Criando Relatórios com PHP

Pablo Dall'Oglio

Novatec

CAPÍTULO 1

Introdução ao PHP

> "Persistir na raiva é como apanhar um pedaço de carvão quente com a intenção de o atirar em alguém. É sempre quem levanta a pedra que se queima."
>
> Buda

O presente capítulo realiza uma introdução sobre as diversas funções, comandos e estruturas de controle básicos da linguagem PHP, que são possivelmente utilizados ao longo do livro. Conheceremos as estruturas básicas da linguagem, suas variáveis e seus operadores e também um conjunto de funções para manipulação de arquivos, de arrays, de bancos de dados, dentre outros.

O que é o PHP?

A linguagem de programação PHP, cuja logomarca é apresentada na figura 1.1, foi criada no outono de 1994 por Rasmus Lerdorf. No início era formada por um conjunto de scripts voltados à criação de páginas dinâmicas que Rasmus utilizava para monitorar o acesso ao seu currículo na internet. À medida que essa ferramenta foi crescendo em funcionalidades, Rasmus teve de escrever uma implementação em C, que permitia que as pessoas desenvolvessem de forma muito simples suas aplicações para web. Rasmus nomeou essa versão de PHP/FI (*Personal Home Pages/Forms Interpreter*) e decidiu disponibilizar seu código na web em 1995 para compartilhá-lo com outras pessoas, bem como receber ajuda e correção de bugs.

Figura 1.1 – Logo do PHP.

Em novembro de 1997, foi lançada a segunda versão do PHP. Nesse momento, aproximadamente 50 mil domínios, ou 1% da internet, utilizavam PHP. No mesmo ano, Andi Gutmans e Zeev Suraski, dois estudantes que utilizavam PHP em um projeto acadêmico de comércio eletrônico, resolveram cooperar com Rasmus para aprimorar o PHP. Para tal, reescreverem todo o código, com base no PHP/FI 2, nascendo assim o

Estruturas de controle

IF

O IF é uma estrutura de controle que introduz um desvio condicional, ou seja, um desvio na execução natural do programa. Caso a condição dada pela expressão seja satisfeita, então serão executadas as instruções do bloco de comandos. Caso a condição não seja satisfeita, o bloco de comandos é simplesmente ignorado. O comando IF pode ser lido como "SE (condição) ENTÃO { comandos... }".

ELSE: É utilizado para indicar um novo bloco de comandos delimitado por { }, caso a condição do IF não seja satisfeita. Pode ser lido como "caso contrário". A utilização do ELSE é opcional.

Na figura 1.2, veja um fluxograma explicando o funcionamento do comando IF. Caso a avaliação da expressão seja verdadeira, o programa parte para a execução de um bloco de comandos, caso seja falsa, parte para a execução do bloco de comandos dada pelo ELSE.

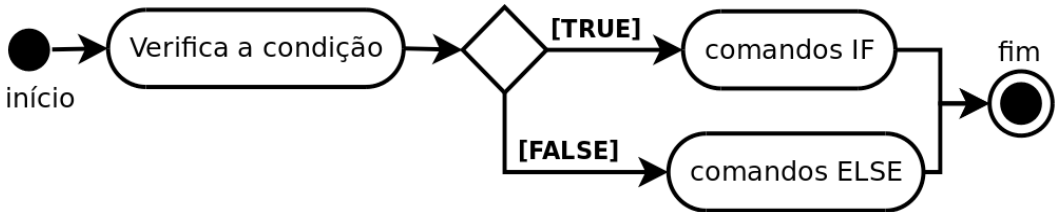


Figura 1.2 – Fluxo do comando IF.

```
if (condição)
{
    comandos se expressão for verdadeira;
}
else
{
    comandos se expressão for falsa;
}
```

Exemplo:

```
<?php
$a = 1;
if ($a == 5) {
    echo "é igual";
}
else {
    echo "não é igual";
}
?>
```

Exemplo:

```
<?php
$txt = "O Rato roeu a roupa do rei de roma";
print str_replace('Rato', 'Leão', $txt);
?>
```

Resultado:

```
O Leão roeu a roupa do rei de roma
```

Manipulação de arrays

A manipulação de arrays no PHP é, sem dúvida, um dos recursos mais poderosos da linguagem. O programador que assimilar bem essa parte, terá muito mais produtividade no seu dia a dia. Isso porque os arrays no PHP servem como verdadeiros contêineres, servindo para armazenar números, strings, objetos, entre outros, de forma dinâmica. Além disso, o PHP nos oferece uma gama enorme de funções para manipulá-los, vistas a seguir.

Criando um array

Arrays são acessados por meio de uma posição, como um índice numérico. Para criar um array, podemos utilizar a função `array([chave =>] valor , ... )`.

```
$cores = array('vermelho', 'azul', 'verde', 'amarelo');
```

ou

```
$cores = array(0=>'vermelho', 1=>'azul', 2=>'verde', 3=>'amarelo');
```

Outra forma de criar um array é simplesmente lhe adicionando valores por meio da seguinte sintaxe:

```
$nomes[] = 'maria';
$nomes[] = 'joão';
$nomes[] = 'carlos';
$nomes[] = 'josé';
```

De qualquer forma, para acessar o array indexado, basta indicar seu índice entre colchetes:

```
echo $cores[0];    // resultado = vermelho
echo $cores[1];    // resultado = azul
echo $cores[2];    // resultado = verde
echo $cores[3];    // resultado = amarelo

echo $nomes[0];    // resultado = maria
echo $nomes[1];    // resultado = joão
```

CAPÍTULO 2

Objetos e bancos de dados

> "Pensar é o trabalho mais difícil que existe. Talvez por isso tão poucos se dediquem a ele."
>
> Henry Ford

Este capítulo abordará a orientação a objetos por meio de exemplos que incluam conceitos básicos como a visibilidade, a herança, entre outros tópicos importantes para a continuidade da leitura do livro. Além da orientação a objetos, este capítulo examinará também o tratamento de exceções e o acesso a bancos de dados de maneira estruturada e também orientada a objetos, utilizando a biblioteca PDO.

Orientação a Objetos

A orientação a objetos é um paradigma de programação que representa uma filosofia para construção de sistemas. No lugar de modelar estruturas de dados e construir um sistema formado por um conjunto de procedimentos, como se fazia em linguagens estruturadas (Cobol, C, Clipper, Pascal), na orientação a objetos modelamos e construímos o sistema utilizando objetos, que são entidades que têm atributos, comportamento, relacionamentos e que representam uma visão de sistema mais próxima do mundo real.

A orientação a objetos é abordada com maior profundidade no livro *PHP Programando com Orientação a Objetos*, do mesmo autor, Novatec Editora.

Introdução

Ao trabalharmos com a orientação a objetos, é fundamental entender o conceito de classes e objetos. Uma classe é uma estrutura estática que define um tipo de dados. Uma classe pode conter atributos (variáveis) e também funções (métodos) para manipular esses atributos. Neste exemplo, declaramos a classe `Produto` com quatro atributos. Os atributos são "variáveis" que existem dentro do contexto de um objeto. Declaramos atributos por meio da palavra chave `public`, entre outras vistas mais adiante.

Herança

A utilização da orientação a objetos e o encapsulamento do código em classes nos orientam em direção a uma maior organização. Mas um dos maiores benefícios que encontramos na utilização desse paradigma é o reuso. A possibilidade de reutilizar partes de código já definidas é o que nos dá maior agilidade no dia a dia, além de eliminar a necessidade de eventuais duplicações ou reescritas de código.

Quando falamos em herança, a primeira imagem que nos aparece na memória é a de uma árvore genealógica com avós, pais, filhos e as características transmitidas geração após geração. O que devemos levar em consideração sobre a herança em orientação a objetos é o compartilhamento de atributos e comportamentos entre as classes de uma mesma hierarquia (árvore). As classes inferiores da hierarquia automaticamente herdam todas as propriedades e métodos das classes superiores, chamadas de superclasses.

Esse recurso tem uma aplicabilidade muito grande, visto que é relativamente comum termos de criar novas funcionalidades em software. Ao utilizarmos a herança, em vez de criarmos uma estrutura totalmente nova (uma classe), podemos reaproveitar uma estrutura existente, que nos forneça uma base abstrata para o desenvolvimento, provendo recursos básicos e comuns.

Já criamos a classe `Conta`, agora podemos aproveitar seu código para criar classes mais específicas como `ContaCorrente` e `ContaPoupanca`, como no diagrama apresentado na figura 2.4.

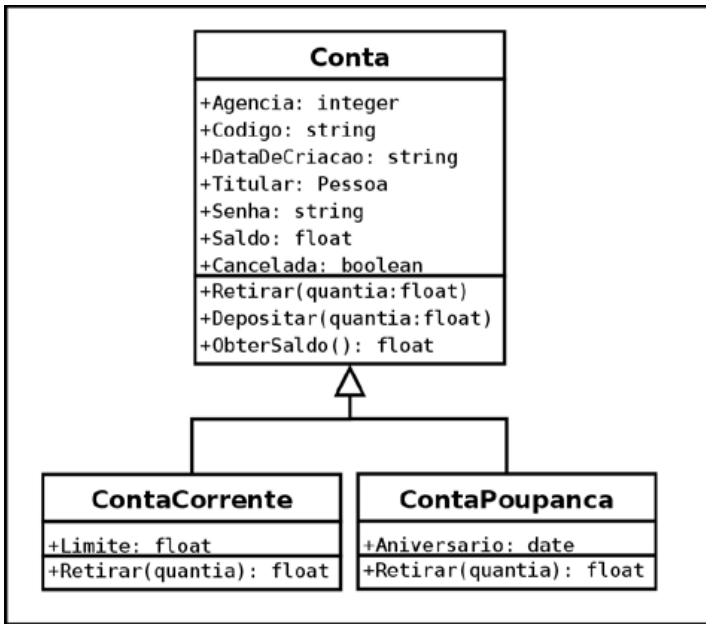


Figura 2.4 – Herança de classes.

PDO PHP Data Objects

O PHP é, em sua maioria, um projeto voluntário cujos colaboradores estão distribuídos geograficamente ao redor de todo o planeta. Como resultado, o PHP evoluiu com base em necessidades individuais para resolver problemas pontuais, movidos por razões diversas. Por um lado, essas colaborações fizeram o PHP crescer rapidamente, por outro, geraram uma fragmentação das exensões de acesso à base de dados, cada qual com sua implementação particular, não havendo real consistência entre as interfaces delas (Oracle, MySQL, PostgreSQL, Sql Server etc.).

Em razão da crescente adoção do PHP, tornou-se necessário unificar o acesso às diferentes extensões de bancos de dados presentes no PHP. Assim surgiu o PDO (PHP Data Objects) cujo objetivo é prover uma API limpa e consistente, unificando a maioria das características presentes nas extensões de acesso a banco de dados.

O PDO não é uma biblioteca completa para abstração do acesso à base de dados, uma vez que não faz a leitura nem a tradução das instruções SQL, adaptando-as aos mais diversos "drivers" de bancos de dados existentes. Simplesmente unifica a chamada de métodos, delegando-os a suas extensões correspondentes, e faz uso do que há de mais recente em termos de orientação a objetos presente no PHP5.

Para conectar em bancos de dados diferentes, a única mudança é na string de conexão. Veja a seguir alguns exemplos nos mais diversos bancos de dados.

Banco	String de Conexão
SQLite	`new PDO('sqlite:teste.db');`
FireBird	`new PDO("firebird:dbname=C:\\base.GDB", "SYSDBA", "masterkey");`
MySQL	`new PDO('mysql:unix_socket=/tmp/mysql.sock;host=localhost;port=3306; dbname=livro', 'user', 'senha');`
Postgres	`new PDO('pgsql:dbname=example;user=user;password=senha;host=localhost');`

Para podermos utilizar o PDO, devemos ter habilitado no `php.ini` as bibliotecas (*drivers*) de acesso ao banco de dados de nossa aplicação. No Linux, habilitamos da seguinte forma:

```
extension=mysql.so
extension=pgsql.so
extension=sqlite.so
extension=pdo_mysql.so
extension=pdo_pgsql.so
extension=pdo_sqlite.so
```

Já no Windows, habilitamos da seguinte forma:

```
extension=php_mysql.dll
extension=php_pgsql.dll
extension=php_sqlite.dll
```

CAPÍTULO 3

Bibliotecas utilizadas

> "A preguiça é a mãe do progresso. Se o homem não tivesse preguiça de caminhar, não teria inventado a roda".
>
> Mario Quintana

Neste capítulo, abordaremos as principais bibliotecas utilizadas ao longo do livro. O objetivo inicial é conhecer as bibliotecas e suas diversas características, sem, no entanto, tratar diretamente da geração de relatórios, que será vista de maneira segmentada nos próximos capítulos. Dentre as bibliotecas utilizadas, veremos a geração de arquivos nos formatos HTML, PDF e RTF, além da geração de gráficos gerenciais.

Geração no formato HTML

HTML (*HyperText Markup Language*) é a principal linguagem para formatação de páginas web. Proposta originalmente no início dos anos de 1990 por Tim Berners-Lee, é utilizada para produzir documentos interpretados pelos navegadores. A linguagem permite criar documentos estruturados por meio de uma estrutura semântica contendo elementos como cabeçalhos, parágrafos, listas, *links*, imagens, entre outros itens. Os elementos são formados por *tags* ou "marcadores" delimitados pelos símbolos "‹" e "›".

Introdução

Se você programa em alguma linguagem para web há algum tempo, deve ter percebido o quanto a exibição de tags HTML deixa o código confuso e pouco legível quando temos de concatenar expressões que mesclam código HTML e variáveis da aplicação PHP.

