar o inventário, ignorando as divergências apontadas, o sistema atualiza o campo INVENTARIO_ESTOQUE com o valor da última contagem realizada (somente para os produtos e locais que participaram da última contagem; os produtos confirmados anteriormente continuam como estavam).

Em seguida o sistema realiza os ajustes necessários no estoque, gerando a movimentação para igualar o estoque do produto e local ao estoque inventariado.

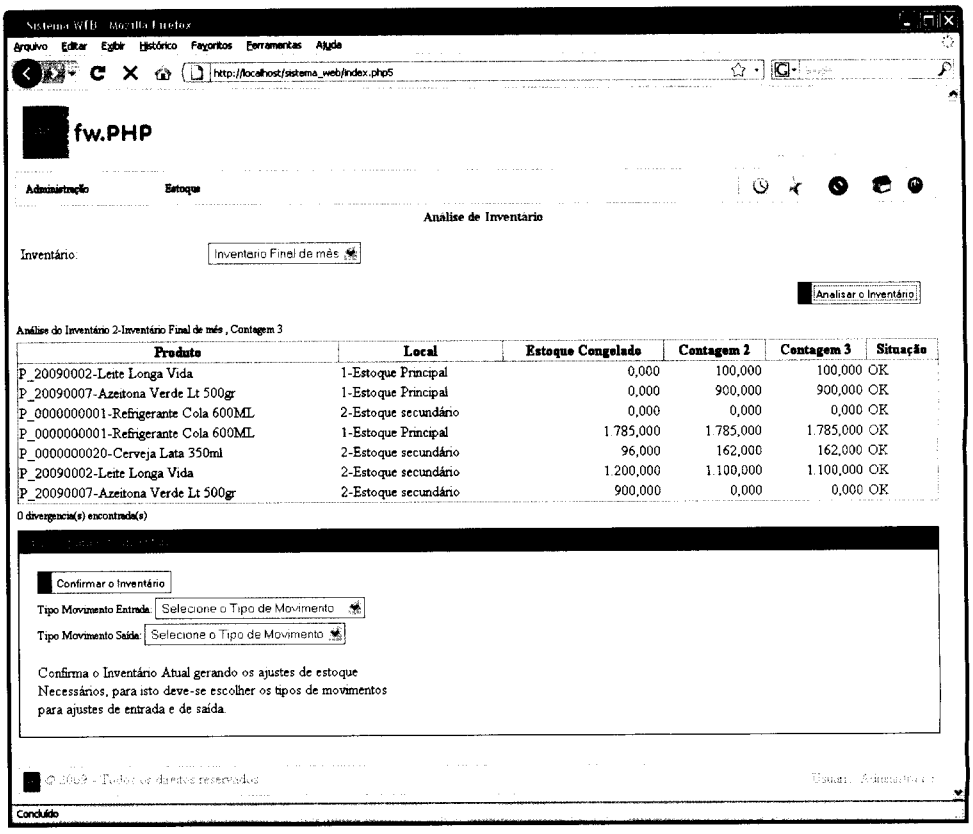


Figura 9.16

Para efetivar o inventário, o sistema executa o método *consolidar()* da classe **inventario**. Esse método instancia a classe **inventarioproduto** e marca os itens que foram considerados corretos na análise do inventário (método *marcaitensok()*), em seguida executa o método *consolidar()* dessa classe. Se os ajustes forem executados corretamente, o inventário é encerrado pelo sistema, ou seja, o campo denominado INVENTARIO_CONSOLIDADO é marcado com o valor 'S'. Além disso o método atualiza o campo INVENTARIO_DATA_REAL com a data do dia.

```
/**
 * Consolida o inventário
 * Executando o lançamento no estoque
 * para ajustes
 *
 */
public function consolidar() {
     $_itens = json_decode($_POST['ITENS']);
     $_ok = Array();
     foreach($_itens as $_item) {
          $_ok[] = "SET INVENTARIO_ESTOQUE=" . number_format($_item[2],3,".","") .
" WHERE " .
                         "(INVENTARIO_CODIGO='{$_POST['INVENTARIO_CODIGO']}'    AND
PRODUTO_CODIGO='{$_item[0]}' AND LOCAL_CODIGO='{$_item[1]}')";
```

```
        }
        $this->_conn->startTransaction();
        $_ip = new inventarioproduto($this->_conn);
        if($_ip->marcaitensok($_ok)===false) {
            $this->_conn->ROLLBACK();
            $_div = new tag(new tipotag("DIV"),
                    Array(new atributo("STYLE","COLOR:red;")),
                        "Erro ao processar o Inventário (marcar itens) - " .
                        "{$this->_conn->getUltimoErro()}!");
        } else {
            // Agora geramos os movimentos de ajuste
            if($_ip->consolidar($_POST['INVENTARIO_CODIGO'],
                            $_POST['TIPO_MOV_E'],$_POST['TIPO_MOV_S'])===false) {
                $this->_conn->ROLLBACK();
                $_div = new tag(new tipotag("DIV"),
                        Array(new atributo("STYLE","COLOR:red;")),
                            "Erro ao Consolidar o Inventário (itens) - " .
                            "{$this->_conn->getUltimoErro()}!");
            } else {
                $this->getCampo("INVENTARIO_DATA_REAL")->setValor(date("d-m-Y"));
                if($this->_conn->executaSQL("UPDATE {$this->_nome_tabela} " .
                            "SET INVENTARIO_CONSOLIDADO='S'," .
                            "INVENTARIO_DATA_REAL=" .
                            "{$this->getCampo("INVENTARIO_DATA_REAL")->toBD()} " .
                            "WHERE INVENTARIO_CODIGO=" .
                            "'{$_POST['INVENTARIO_CODIGO']}'"
                            )===false) {
                    $this->_conn->ROLLBACK();
                    $_div = new tag(new tipotag("DIV"),
                            Array(new atributo("STYLE","COLOR:red;")),
                                "Erro ao Consolidar o Inventário - " .
                                "{$this->_conn->getUltimoErro()} !");
                } else {
                    $this->_conn->commit();
                    $_div = new tag(new tipotag("DIV"),
                            Array(new atributo("STYLE","COLOR:blue;")),
                            "Inventário Consolidado!");
                }
            }
        }
        echo utf8_encode($_div->toHTML());
}
```