Ao longo desta seção, criaremos algumas classes voltadas à apresentação de elementos HTML. O objetivo dessas classes é substituir a utilização das *tags* presentes no código HTML por classes e seus métodos, podendo construir uma página HTML por meio de objetos, em que cada um deles representará um elemento gráfico (texto, imagem, tabela), como na figura 3.1, na qual temos uma pequena página contendo alguns elementos, como um parágrafo, um botão, uma imagem, uma tabela, entre outros.

```
        // adiciona uma linha para os dados
        $linha = $tabela->addRow();
        $linha->bgcolor = $bgcolor;

        // adiciona as células
        $linha->addCell($pessoa[0]);
        $linha->addCell($pessoa[1]);
        $linha->addCell($pessoa[2]);
        $x = $linha->addCell($pessoa[3]);
        $x->align = 'right';

        $total += $pessoa[3];
        $i++;
    }

    // instancia uma linha para o totalizador
    $linha = $tabela->addRow();

    // adiciona células
    $celula = $linha->addCell('Total');
    $celula->colspan = 3;

    $celula = $linha->addCell($total);
    $celula->bgcolor = "#a0a0a0";
    $celula->align   = "right";

    // exibe tabela
    $tabela->show();
    ?>
```

Geração no formato PDF

Nesta seção, estudaremos como gerar documentos no formato PDF em PHP. PDF (*Portable Document Format*) é um formato de arquivos criado pela Adobe Systems em 1993, com o objetivo de permitir a representação de documentos de maneira independente de sofware, hardware ou sistema operacional. Um documento PDF contém uma descrição completa de um *layout* de documento fixo, que inclui elementos textuais, fontes, imagens e gráficos 2D. Originalmente proprietário, o formato PDF foi oficialmente publicado como um padrão aberto em 1998 pela ISO (*International Organization for Standardization*). Atualmente qualquer pessoa pode escrever aplicativos que manipulam arquivos nesse formato. O formato PDF é suportado por diversos aplicativos como Adobe Reader e Foxit Reader em Windows e Evince e KPDF em Linux.

procura exibir uma grande quantidade de texto por meio da repetição da string "teste" mil vezes. Dessa forma, serão criadas três páginas do documento, contendo essa string, com o cabeçalho e os rodapés. A figura 3.13 apresenta o resultado do programa.

Figura 3.13 – Manipulação de cabeçalhos e rodapés em PDF.

fpdf5.php

```
<?php
require('app.util/pdf/fpdf.php');

class Documento1 extends FPDF
{
    /**
     * Define o método para imprimir o cabeçalho
     */
    function Header()
    {
        $this->Image('images/banner.jpg', 20, 10);
        $this->Ln(70);
        $this->SetFont('Arial', 'B', 16);
        $this->Cell(520, 20, 'Título do documento', 1, 0, 'C');
        $this->Ln(40);
    }

    /**
     * Define o método para imprimir o rodapé
     */
    function Footer()
    {
        $this->SetY(-15);
        $this->SetFont('Arial','B',8);
        $this->Cell(0,10,'Página '.$this->PageNo().'/{nb}',0,0,'C');
    }
}
$pdf = new Documento1('P', 'pt', 'A4');
```

```
$pdf->AddPage();
$pdf->AliasNbPages();
$pdf->Write(20, str_repeat('teste ', 1000));
$pdf->Output();
?>
```

Geração no formato RTF

Nesta seção, estudaremos como gerar documentos no formato RTF em PHP. RTF (*Rich Text Format*) é um formato de arquivos para representar documentos desenvolvido pela Microsoft Corporation em 1987. O objetivo do formato de arquivos RTF era ser utilizado como formato-padrão de arquivos nos produtos da Microsoft, bem como permitir o intercâmbio de documentos entre diferentes plataformas. Foi utilizado pela primeira vez como parte do Microsoft Word 3. Além de manter o formato, a Microsoft tem os direitos de propriedade intelectual sobre o padrão RTF. Tal formato é suportado pela maioria dos editores de texto, como WordPad e Microsoft Word no Windows, TextEdit no MacOS, além do AbiWord e OpenOffice no Linux.

Introdução

Para gerar documentos no formato RTF em PHP, utilizaremos a biblioteca PHPRtfLite (http://phprtf.sf.net), que é formada por um conjunto de classes com o objetivo de construir documentos no formato RTF. Essas classes abrangem a construção de páginas, seções, cabeçalhos, rodapés, parágrafos, imagens, tabelas, entre outros itens. A biblioteca PHPRtfLite não tem como pré-requisitos extensões ou outras bibliotecas. Dessa forma, pode ser utilizada a partir de qualquer instalação básica PHP. O desenvolvimento dessa biblioteca é relativamente recente (2007), entretanto tem demonstrado ser consistente com a construção de documentos em aplicações comerciais. A biblioteca PHPRtfLite é disponibilizada sob uma licença permissiva (LGPL) que permite seu uso tanto para aplicações livres quanto para aplicações comerciais.

Manipulação básica de texto

O objetivo deste primeiro exemplo sobre a utilização do formato RTF para gerar documentos é demonstrar alguns aspectos simples, como a formatação de textos, fontes, imagens e *links*.

O exemplo inicia com a requisição simples à biblioteca RTF por meio da função `require`. Logo após, utilizamos o método `registerAutoloader()`, que registrará internamente o método da biblioteca `PHPRtfLite` responsável por carregar automaticamente as classes necessárias sempre que estas forem requisitadas pela primeira vez. Depois, instanciamos um objeto da classe `PHPRtfLite`.

Depois de vermos mais este exemplo de utilização do método `writeText()`, utilizaremos o método `addImage()` para adicionar uma imagem ao documento. Esse método recebe a localização da imagem a ser exibida no documento, um objeto para definir a formatação do parágrafo no qual a figura será inserida e mais dois parâmetros para indicar a largura e a altura. Como indicamos somente a largura (3 cm), a altura é calculada de maneira proporcional para manter o aspecto original da figura.

Por último, inserimos um *hyperlink* no documento por meio do método `writeHyperlink()`, que recebe o endereço que desejamos referenciar, o rótulo de texto a ser inserido, um objeto contendo a formatação da fonte e outro contendo a formatação do parágrafo. A figura 3.14 apresenta o resultado do programa.

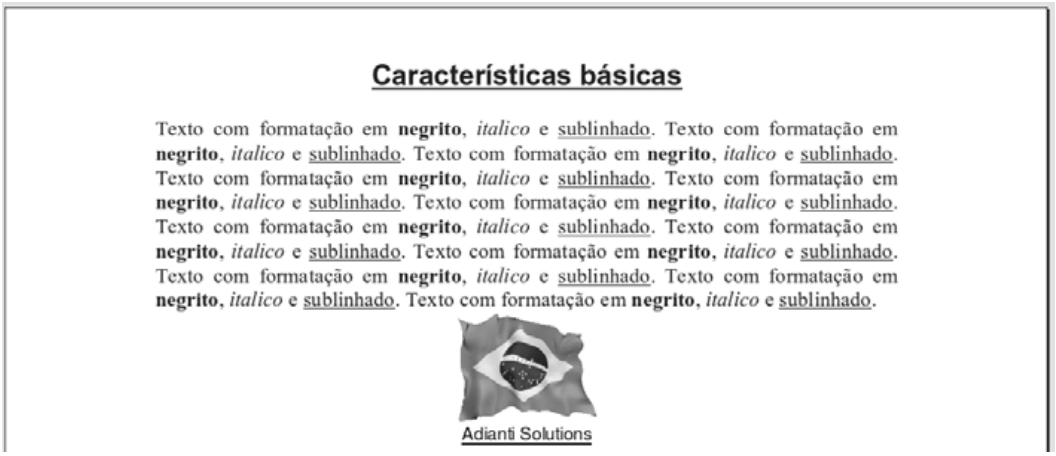


Figura 3.14 – Manipulação básica de texto em RTF.

rtf1.php

```
<?php
// inclui a classe Rtf
require 'app.util/rtf/PHPRtfLite.php';
// registra o class loader
PHPRtfLite::registerAutoloader();

// instancia a classe Rtf
$rtf = new PHPRtfLite;

// adiciona uma seção ao documento
$secao = $rtf->addSection();

// define características de fonte
$fontTitulo = new PHPRtfLite_Font(16, 'Arial');
$fontTitulo->setBold();
$fontTitulo->setUnderline();
```

```
$rtf->setPaperWidth(21);
$rtf->setPaperHeight(16);

$font_head = new PHPRtfLite_Font(12, 'Arial', '#FF0000');
$font_foot = new PHPRtfLite_Font(12, 'Arial', '#58A055');

$header = $rtf->addHeader();
$header->writeText('<b>Cabeçalho do documento</b>', $font_head,
    new PHPRtfLite_ParFormat('center'));
$header->addImage('images/banner.jpg', new PHPRtfLite_ParFormat('center'), 15);

$footer = $rtf->addFooter();
$footer->writeText('<b>Rodapé do documento</b>', $font_foot,
    new PHPRtfLite_ParFormat('center'));
$footer->writeText('<i>Página <pagenum></i>', $font_foot, new PHPRtfLite_ParFormat('center'));

// adiciona uma seção ao documento
$secao = $rtf->addSection();

$secao->writeText(str_repeat('texto qualquer ', 1000), new PHPRtfLite_Font(8, 'Verdana'),
    new PHPRtfLite_ParFormat('justify'));

// envia o RTF ao usuário
$rtf->sendRtf();
?>
```

Geração de gráficos

Nesta seção, abordaremos a criação de gráficos. Gráficos buscam representar de forma visual valores numéricos, facilitando a compreensão pelo ser humano. Existem diversos tipos de gráficos, como colunas, linhas, pizza, dispersão, área, radar, entre outros.

Introdução

Para gerarmos gráficos em PHP, será utilizada a biblioteca JPGraph (*http://www.aditus.nu/jpgraph*). A biblioteca JPGraph é uma biblioteca orientada a objetos, construída completamente em linguagem PHP. A biblioteca JPGraph utiliza de algumas extensões da linguagem, como "`GD`", "`libpng`" ou "`libjpeg`", entre outras. É importante notar que essas dependências geralmente já são parte das instalações-padrão da linguagem PHP nas mais diversas plataformas, em razão de sua popularidade. A biblioteca pode ser utilizada para criar uma grande diversidade de gráficos, como colunas, linhas, pizza, entre outros.

A biblioteca JPGraph é disponibilizada sob uma licença dupla. Para o desenvolvimento de aplicações livres, pode ser utilizada sem custo. Entretanto, para o desenvolvimento de aplicações comerciais, uma licença profissional deve ser adquirida.

`Add()`. Depois, são definidas a cor (`SetColor()`) e a espessura (`SetWeight()`) de cada uma das linhas. Por fim, é definida a palavra que representará cada uma das linhas do gráfico na legenda por meio do método `SetLegend()`.

Ao final, o gráfico é exibido no navegador por meio do método `Stroke()`. A figura 3.20 apresenta o resultado do programa.

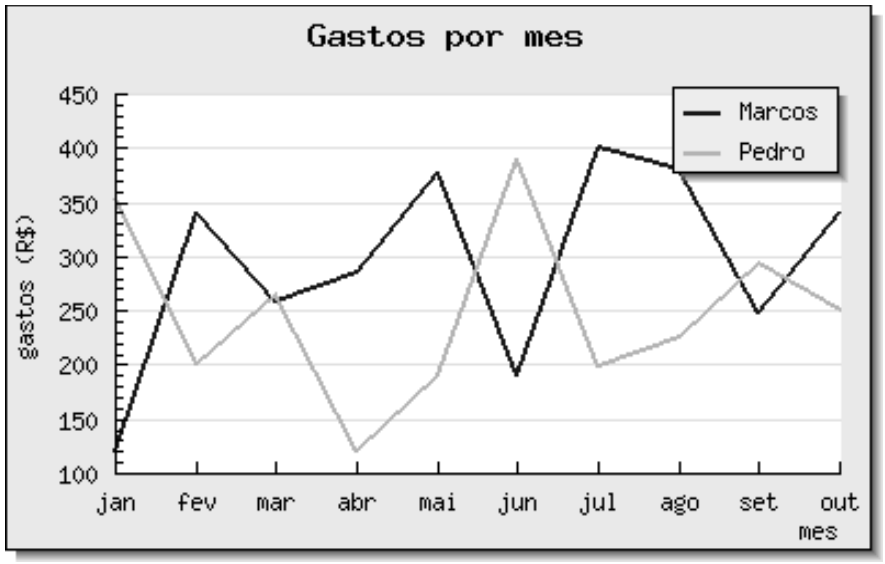


Figura 3.20 – Geração de gráficos de múltiplas linhas.

graph2.php

```
<?php
// inclui a biblioteca base e para gráficos de linha
require_once ('app.util/graph/jpgraph.php');
require_once ('app.util/graph/jpgraph_line.php');

// define os vetores com os dados a serem exibidos
$dados1 = array(120,341,259,285,378,190,401,380,248,340);
$dados2 = array(354,200,265,120,191,391,198,225,293,251);
$rotulos= array('jan', 'fev', 'mar', 'abr', 'mai', 'jun', 'jul', 'ago', 'set', 'out');

// cria o gráfico e define a escala
$graph = new Graph(400,250);
$graph->SetScale("textlin");
// habilita sombreamento na imagem
$graph->SetShadow();
// define as margens
$graph->SetMargin(50,20,40,40);

// define o título do gráfico e dos eixos
$graph->title->Set('Gastos por mes');
```

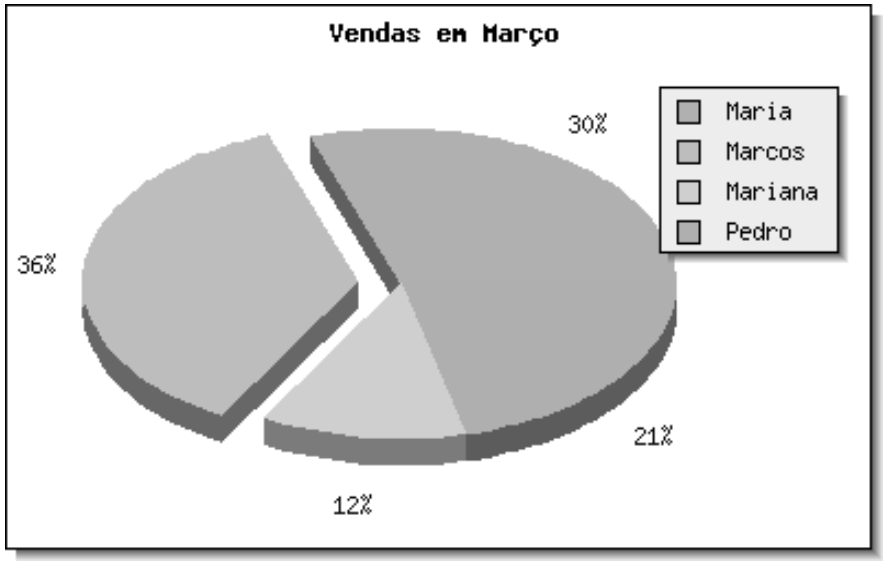


Figura 3.26 – Geração de gráficos de pizza 3D com links.

graph8.php

```
<?php
// inclui a biblioteca base e para gráficos de torta
require_once ('app.util/graph/jpgraph.php');
require_once ('app.util/graph/jpgraph_pie.php');
require_once ('app.util/graph/jpgraph_pie3d.php');

// Define um vetor com os dados a serem exibidos
$dados   = array(100, 120, 40, 70);
$rotulos = array('Maria', 'Marcos', 'Mariana', 'Pedro');

// cria o gráfico
$graph = new PieGraph(400,250);
// habilita sombreamento na imagem
$graph->SetShadow();

// define o título do gráfico
$graph->title->Set("Vendas em Março");
$graph->title->SetFont(FF_FONT1,FS_BOLD);

// cria uma plotagem do tipo torta 3D
$pieplot = new PiePlot3D($dados);

// indica uma fatia para estar em destaque
$pieplot->ExplodeSlice(1);

// posiciona o centro do gráfico
$pieplot->SetCenter(0.45);
```

CAPÍTULO 4

Relatórios tabulares

> "Só existem dois dias no ano em que nada pode ser feito: um se chama ontem e o outro, amanhã. Portanto, hoje é o dia certo para amar, acreditar, fazer e principalmente viver. "
>
> Dalai Lama

Neste capítulo, veremos a construção dos primeiros relatórios envolvendo diretamente a utilização de banco de dados. Neste momento, desenvolveremos relatórios tabulares, que recebem esse nome pelo fato de assemelharem-se ao formato de uma tabela ou planilha. Nos capítulos seguintes, serão desenvolvidos relatórios mais elaborados, com quebras, agrupamentos, totalizações e diferentes níveis hierárquicos.

Base de dados de exemplos

A partir deste capítulo, a maioria dos exemplos desenvolvidos utilizará dados armazenados em um banco de dados. Para facilitar a distribuição e a instalação dos exemplos, optamos por criar uma base de dados em SQLite, que é um banco de dados no formato de arquivos que suporta a linguagem SQL e utiliza a extensão ".db". O driver de acesso a bancos de dados SQLite é integrado ao PHP, ou seja, não é necessária a instalação de software adicional algum para o funcionamento dos exemplos aqui desenvolvidos. Além da base de dados no formato .db, será disponibilizado o script de criação dela com o nome exemplos.sql.

O diagrama apresentado na figura 4.1 representa a estrutura de tabelas utilizadas nos exemplos ao longo do livro. A tabela `pessoa` terá um cadastro de pessoas. Cada pessoa estará relacionada a uma cidade. Esse banco de dados terá registro de vendas realizadas. Cada venda é realizada em uma determinada data (`dt_venda`) para uma pessoa (`id_cliente`), por um vendedor (`id_vendedor`), em uma filial (`id_filial`). Uma venda é composta de itens. A tabela `venda_itens` relaciona quais produtos (`id_produto`), quais quantidades de cada produto (`quantidade`) e quais valores praticados de cada produto (`valor`) integram uma venda. A tabela `produto` registra o cadastro de produtos, contendo

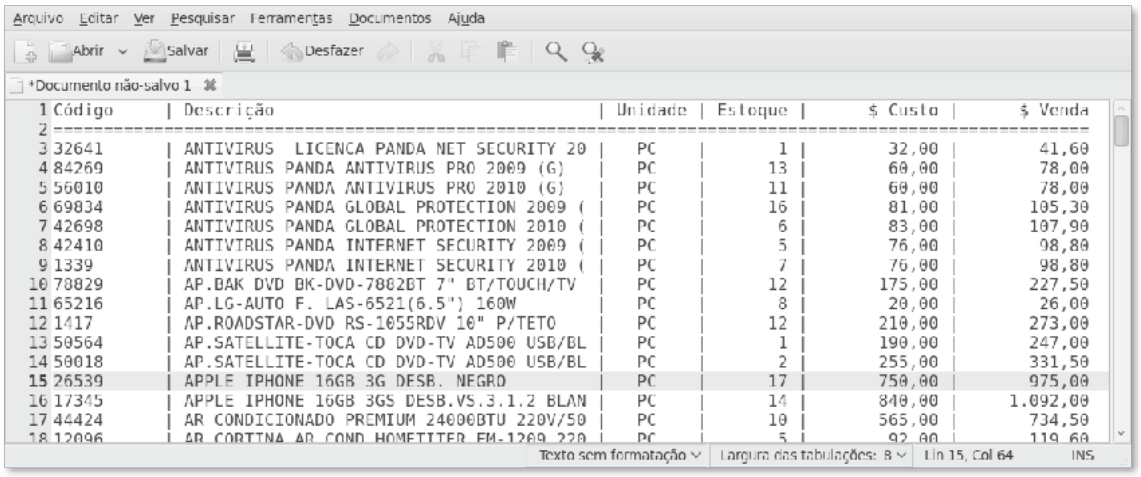
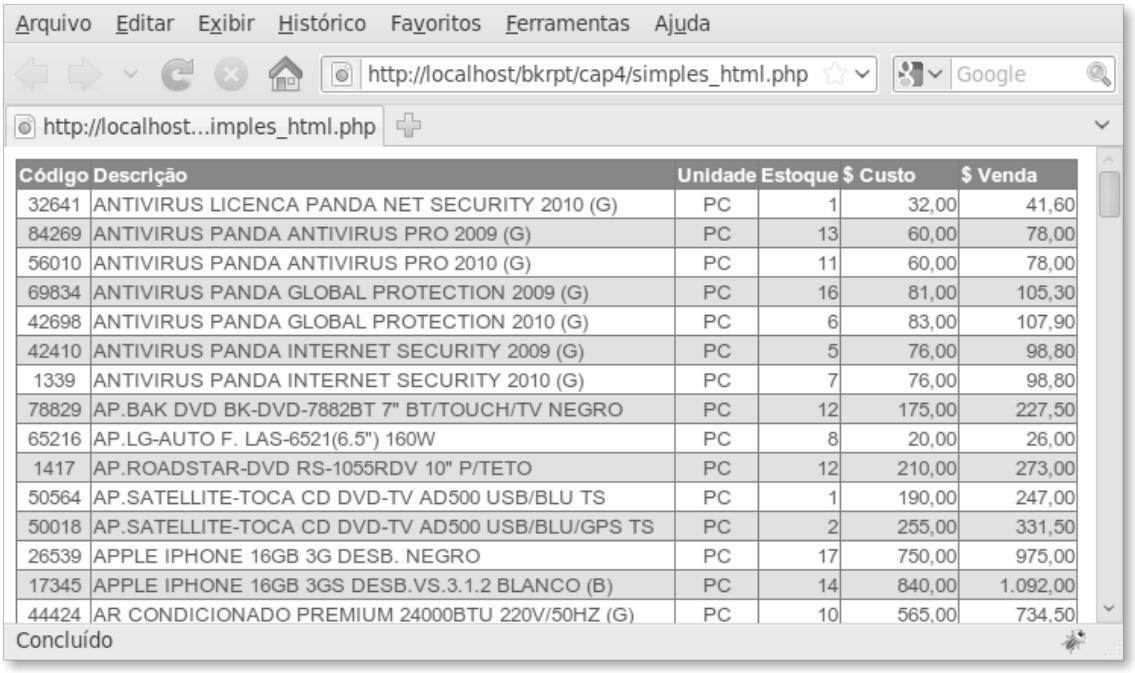


Figura 4.2 – Relatório no formato TXT.

simples_txt.php

```
<?php
/**
 * Carrega uma classe quando ela é necessária,
 * ou seja, quando ela é instancia pela primeira vez.
 */
function __autoload($classe)
{
    if (file_exists("app.ado/{$classe}.class.php"))
    {
        include_once "app.ado/{$classe}.class.php";
    }
}

try
{
    $conn = TConnection::open('exemplos');    // abre conexão com base de dados

    // define a consulta SQL
    $sql = 'SELECT id, descricao, unidade, estoque, preco_custo, preco_venda'.
           ' FROM produto'.
           ' ORDER BY descricao';

    // cria o arquivo TXT
    $handle = fopen('app.output/output.txt', 'w');
    $result = $conn->query($sql);     // executa a instrução SQL

    // monta o cabeçalho do relatório
    $linha = str_pad('Código', 10, ' ', STR_PAD_RIGHT)       . ' | ';
    $linha.= str_pad('Descrição', 40, ' ', STR_PAD_RIGHT) . ' | ';
```



Código	Descrição	Unidade	Estoque	$ Custo	$ Venda
32641	ANTIVIRUS LICENCA PANDA NET SECURITY 2010 (G)	PC	1	32,00	41,60
84269	ANTIVIRUS PANDA ANTIVIRUS PRO 2009 (G)	PC	13	60,00	78,00
56010	ANTIVIRUS PANDA ANTIVIRUS PRO 2010 (G)	PC	11	60,00	78,00
69834	ANTIVIRUS PANDA GLOBAL PROTECTION 2009 (G)	PC	16	81,00	105,30
42698	ANTIVIRUS PANDA GLOBAL PROTECTION 2010 (G)	PC	6	83,00	107,90
42410	ANTIVIRUS PANDA INTERNET SECURITY 2009 (G)	PC	5	76,00	98,80
1339	ANTIVIRUS PANDA INTERNET SECURITY 2010 (G)	PC	7	76,00	98,80
78829	AP.BAK DVD BK-DVD-7882BT 7" BT/TOUCH/TV NEGRO	PC	12	175,00	227,50
65216	AP.LG-AUTO F. LAS-6521(6.5") 160W	PC	8	20,00	26,00
1417	AP.ROADSTAR-DVD RS-1055RDV 10" P/TETO	PC	12	210,00	273,00
50564	AP.SATELLITE-TOCA CD DVD-TV AD500 USB/BLU TS	PC	1	190,00	247,00
50018	AP.SATELLITE-TOCA CD DVD-TV AD500 USB/BLU/GPS TS	PC	2	255,00	331,50
26539	APPLE IPHONE 16GB 3G DESB. NEGRO	PC	17	750,00	975,00
17345	APPLE IPHONE 16GB 3GS DESB.VS.3.1.2 BLANCO (B)	PC	14	840,00	1.092,00
44424	AR CONDICIONADO PREMIUM 24000BTU 220V/50HZ (G)	PC	10	565,00	734,50

Figura 4.3 – Relatório no formato HTML.

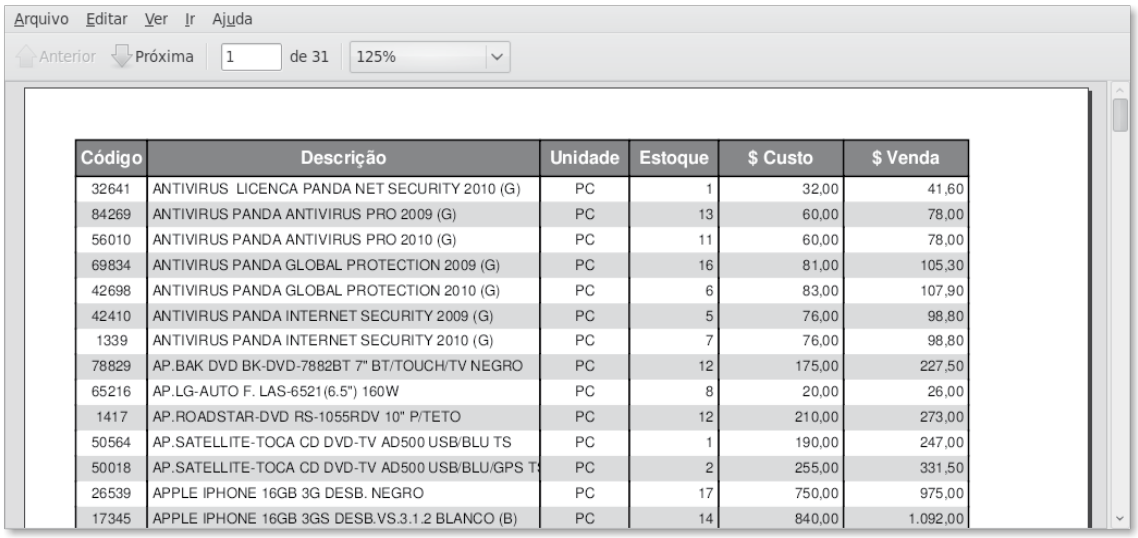
simples_html.php

```
<?php
function __autoload($classe)
{
    ...
}

try
{
    $conn = TConnection::open('exemplos');    // abre uma conexão

    // cria um estilo para o cabeçalho
    $estilo_cabecalho = new TStyle('cabecalho');
    $estilo_cabecalho->font_family = 'arial,verdana,sans-serif';
    $estilo_cabecalho->color = '#ffffff';
        ...
    $estilo_cabecalho->show();

    // cria um estilo para os dados
    $estilo_dados = new TStyle('dados');
    $estilo_dados->font_family = 'arial,verdana,sans-serif';
    $estilo_dados->color = '#2D2D2D';
        ...
    $estilo_dados->show();
```

formatamos as colunas `preco_custo` e `preco_venda` por meio do método `number_format()`. É importante ressaltar que esses cálculos totalizadores devem ser realizados antes da formatação das colunas, uma vez que a formatação descaracteriza os dados, influenciando diretamente os resultados de cálculos matemáticos.