O método *consolidar()* da classe **inventarioproduto** busca todos os registros relacionados ao inventário informado e lança os ajustes de estoque, utilizando os tipos de movimentos selecionados para entrada e saída.

Para lançar os movimentos de ajuste, as classes **entradas** e **saídas** são instanciadas, e conforme a necessidade do ajuste uma ou outra é utilizada.

```
/**
 * Ajusta os valores-padrão para o movimento
 *
 * @param historicomovimento $_his
 */
protected function ajustaHistorico(historicomovimento &$_his) {
    $_his->getCampo("PRODUTO_CODIGO")->setValor(
                $this->getCampo("PRODUTO_CODIGO")->getValor());
    $_his->getCampo("LOCAL_CODIGO")->setValor(
                $this->getCampo("LOCAL_CODIGO")->getValor());
    $_his->getCampo("HISTORICO_DOCUMENTO")->setValor(
```

```
                    'INV_' . $this->getCampo("INVENTARIO_CODIGO")->getValor());
        $_his->getCampo("HISTORICO_DATA")->setValor(date("d-m-Y"));
        $_his->getCampo("HISTORICO_VALOR_UNIT")->setValor(
                    $this->getCampo("PRODUTO_CUSTOMEDIO")->toHTML());
        $_his->getCampo("HISTORICO_HISTORICO")->setValor("Ajuste de Inventário");
}

/**
 * Gera os movimentos de ajuste de estoque para
 * o inventário informado
 *
 * @param string $_iv
 * @param string $_tme
 * @param string $_tms
 * @return boolean
 */
public function consolidar($_iv,$_tme,$_tms) {
        $_ent = new entradas($this->_conn);
        $_sai = new saidas($this->_conn);
        if($this->_conn->executaSQL("SELECT * FROM {$this->_nome_tabela} WHERE " .
                "INVENTARIO_CODIGO='{$_iv}'")!==false&&
           $this->_conn->getNumRows()>0) {
             while($_dados=$this->_conn->proximo()) {
                  $_itens[] = $_dados;
             }
             foreach($_itens as $_campos) {
                  foreach($_campos as $_nome=>$_vlr) {
                       $this->getCampo($_nome)->setValor($_vlr);
                  }
                  $_div = $this->getCampo("PRODUTO_ESTOQUE")->getValor() -
                          $this->getCampo("INVENTARIO_ESTOQUE")->getValor();
                  if($_div<0) {
                       // Entrada no estoque
                       $this->ajustaHistorico($_ent);
                       $_div *=-1;
                       $_ent->getCampo("HISTORICO_VALOR_TOTAL")->setValor(
                            $this->getCampo("PRODUTO_CUSTOMEDIO")->getValor()*$_div);
                       $_ent->getCampo("TIPOMOV_CODIGO")->setValor($_tme);
                       $_ent->getCampo("HISTORICO_QUANTIDADE")->setValor($_div);
                       if($_ent->incluir()===false) {
                            return false; //temos um erro aqui
                       }
                  } elseif($_div>0) {
                       // Saída
                       $this->ajustaHistorico($_sai);
                       $_sai->getCampo("CUSTO_SAIDA")->setValor(
                            $this->getCampo("PRODUTO_CUSTOMEDIO")->toHTML());
                       $_sai->getCampo("TIPOMOV_CODIGO")->setValor($_tms);
                       $_sai->getCampo("HISTORICO_QUANTIDADE")->setValor($_div);
                       if($_sai->incluir()===false) {
                            return false;
                       }
                  }
             }
        } else {
             return false;
        }
        return true;
}
```

O método *ajustaHistorico()* é utilizado apenas para evitar a repetição dos comandos de preenchimento dos campos comuns na classe **historicomovimento**.