Depois de totalizarmos e formatarmos as variáveis, adicionamos as células contendo os dados ao relatório por meio do método `Cell()`. Utilizamos a variável `$colore` como parâmetro do método `Cell()`. A variável `$colore` indicará se as células exibidas apresentarão cor de fundo ou não. Após a criação das células, a linha é quebrada por meio do método `Ln()` e a variável `$colore` muda de conteúdo (`TRUE`/`FALSE`).

Depois de sairmos do `foreach`, a fonte é redefinida (Arial Bold 10) e adicionamos uma célula de total (Total) que ocupará a largura correspondente a quatro colunas de dados do relatório. Depois, adicionamos células contendo os dados totais (`$total_custo` e `$total_venda`, devidamente formatadas e alinhadas à direita). Ao final, o relatório PDF é armazenado no arquivo *output.pdf* dentro da pasta *app.output* e um *link* é apresentado ao usuário para realizar *download* deste. A figura 4.4 representa o início do relatório gerado.



Código	Descrição	Unidade	Estoque	$ Custo	$ Venda
32641	ANTIVIRUS LICENCA PANDA NET SECURITY 2010 (G)	PC	1	32,00	41,60
84269	ANTIVIRUS PANDA ANTIVIRUS PRO 2009 (G)	PC	13	60,00	78,00
56010	ANTIVIRUS PANDA ANTIVIRUS PRO 2010 (G)	PC	11	60,00	78,00
69834	ANTIVIRUS PANDA GLOBAL PROTECTION 2009 (G)	PC	16	81,00	105,30
42698	ANTIVIRUS PANDA GLOBAL PROTECTION 2010 (G)	PC	6	83,00	107,90
42410	ANTIVIRUS PANDA INTERNET SECURITY 2009 (G)	PC	5	76,00	98,80
1339	ANTIVIRUS PANDA INTERNET SECURITY 2010 (G)	PC	7	76,00	98,80
78829	AP.BAK DVD BK-DVD-7882BT 7" BT/TOUCH/TV NEGRO	PC	12	175,00	227,50
65216	AP.LG-AUTO F. LAS-6521(6.5") 160W	PC	8	20,00	26,00
1417	AP.ROADSTAR-DVD RS-1055RDV 10" P/TETO	PC	12	210,00	273,00
50564	AP.SATELLITE-TOCA CD DVD-TV AD500 USB/BLU TS	PC	1	190,00	247,00
50018	AP.SATELLITE-TOCA CD DVD-TV AD500 USB/BLU/GPS T	PC	2	255,00	331,50
26539	APPLE IPHONE 16GB 3G DESB. NEGRO	PC	17	750,00	975,00
17345	APPLE IPHONE 16GB 3GS DESB.VS.3.1.2 BLANCO (B)	PC	14	840,00	1.092,00

Figura 4.4 – Relatório no formato PDF.

simples_pdf.php

```
<?php
function __autoload($classe)
{
    ...
}
```

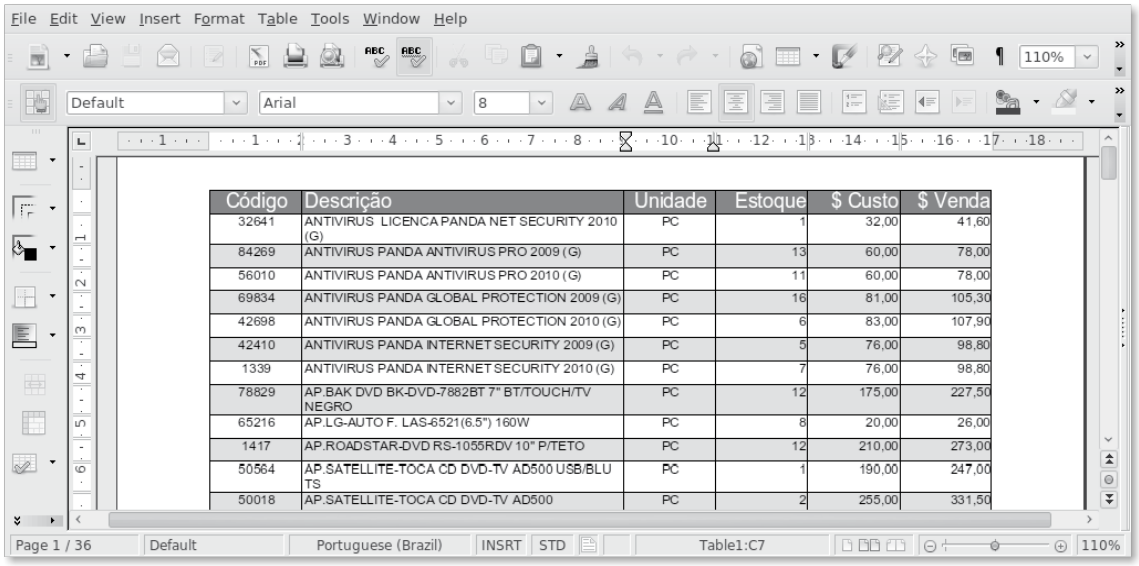
de parágrafo. Após criarmos a linha contendo os títulos das colunas, utilizamos o método `setBackgroundForCellRange()` para colorir o fundo da primeira linha do relatório.

Antes de iniciarmos o `foreach`, percorrendo os dados do relatório, criamos a variável de controle `$colore` (que controlará a cor de fundo da linha dos dados), as variáveis totalizadoras `$total_custo` e `$total_venda`, além da variável `$linha`, que representará a linha atual do relatório, o que é necessário para o método `writeToCell()`.

Dentro do `foreach`, acumulamos as variáveis totalizadoras, formatamos as colunas `preco_custo` e `preco_venda`, conforme o formato monetário, e definimos qual será a cor de fundo utilizada, por meio da variável de controle `$colore`. A linha que terá os dados é adicionada por meio do método `addRow()`. Depois, cada uma das colunas resultantes da base de dados é escrita na tabela por meio do método `writeToCell()`. Posteriormente, o método `setBackgroundForCellRange()` é executado para colorir a linha de dados com a cor definida pela variável `$bgcolor`. Ainda dentro do `foreach`, a variável `$colore` alterna seu valor (`TRUE/FALSE`) e a variável `$linha` tem seu conteúdo incrementado.

Saindo do `foreach`, adicionamos mais uma linha à tabela, desta vez para apresentar o resultado total. As primeiras quatro células da tabela são mescladas por meio do método `mergeCellRange()`. Nesse espaço será exibido o texto "Total". Ao lado da célula contendo o texto "Total", são exibidas as variáveis `$total_custo` e `$total_venda` formatadas e alinhadas à direita. Posteriormente, a linha de total é colorida por meio do método `setBackgroundForCellRange()` e utilizamos o método `setBorderForCellRange()` para apresentar bordas em toda a tabela.

Por fim, o relatório é salvo no arquivo *output.rtf*, dentro da pasta *app.output*, e um *link* é escrito na tela para que o usuário realize o *download* desse arquivo. A figura 4.5 representa o início do relatório gerado.



Código	Descrição	Unidade	Estoque	$ Custo	$ Venda
32641	ANTIVIRUS LICENCA PANDA NET SECURITY 2010 (G)	PC	1	32,00	41,60
84269	ANTIVIRUS PANDA ANTIVIRUS PRO 2009 (G)	PC	13	60,00	78,00
56010	ANTIVIRUS PANDA ANTIVIRUS PRO 2010 (G)	PC	11	60,00	78,00
69834	ANTIVIRUS PANDA GLOBAL PROTECTION 2009 (G)	PC	16	81,00	105,30
42698	ANTIVIRUS PANDA GLOBAL PROTECTION 2010 (G)	PC	6	83,00	107,90
42410	ANTIVIRUS PANDA INTERNET SECURITY 2009 (G)	PC	5	76,00	98,80
1339	ANTIVIRUS PANDA INTERNET SECURITY 2010 (G)	PC	7	76,00	98,80
78829	AP.BAK DVD BK-DVD-7882BT 7" BT/TOUCH/TV NEGRO	PC	12	175,00	227,50
65216	AP.LG-AUTO F. LAS-6521(6.5") 160W	PC	8	20,00	26,00
1417	AP.ROADSTAR-DVD RS-1055RDV 10" P/TETO	PC	12	210,00	273,00
50564	AP.SATELLITE-TOCA CD DVD-TV AD500 USB/BLUTS	PC	1	190,00	247,00
50018	AP.SATELLITE-TOCA CD DVD-TV AD500	PC	2	255,00	331,50

Figura 4.5 – Relatório no formato RTF.

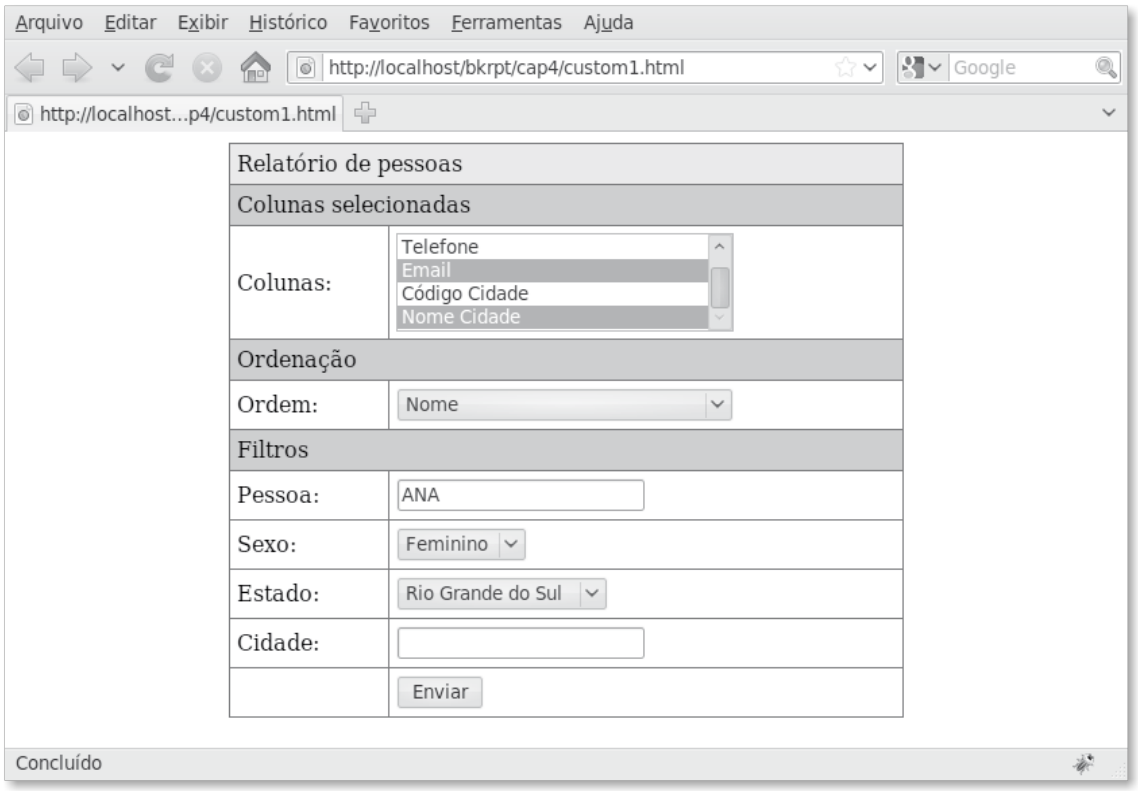


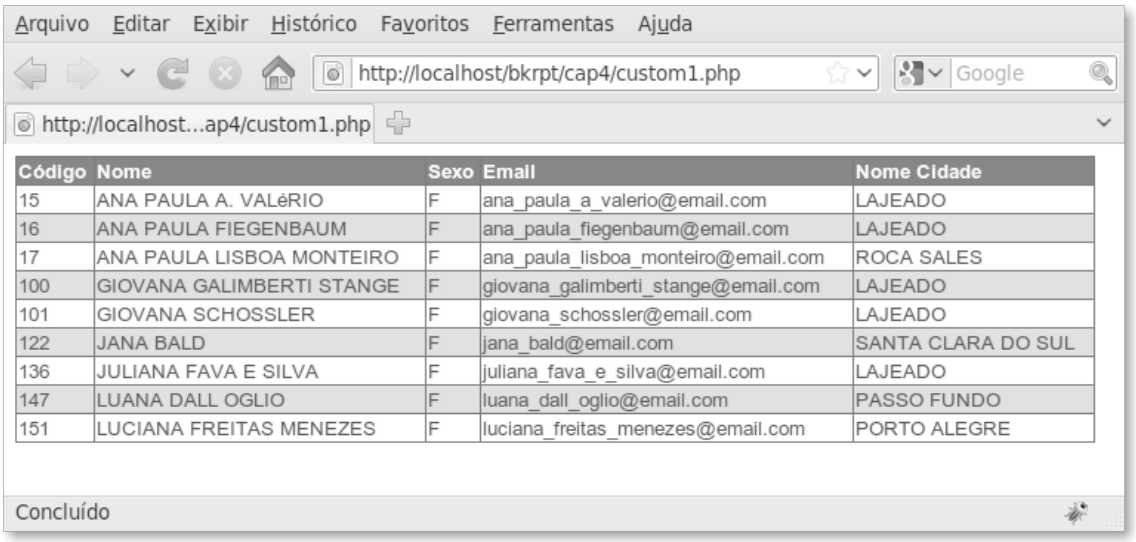
Figura 4.10 – Formulário de personalização do relatório.

custom1.html

```
<html>
    <body>
        <form method="POST" action="custom1.php">
            <table align="center" width="500px" border="1"
                    style="border-collapse:collapse" cellpadding=5>
                <tr bgcolor="#dfdfdf">
                    <td colspan=2>Relatório de pessoas</td>
                </tr>

                <tr bgcolor="#AEB5D8">
                    <td colspan=2>Colunas selecionadas</td>
                </tr>
                <tr>
                    <td>Colunas: </td>
                    <td>
                        <select name="colunas[]" multiple="multiple" size=4 style="width:250px">
                        <option value="pessoa.id">Código</option>
                        <option value="pessoa.nome">Nome</option>
                        <option value="pessoa.sexo">Sexo</option>
                        ...
                        </select>
```

Depois de formar a linha contendo os títulos das colunas, percorremos os dados resultantes por meio de um `foreach` sobre a variável `$result`. Para cada linha resultante, adicionamos uma linha à tabela. Dentro desse laço de iteração, percorremos as colunas selecionadas pelo usuário (`$_REQUEST['colunas']`) para somente adicionar à linha de dados realmente as colunas selecionadas. Neste ponto, definimos o alinhamento da coluna com base na informação armazenada no vetor `$align`. Depois de adicionar todas as linhas resultantes, apresentamos a tabela com o relatório por meio do método `show()` do objeto `$tabela`.

No relatório a seguir, apresentado na figura 4.11, podemos ver o resultado de um relatório de pessoas em que o usuário selecionou visualizar as colunas código, nome, sexo, email e nome da cidade, ordenadas pelo nome da pessoa, e estabeleceu como filtros do relatório somente listar as pessoas cujo nome contém "Ana", são do sexo "Feminino" e residem no estado do "Rio Grande do Sul".



Código	Nome	Sexo	Email	Nome Cidade
15	ANA PAULA A. VALéRIO	F	ana_paula_a_valerio@email.com	LAJEADO
16	ANA PAULA FIEGENBAUM	F	ana_paula_fiegenbaum@email.com	LAJEADO
17	ANA PAULA LISBOA MONTEIRO	F	ana_paula_lisboa_monteiro@email.com	ROCA SALES
100	GIOVANA GALIMBERTI STANGE	F	giovana_galimberti_stange@email.com	LAJEADO
101	GIOVANA SCHOSSLER	F	giovana_schossler@email.com	LAJEADO
122	JANA BALD	F	jana_bald@email.com	SANTA CLARA DO SUL
136	JULIANA FAVA E SILVA	F	juliana_fava_e_silva@email.com	LAJEADO
147	LUANA DALL OGLIO	F	luana_dall_oglio@email.com	PASSO FUNDO
151	LUCIANA FREITAS MENEZES	F	luciana_freitas_menezes@email.com	PORTO ALEGRE

Figura 4.11 – Resultado da execução do relatório.

```
$result = $conn->query($sql);      // executa a instrução SQL

$tabela = new TTable;      // instancia objeto tabela
// define algumas propriedades da tabela
$tabela->width = 730;
$tabela->border = 1;
$tabela->style = "border-collapse:collapse";

// instancia uma linha para o cabeçalho
$cabecalho = $tabela->addRow();

// adiciona células de título das colunas
foreach ($_REQUEST['colunas'] as $coluna)
{
```

CAPÍTULO 5

Relatórios hierárquicos e matriciais

> "Viver é enfrentar um problema atrás do outro. O modo como você o encara é que faz a diferença."
>
> Benjamin Franklin

Neste capítulo, abordaremos a criação de relatórios hierárquicos, que apresentam as informações divididas em quebras e seções, permitindo totalizações em diferentes níveis, e também relatórios matriciais, que apresentam a informação de maneira sintetizada, utilizando o formato de uma matriz como meio de visualização.

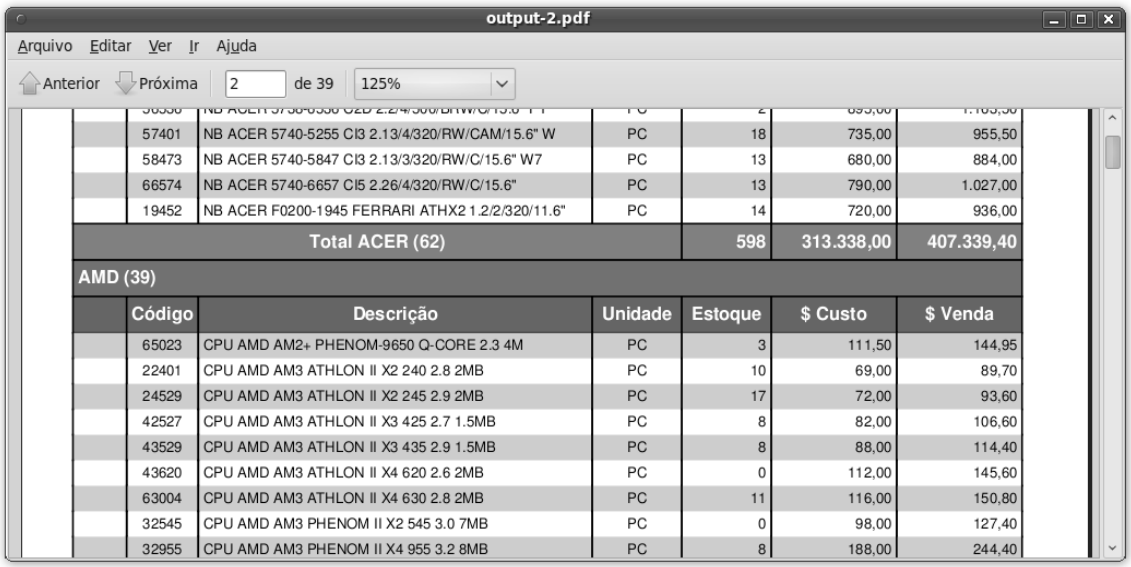
Introdução

No capítulo 4, abordamos a geração de relatórios tabulares. Nesse sentido, foram desenvolvidos relatórios simples, com conteúdo e estrutura predefinidos em nível de programação, relatórios parametrizados, que permitiam ao usuário estabelecer filtros sobre os dados a serem apresentados, relatórios customizados, que também permitiam ao usuário selecionar colunas e ordenações e, por último, relatórios interligados, que permitiam ao usuário realizar saltos entre um relatório e outro.

Neste capítulo, abordaremos a criação e apresentação de relatórios contendo agrupamentos de informações. O objetivo deste capítulo é demonstrar de quais formas podemos realizar agrupamentos e sumarizações sobre a informação presente no banco de dados no momento de ser apresentada ao usuário. Dessa forma, o capítulo será dividido em duas partes: relatórios hierárquicos e relatórios matriciais.

Relatórios hierárquicos, muitas vezes conhecidos também como relatórios com quebras ou agrupamentos, permitem apresentar a informação tabular em um nível detalhado, separada em grupos de dados também conhecidos como quebras, permitindo aplicar fórmulas totalizadoras conforme o nível de agrupamento desejado. A característica principal desse tipo de relatório é a segmentação da informação conforme os grupos e a totalização dos dados após a apresentação deles dentro de cada agrupamento.

Já as variáveis `$total_custo_quebra`, `$total_venda_quebra` e `$total_estoq_quebra` armazenam os totais das colunas de preço de custo, preço de venda e quantidade em estoque dentro da quebra do relatório, ou seja, acumulam os valores dentro de um mesmo fabricante. Cada vez que a coluna do fabricante troca de valor e a quebra é totalizada, essas variáveis são reinicializadas. A variável `$controle_quebra` armazenará o último fabricante (valor anterior da quebra). Essa variável controla a troca de quebra. Dessa maneira, sempre que a comparação dessa variável com o conteúdo da coluna "`fabricante`" do produto atual resultar em uma diferença, teremos uma troca de grupo (quebra). A figura 5.2 apresenta o resultado do relatório.



	57401	NB ACER 5740-5255 CI3 2.13/4/320/RW/CAM/15.6" W	PC	18	735,00	955,50
	58473	NB ACER 5740-5847 CI3 2.13/3/320/RW/C/15.6" W7	PC	13	680,00	884,00
	66574	NB ACER 5740-6657 CI5 2.26/4/320/RW/C/15.6"	PC	13	790,00	1.027,00
	19452	NB ACER F0200-1945 FERRARI ATHX2 1.2/2/320/11.6"	PC	14	720,00	936,00
	Total ACER (62)			598	313.338,00	407.339,40
AMD (39)						
	Código	Descrição	Unidade	Estoque	$ Custo	$ Venda
	65023	CPU AMD AM2+ PHENOM-9650 Q-CORE 2.3 4M	PC	3	111,50	144,95
	22401	CPU AMD AM3 ATHLON II X2 240 2.8 2MB	PC	10	69,00	89,70
	24529	CPU AMD AM3 ATHLON II X2 245 2.9 2MB	PC	17	72,00	93,60
	42527	CPU AMD AM3 ATHLON II X3 425 2.7 1.5MB	PC	8	82,00	106,60
	43529	CPU AMD AM3 ATHLON II X3 435 2.9 1.5MB	PC	8	88,00	114,40
	43620	CPU AMD AM3 ATHLON II X4 620 2.6 2MB	PC	0	112,00	145,60
	63004	CPU AMD AM3 ATHLON II X4 630 2.8 2MB	PC	11	116,00	150,80
	32545	CPU AMD AM3 PHENOM II X2 545 3.0 7MB	PC	0	98,00	127,40
	32955	CPU AMD AM3 PHENOM II X4 955 3.2 8MB	PC	8	188,00	244,40

Figura 5.2 – Relatório com uma quebra em PDF.

quebra_uma_pdf.php

```
<?php
function __autoload($classe)
{
    if (file_exists("app.ado/{$classe}.class.php"))
    {
        include_once "app.ado/{$classe}.class.php";
    }
}

try
{
    $conn = TConnection::open('exemplos');    // abre uma conexão

    // define a consulta
    $sql = "SELECT fabricante.nome || ' (' || fabricante.id || ')' as fabricante, ".
           "       produto.id, produto.descricao, produto.unidade, ".
```

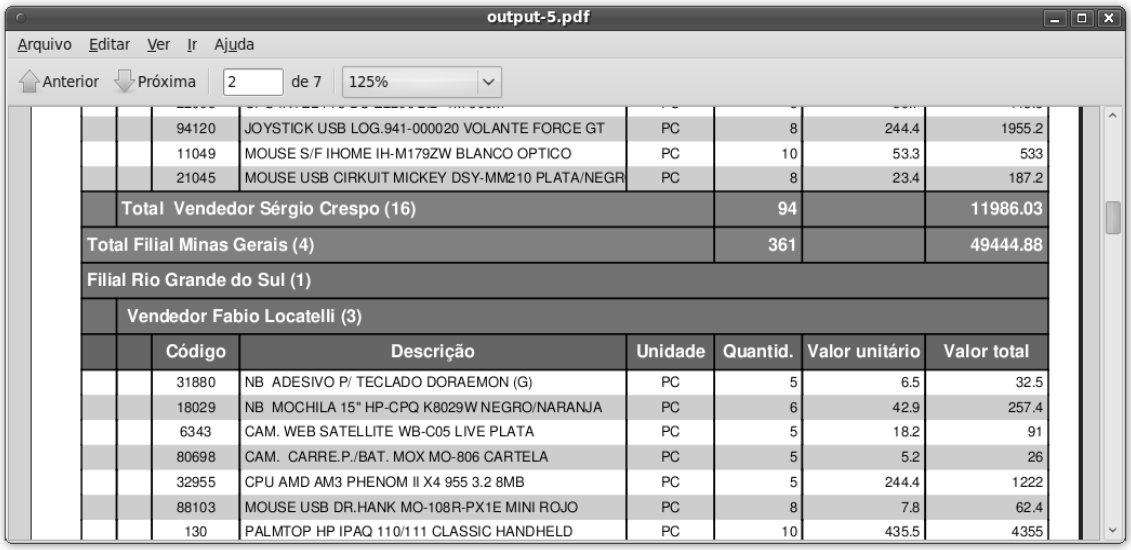


Figura 5.4 – Relatório com duas quebras em PDF.

quebras_duas_pdf.php

```
<?php
function __autoload($classe)
{
    ...
}

try
{
    $conn = TConnection::open('exemplos');    // abre uma conexão

    // define a consulta
    $sql = "SELECT filial.nome || ' (' || filial.id || ')' as filial," .
           "       ' Vendedor ' || vendedor.nome || ' (' || vendedor.id || ')' as vendedor," .
           "       produto.id, produto.descricao, produto.unidade, ".
           "       venda_itens.quantidade, venda_itens.valor" .
           " FROM  venda, venda_itens, pessoa, vendedor, filial, produto" .
           " WHERE venda.id = venda_itens.id_venda AND" .
           "       venda.id_cliente = pessoa.id AND" .
           "       venda.id_vendedor = vendedor.id AND" .
           "       venda.id_filial = filial.id AND" .
           "       venda_itens.id_produto = produto.id".
           " ORDER BY 1, 2";

    $result = $conn->query($sql);     // executa a instrução SQL

    require('app.util/pdf/fpdf.php');     // inclui a classe FPDF
    $pdf = new FPDF('P', 'pt', 'A4');     // instancia a classe FPDF
```

Duas dimensões no formato HTML

Nesse primeiro relatório matricial, será apresentado um resumo de vendas conforme dois agrupamentos (dimensões) de dados: vendedor e mês. Assim, a primeira dimensão será exibida nas linhas (vendedor) e a outra, nas colunas (mês), enquanto no interior do relatório será apresentada a quantidade vendida. É importante ressaltar que cada célula do interior do relatório deve ser lida como uma interseção das duas dimensões, que são vendedor (linhas) e mês (colunas). O relatório apresenta, ainda, duas totalizações, sendo uma na vertical, apresentando o total de vendas de cada mês, e outra na horizontal, apresentando o total de vendas de cada vendedor.