A emissão do relatório Razão de estoque mostra como ficou o estoque após a consolidação do inventário. A Figura 9.17 apresenta como exemplo o produto 'Cerveja Lata 350 ml' que teve o estoque principal zerado e o estoque secundário aumentado para 162 unidades, o que implica no lançamento de uma saída no local 'Estoque principal' e um lançamento de entrada no local 'Estoque secundário', fazendo com que o saldo final do produto passe a ser 162 unidades.

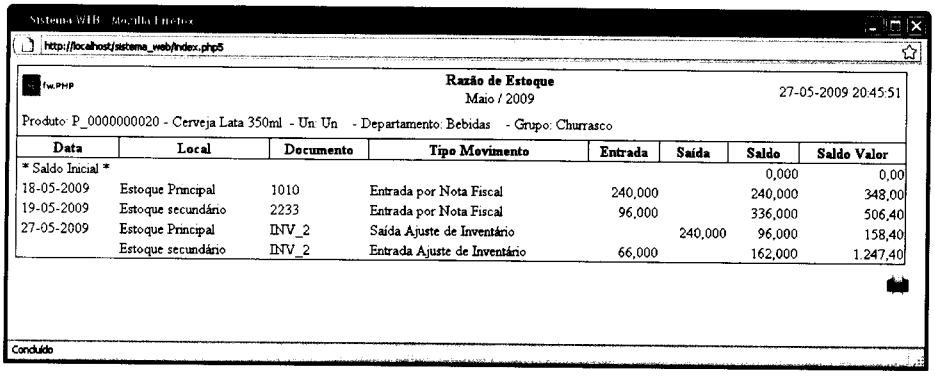


Figura 9.17

Com isso encerramos o desenvolvimento do sistema de controle de estoque.

Referências bibliográficas

AMBLER, S. W. **The Object Primer:** *The Application Developer's Guide to Object Orientation and the UML*. Cambridge University Press, 2001.

SOARES, W. **PHP 5:** *Conceitos, Programação e Integração com Banco de Dados*. São Paulo: Érica, 2008.

_____. **Ajax:** *Guia prático para Windows*. São Paulo: Érica, 2006.

_____. **Crie um Framework para Sistemas Web com PHP 5 e AJAX**. São Paulo: Érica, 2009.

Sites

http://www.php.net	Site oficial PHP
http://www.zend.com	Site oficial Zend Technologies
http://www.apache.org	Site oficial Apache Foundation
http://www.postgresql.org	Site oficial PostgreSQL
http://www.mysql.com	Site oficial MySQL AB
http://www.w3c.org	Site oficial W3C
http://www.leigeber.com	Site com vários códigos javascript
http://script.aculo.us	Site oficial do script.aculo.us
http://www.prototypejs.org	Site oficial do prototype

Marcas registradas

PHP é marca registrada de PHP Group.

MySQL é marca registrada de MySQL AB.

PostgreSQL é marca registrada de PostgreSQL Inc.

Apache é marca registrada de Apache Software Foundation.

Todos os demais nomes registrados, marcas registradas ou direitos de uso citados neste livro pertencem aos respectivos proprietários.

Índice remissivo

Crie um Sistema Web com

PHP 5 e AJAX

CONTROLE DE ESTOQUE

O controle de estoque é um processo fundamental para qualquer empresa do ramo de produção e venda de produtos, pois contribui para uma gestão eficiente e com qualidade.

Pautada na tecnologia, esta obra ensina como criar um sistema de controle de estoque voltado para o ambiente web e apoiado por um framework de desenvolvimento.

Aborda noções básicas do framework de desenvolvimento fw.PHP, expandindo-o com as funcionalidades de apoio para o sistema proposto. Conceitua sistema de gerenciamento de estoques, bem como descreve a construção das tabelas no banco de dados e dos módulos de cadastro, entrada e saída de estoque, estorno de movimentação e um módulo para relatórios operacionais e gerenciais.

Apresenta um módulo completo de inventário com as rotinas necessárias à correta execução desse processo. Para elaboração do sistema o livro utiliza PHP 5, javascript e AJAX, tendo a orientação a objetos como premissa.

www.editoraerica.com.br

Código: 2403
ISBN: 978-85-365-0240-3