A ideia geral desse relatório é transmitida pela figura 5.5, que não contém exatamente os mesmos dados apresentados no relatório que desenvolveremos, mas apenas uma ideia abstrata. No lado esquerdo da figura, constam os dados de origem que serão utilizados como base para confecção do relatório matricial. Esses dados são extraídos da base de dados por meio de um simples comando SELECT que lista o nome do vendedor, o mês em que ocorreu a venda e o valor vendido. A partir desses dados, chegamos à parte direita da figura, que exibe um relatório matricial de duas dimensões já pronto.

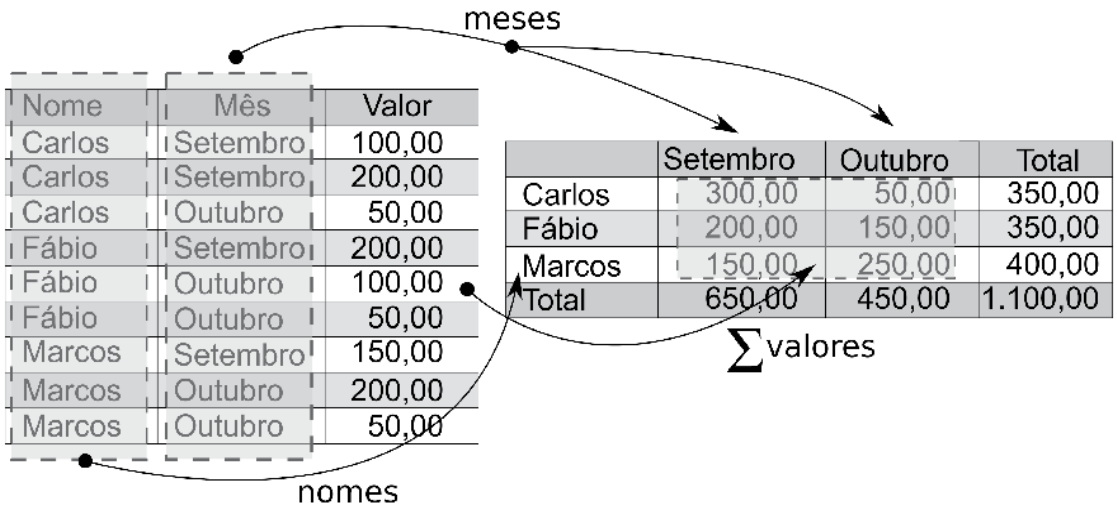
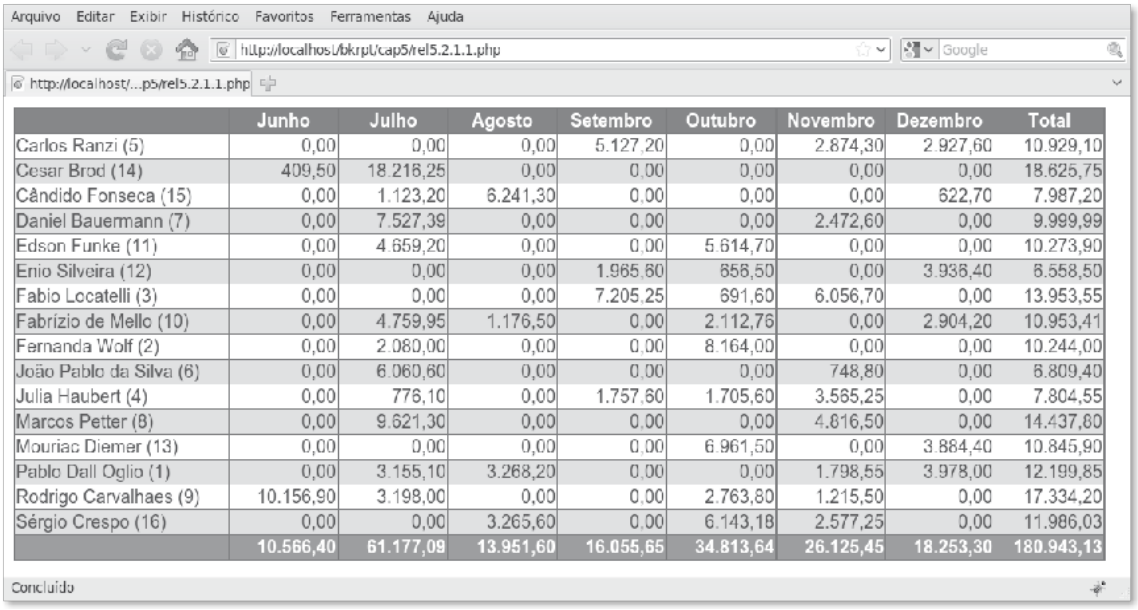


Figura 5.5 – Formação de uma matriz com duas dimensões.

Neste relatório, podemos observar que duas colunas do relatório foram selecionadas como agrupamentos ou dimensões do relatório matricial: nome do vendedor e mês da venda. Cada informação de agrupamento foi disposta em um eixo diferente do relatório resultante, sendo que o nome do vendedor foi exibido ao longo das linhas, enquanto o mês de realização das vendas, ao longo das colunas.

É importante ressaltar que em um relatório matricial a informação de agrupamento é apresentada de maneira distinta, ou seja, tanto os nomes dos vendedores quanto os nomes dos meses são apresentados uma única vez em cada célula. No interior do



Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

http://localhost/bkrpt/cap5/rel5.2.1.1.php

http://localhost/...p5/rel5.2.1.1.php

	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro	Total
Carlos Ranzi (5)	0,00	0,00	0,00	5.127,20	0,00	2.874,30	2.927,60	10.929,10
Cesar Brod (14)	409,50	18.216,25	0,00	0,00	0,00	0,00	0,00	18.625,75
Cândido Fonseca (15)	0,00	1.123,20	6.241,30	0,00	0,00	0,00	622,70	7.987,20
Daniel Bauermann (7)	0,00	7.527,39	0,00	0,00	0,00	2.472,60	0,00	9.999,99
Edson Funke (11)	0,00	4.659,20	0,00	0,00	5.614,70	0,00	0,00	10.273,90
Enio Silveira (12)	0,00	0,00	0,00	1.965,60	656,50	0,00	3.936,40	6.558,50
Fabio Locatelli (3)	0,00	0,00	0,00	7.205,25	691,60	6.056,70	0,00	13.953,55
Fabrízio de Mello (10)	0,00	4.759,95	1.176,50	0,00	2.112,76	0,00	2.904,20	10.953,41
Fernanda Wolf (2)	0,00	2.080,00	0,00	0,00	8.164,00	0,00	0,00	10.244,00
João Pablo da Silva (6)	0,00	6.060,60	0,00	0,00	0,00	748,80	0,00	6.809,40
Julia Haubert (4)	0,00	776,10	0,00	1.757,60	1.705,60	3.565,25	0,00	7.804,55
Marcos Petter (8)	0,00	9.621,30	0,00	0,00	0,00	4.816,50	0,00	14.437,80
Mouriac Diemer (13)	0,00	0,00	0,00	0,00	6.961,50	0,00	3.884,40	10.845,90
Pablo Dall Oglio (1)	0,00	3.155,10	3.268,20	0,00	0,00	1.798,55	3.978,00	12.199,85
Rodrigo Carvalhaes (9)	10.156,90	3.198,00	0,00	0,00	2.763,80	1.215,50	0,00	17.334,20
Sérgio Crespo (16)	0,00	0,00	3.265,60	0,00	6.143,18	2.577,25	0,00	11.986,03
	10.566,40	61.177,09	13.951,60	16.055,65	34.813,64	26.125,45	18.253,30	180.943,13

Concluído

Figura 5.6 – Relatório com duas dimensões em HTML.

matriz_duas_html.php

```
<?php
function __autoload($classe)
{
    ...
}

try
{
    // cria um estilo para o cabeçalho
    $estilo_cabecalho = new TStyle('cabecalho');
    $estilo_cabecalho->font_family = 'arial,verdana,sans-serif';
    $estilo_cabecalho->color = '#ffffff';
    ...
    $estilo_cabecalho->show();

    // cria um estilo para o total
    $estilo_total = new TStyle('total');
    $estilo_total->font_family = 'arial,verdana,sans-serif';
    $estilo_total->color = '#ffffff';
    ...
    $estilo_total->show();

    // cria um estilo para os dados
    $estilo_dados = new TStyle('dados');
    $estilo_dados->color = '#2D2D2D';
    ...
```

```
    echo '</center>';
}
catch (Exception $e)
{
    echo $e->getMessage();     // exibe a mensagem de erro
}
?>
```

Obs.: A versão sem cortes desse código encontra-se para download no site da editora Novatec.

Quatro dimensões no formato HTML

Nos dois exemplos anteriores, foram apresentados relatórios matriciais que exibiam os dados conforme duas dimensões: vendedor e mês. Os dois próximos exemplos de relatórios matriciais mostrarão as informações de vendas conforme quatro dimensões: filial, nome do vendedor, mês de venda e sexo do cliente. Duas dessas dimensões (filial e vendedor) serão dispostas nas linhas e as outras duas (mês da venda e sexo do cliente), nas colunas.

A figura 5.8 procura demonstrar, de maneira abstrata, a ideia geral dos próximos dois exemplos. É importante ressaltar que os dados contidos nessa figura são fictícios e não correspondem aos dados presentes no banco de dados que serão apresentados no próximo exemplo. Em seu lado esquerdo, a figura tem uma tabela que representa os dados de origem trazidos pela consulta SQL realizada no banco de dados. Depois de serem processados, esses dados resultam na matriz da direita, que representa o relatório matricial resultante.

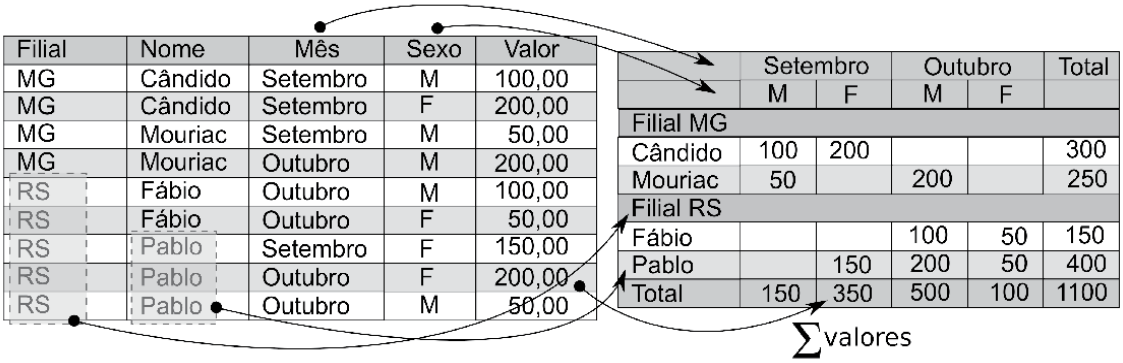
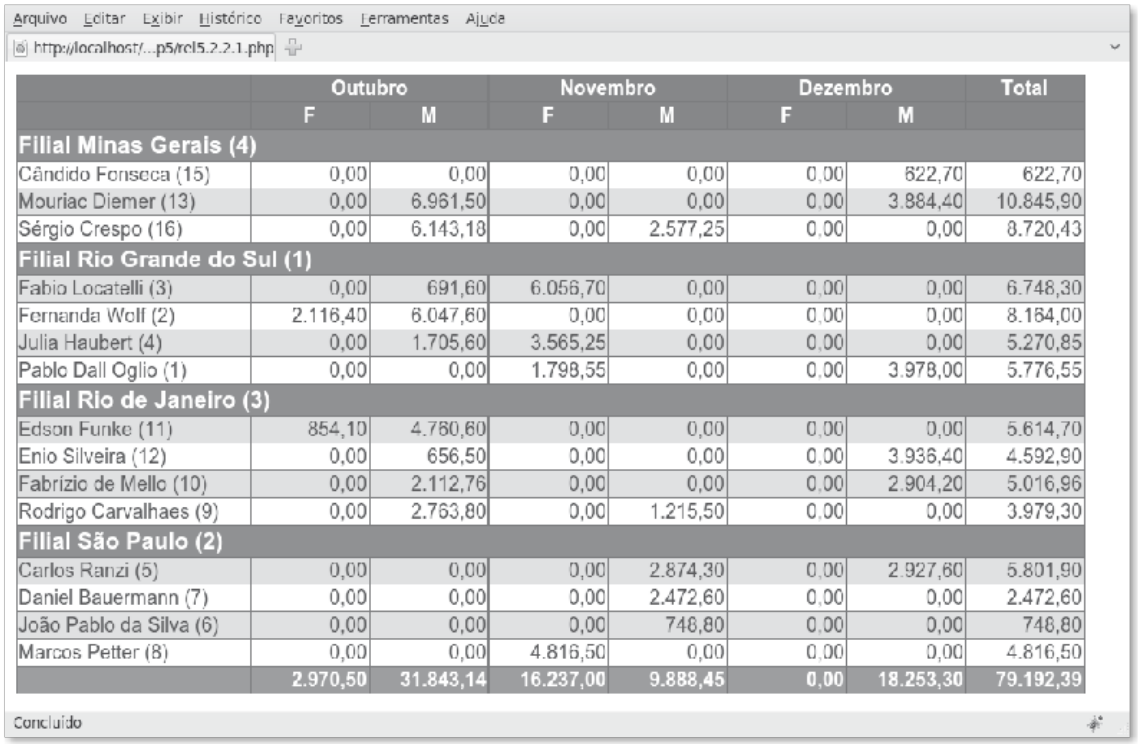


Figura 5.8 – Formação de uma matriz com quatro dimensões.

A tabela da esquerda, que contém os dados de origem, tem cinco colunas, e quatro delas se transformarão em dimensões do relatório matricial, e a quinta delas, que representa o valor vendido, será a coluna a ser totalizada no interior do relatório resultante. As duas primeiras colunas da tabela da esquerda (filial e nome do vendedor) serão as

A variável `$matriz` contém uma estrutura de dados que armazena os totais de vendas de maneira similar como estes serão exibidos. Trata-se de uma matriz com quatro índices (filial, vendedor, mês da venda e sexo do cliente) que será utilizada para a exibição do relatório. Já a matriz `$totais_vendedor` armazenará os totais de cada vendedor e será utilizada para apresentar a última coluna do relatório. Dessa forma, essa matriz armazena a totalização de vendas conforme a filial e o vendedor. Já a matriz `$totais_mes` armazenará a totalização de vendas conforme o mês e o sexo e será utilizada para apresentar a última linha do relatório.

Os vetores `$sexos` e `$meses` armazenarão os diferentes valores das colunas sexo e mês, resultantes da consulta ao banco de dados, para montagem das colunas do relatório. Esses vetores são unificados (`array_unique()`) para garantir que não contenham valores repetidos e são ordenados (`sort()`). A figura 5.9 apresenta o resultado do relatório.


Arquivo Editar Exibir Histórico Favoritos Ferramentas Ajuda

http://localhost/...p5/rel5.2.2.1.php

	Outubro		Novembro		Dezembro		Total
	F	M	F	M	F	M	
Filial Minas Gerais (4)							
Cândido Fonseca (15)	0,00	0,00	0,00	0,00	0,00	622,70	622,70
Mouriac Diemer (13)	0,00	6.961,50	0,00	0,00	0,00	3.884,40	10.845,90
Sérgio Crespo (16)	0,00	6.143,18	0,00	2.577,25	0,00	0,00	8.720,43
Filial Rio Grande do Sul (1)							
Fabio Locatelli (3)	0,00	691,60	6.056,70	0,00	0,00	0,00	6.748,30
Fernanda Wolf (2)	2.116,40	6.047,60	0,00	0,00	0,00	0,00	8.164,00
Julia Haubert (4)	0,00	1.705,60	3.565,25	0,00	0,00	0,00	5.270,85
Pablo Dall Oglio (1)	0,00	0,00	1.798,55	0,00	0,00	3.978,00	5.776,55
Filial Rio de Janeiro (3)							
Edson Funke (11)	854,10	4.760,60	0,00	0,00	0,00	0,00	5.614,70
Enio Silveira (12)	0,00	656,50	0,00	0,00	0,00	3.936,40	4.592,90
Fabrízio de Mello (10)	0,00	2.112,76	0,00	0,00	0,00	2.904,20	5.016,96
Rodrigo Carvalhaes (9)	0,00	2.763,80	0,00	1.215,50	0,00	0,00	3.979,30
Filial São Paulo (2)							
Carlos Ranzi (5)	0,00	0,00	0,00	2.874,30	0,00	2.927,60	5.801,90
Daniel Bauermann (7)	0,00	0,00	0,00	2.472,60	0,00	0,00	2.472,60
João Pablo da Silva (6)	0,00	0,00	0,00	748,80	0,00	0,00	748,80
Marcos Petter (8)	0,00	0,00	4.816,50	0,00	0,00	0,00	4.816,50
	2.970,50	**31.843,14**	**16.237,00**	**9.888,45**	**0,00**	**18.253,30**	**79.192,39**

Concluído

Figura 5.9 – Relatório com quatro dimensões em HTML.

matriz_quatro_html.php

```
<?php
function __autoload($classe)
{
    ...
}
```

CAPÍTULO 6

Gráficos e documentos

> "Eu prefiro ser essa metamorfose ambulante do que ter aquela velha opinião formada sobre tudo."
>
> Raul Seixas

Neste capítulo será abordada a geração de gráficos, bem como de documentos. O capítulo aborda quatro tópicos principais: geração de gráficos de linhas e colunas segundo os dados existentes na base de dados, de gráficos com interação com o usuário, de documentos no formato PDF e de documentos no formato RTF.

Gráficos

No capítulo 3 foram apresentadas as bibliotecas utilizadas ao longo do livro, entre elas a JPGraph. Os exemplos desse capítulo utilizavam apenas dados estáticos, definidos na forma de vetores, para gerar gráficos. Os próximos exemplos tratam da geração desses mesmos gráficos, mas conforme os dados provindos da base de dados utilizada até o momento.

Gráfico de linhas

O próximo exemplo trata da geração de um gráfico de linhas a partir dos dados existentes na base de dados utilizada até o momento. O objetivo do gráfico será projetar as vendas (R$) realizadas em cada uma das filiais ao longo dos meses. Nesse gráfico, cada linha representará uma filial, conforme podemos conferir na legenda. O gráfico apresentará, no eixo X, os meses em que as vendas ocorreram e, no eixo Y, o montante de vendas.

O programa inicia com a apresentação do método `__autoload()`, que definirá a partir de quais locais as classes de conexão com a base de dados (*app.ado*) serão carregadas. A partir de então, define o ano (variável `$ano`) que será utilizado na consulta SQL para montar o gráfico. Após a abertura da conexão com a base de dados, a consulta SQL é definida, tendo como objetivo listar o montante realizado em vendas agrupado por

`$plotdata`, que contém sempre seis valores (desde julho até dezembro) para serem projetados. Só depois, o objeto `LinePlot` é instanciado, recebendo esse vetor como parâmetro. Em seguida, são definidas algumas características de cada linha como cor, espessura, exibição de marcas e valores e, ao final, o objeto `LinePlot` é adicionado ao gráfico por meio do método `Add()` da classe `Graph` e o gráfico é exibido em tela por meio do método `Stroke()`. A figura 6.1 apresenta o resultado do gráfico.

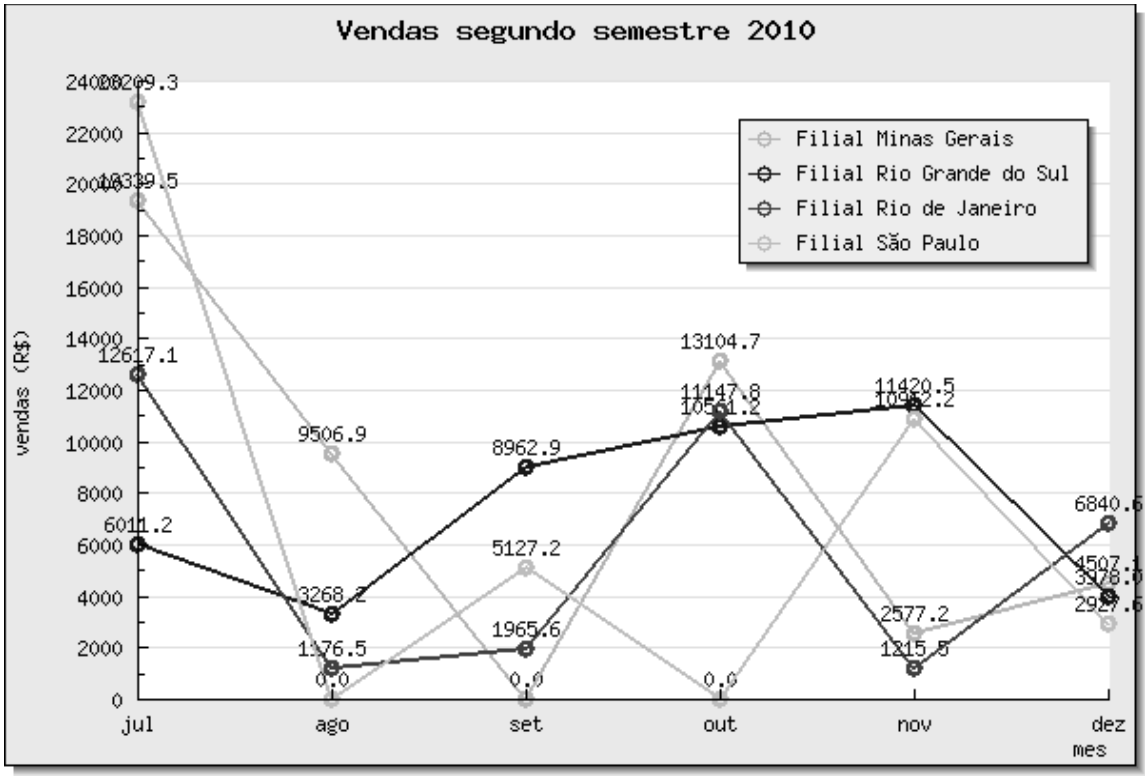


Figura 6.1 – Gráfico de linhas representando as vendas de cada filial por mês.

grafico_linhas.php

```
<?php
function __autoload($classe)
{
    ...
}

try
{
    $ano = 2010;        // define o ano-base da consulta
    $conn = TConnection::open('exemplos');    // abre uma conexão
```

seis posições, e o volume de vendas de cada mês, quando essa informação existir, ou zero, quando não existir.

Depois de instanciar um objeto `BarPlot`, algumas características são definidas, como a cor de cada barra (`SetFillColor()`), se os valores deverão ser exibidos ou não, o nome que representará a barra na legenda (`SetLegend()`), se as barras terão sombreamento (`SetShadow()`), entre outras. Por fim, a barra é adicionada ao vetor `$barplots`, que terá todas as barras (uma para cada filial). Depois de percorrer a matriz `$data`, é criado um objeto da classe `GroupBarPlot`, que representa um agrupamento de barras. Esse objeto recebe como parâmetro o vetor `$barplots`, sendo responsável por sua exibição. Esse objeto `GroupBarPlot` é, então, adicionado ao gráfico por meio do método `Add()` da classe `Graph`, que, então, é exibido em tela pelo método `Stroke()`. A figura 6.2 apresenta o resultado do gráfico.

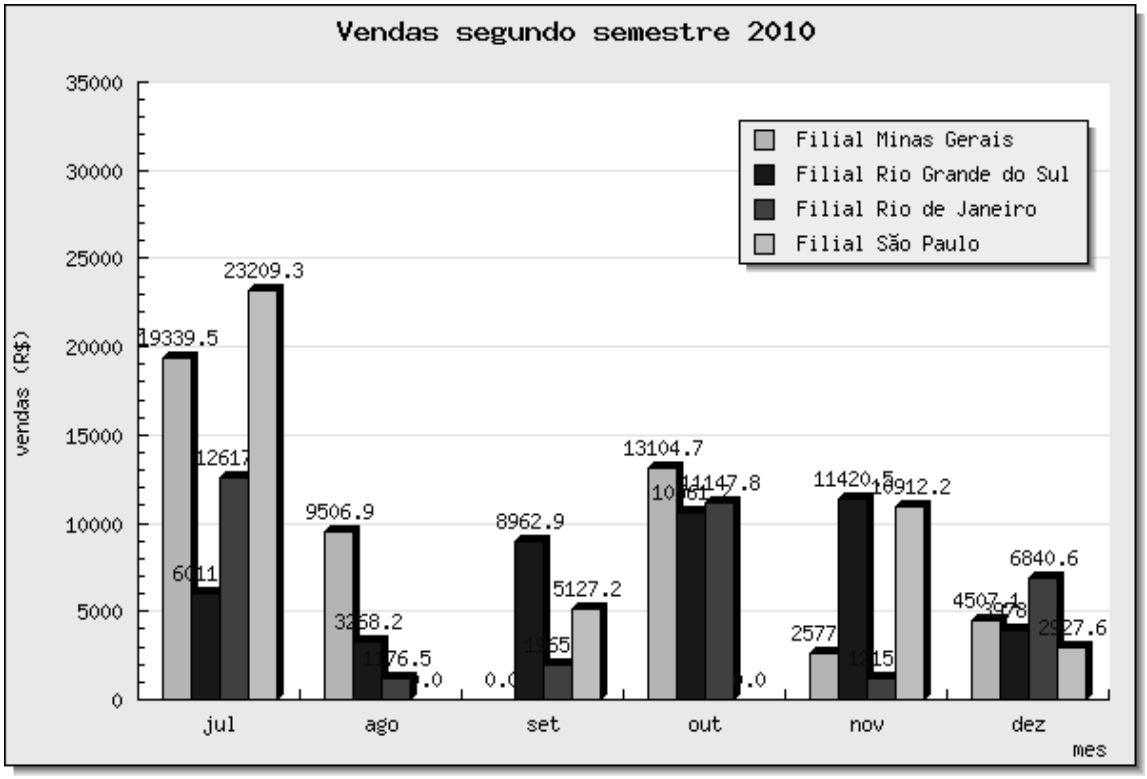


Figura 6.2 – Gráfico de colunas representando as vendas de cada filial por mês.

grafico_colunas.php

```
<?php
function __autoload($classe)
{
    ...
}

try
```

Depois de percorrer os dados retornados da base de dados, o gráfico é instanciado por meio da criação do objeto da classe `PieGraph`. Depois, algumas características do gráfico como sombreamento (`SetShadow()`) e título são definidas. Depois, um objeto do tipo `PiePlot3D` é criado, recebendo como parâmetro o vetor que contém os dados a serem projetados (`$dados`).

O método `ExplodeSlice()` define uma fatia a ser distanciada do gráfico, indicando um destaque maior. O método `SetCenter()` define de maneira percentual (entre zero e um) a posição do centro do gráfico em relação aos eixos horizontal e vertical. O método `SetCSIMTargets()` define os vetores contendo os *links* e rótulos de *link*. O método `SetLegends()` atribui as legendas (nomes de filiais) ao gráfico. Ao final, o objeto `PiePlot3D` é adicionado ao objeto `PieGraph`, por meio do método `Add()`, e o gráfico é exibido em tela, por meio do método `StrokeCSIM()` da classe `PieGraph`, que, em vez de exibir apenas uma imagem em tela, é responsável também pela geração do código HTML que faz o mapeamento dos *links*, permitindo que o usuário clique sobre as fatias da pizza. A figura 6.3 apresenta o resultado do gráfico.

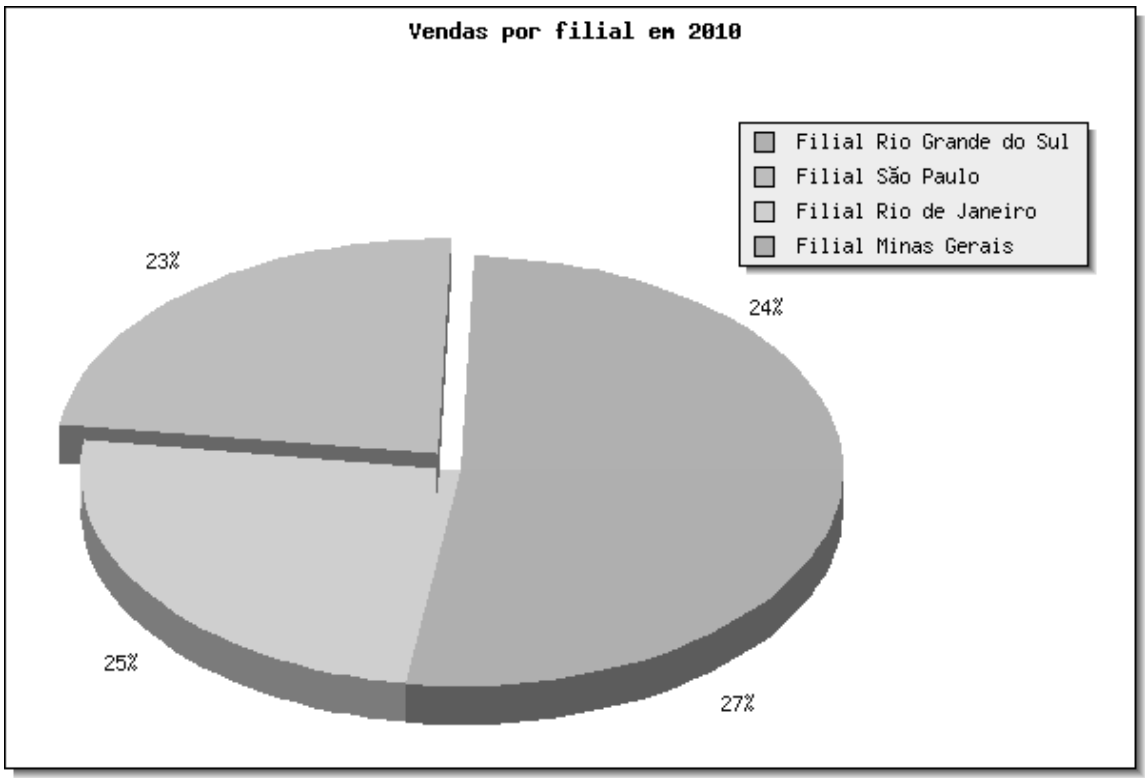


Figura 6.3 – Gráfico de pizza representando o percentual de vendas para cada filial.

grafico_pizza.php

```
<?php
function __autoload($classe)
{
```

O programa a seguir procura demonstrar como se dá a utilização da classe recém-criada `NotaFiscalResumida`. O resultado da execução desse programa pode ser visualizado na figura 6.5.

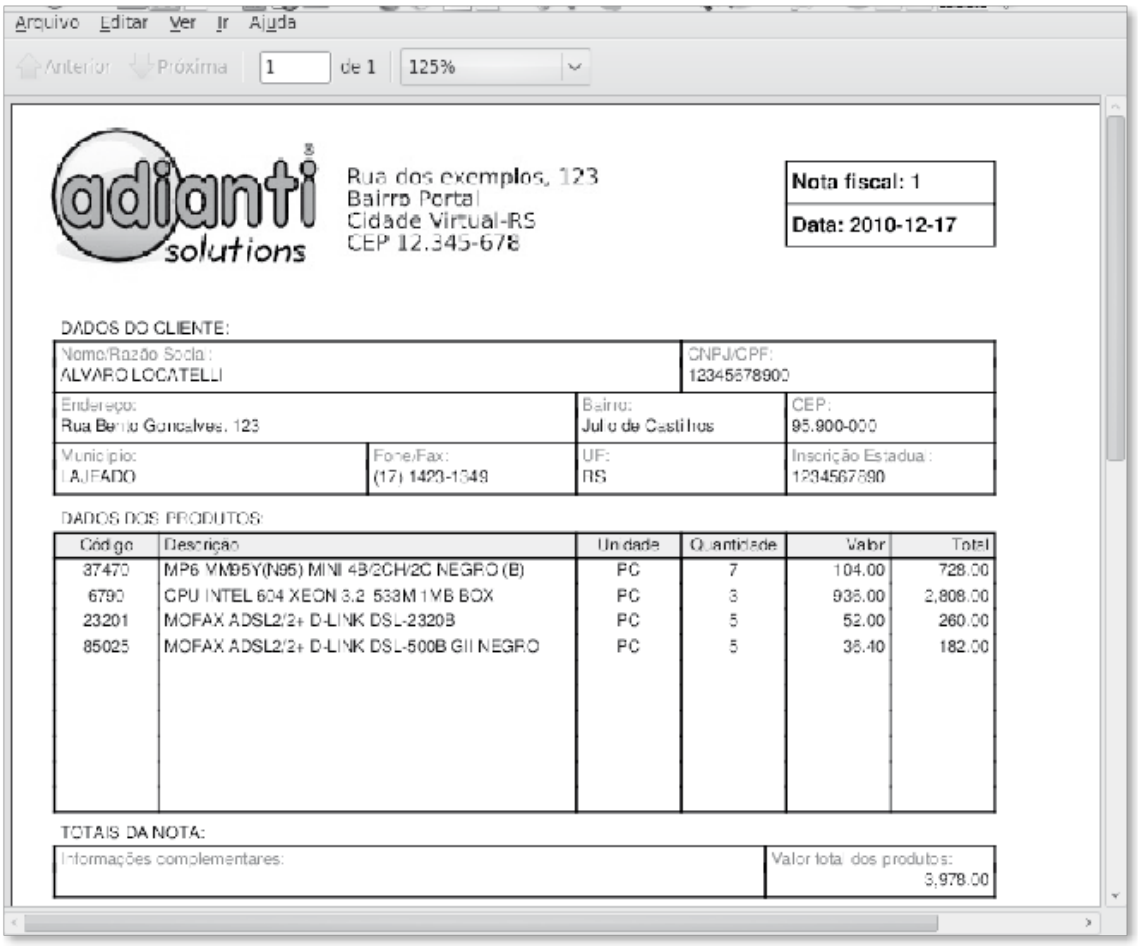


Figura 6.5 – Nota fiscal gerada no formato PDF.

Este exemplo inicia com a definição do método `__autoload()`, que define como se dará a carga das classes de conexão com a base de dados a partir da classe *app.ado*. Em seguida, inicia um bloco `try/catch` com a utilização da classe.

Para emitir as notas fiscais, serão utilizados os dados das tabelas: `pessoa`, `cidade`, `venda`, `venda_itens` e `produto`. Dessa forma, será emitida uma nota fiscal para cada venda realizada. Os produtos integrantes de uma venda farão parte do corpo da nota fiscal.

O código inicia com a declaração da variável `$id_venda`, que define qual o código da venda que será utilizado para emissão da nota fiscal. Esse código está definido de maneira fixa, mas, em uma aplicação real, seria passado via formulário de parâmetros. Logo depois de definir o código da venda, abrimos uma conexão com a base de dados e definimos uma consulta SQL para obter, a partir do código da venda, a data em que esta foi realizada e o código do cliente.

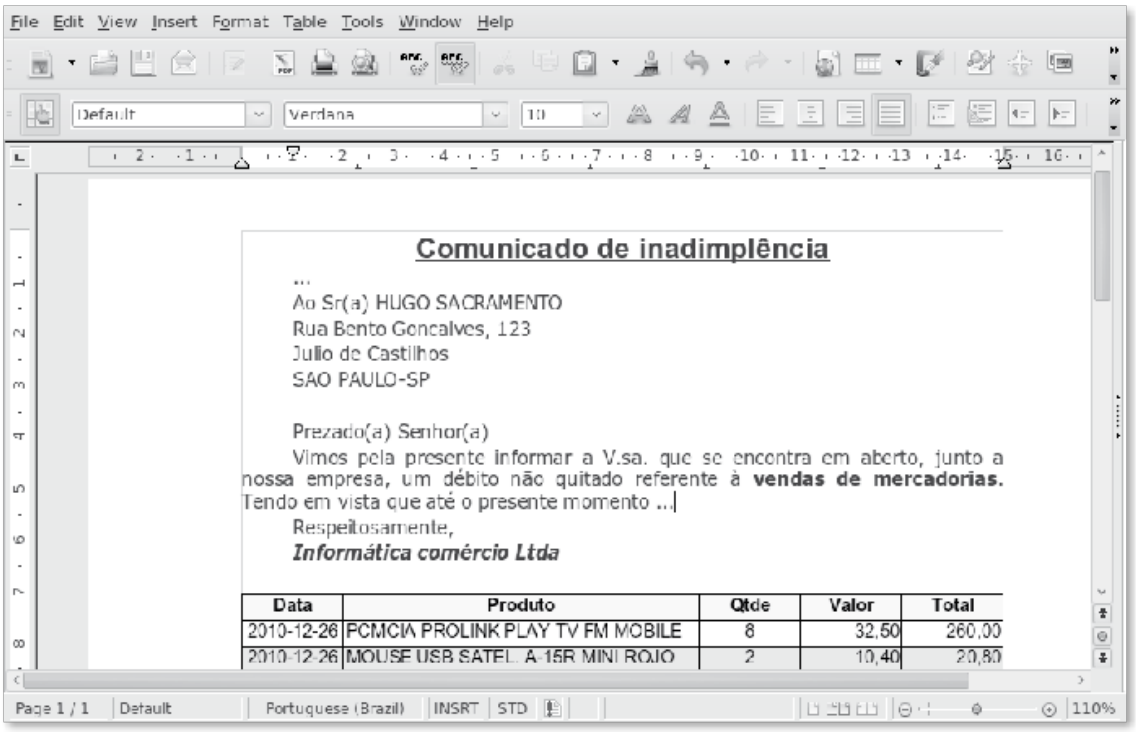


Figura 6.6 – Carta de inadimplência gerada no formato RTF.

Em documentos como cartas de inadimplência, contratos de prestação de serviços, entre outros, é bastante comum o texto do documento sofrer alterações ao longo do tempo, seja por mudanças nas regras de negócio, mudanças na legislação, entre outros fatores. Dessa forma, não é uma boa prática manter esse conteúdo como parte integrante do código-fonte da aplicação, uma vez que qualquer alteração subsequente dependerá da ação do desenvolvedor.

Para evitar esse tipo de manutenção, é recomendável manter o conteúdo do documento em um local que possa ser editado pelo usuário, de preferência pela própria interface da aplicação. Para implementar essa separação em nosso exemplo, o texto da carta ficará armazenado em um arquivo externo à aplicação, chamado de *carta.txt*. Esse documento poderá ser editado pelo usuário e conter inclusive *tags* html como `b` (*bold*), `i` (*italic*) e `u` (*underline*), uma vez que essas *tags* são suportadas pela PHPRtfLite. Como é impossível escrever um único texto de carta que sirva para todos os clientes, esse texto deverá conter variáveis como `$nome`, `$endereco`, `$bairro` e `$municipio`, que serão posteriormente substituídas pelo programa. A utilização de variáveis permite que o texto da carta contenha os dados retornados posteriormente pelo banco de dados, tornando a carta personalizada.

O programa inicia com a declaração do método `__autoload()`, que define como se dá a carga das classes de conexão ao banco de dados. A seguir, já dentro do bloco `try/catch`, definimos o código do cliente para o qual emitiremos a carta de inadimplência. É importante tornar claro que, em uma aplicação real, nunca estaremos definindo

CAPÍTULO 7

Uma biblioteca para geração de relatórios

> "A perfeição não é alcançada quando já não há mais nada para adicionar, mas quando já não há mais nada que se possa retirar."
>
> Antoine de Saint-Exupéry

O objetivo principal deste capítulo é construir um conjunto de classes que permita ao programador desenvolver relatórios tabulares e hierárquicos em formatos diversos, como HTML, PDF e RTF, sem necessariamente conhecer detalhes do funcionamento dessas bibliotecas, como é o caso da FPDF para arquivos PDF ou da PHPRtfLite para arquivos RTF. Para isso, serão criadas classes que automatizam a elaboração desses tipos de relatório, aumentando a produtividade no desenvolvimento de aplicações corporativas no que diz respeito à criação de relatórios operacionais.

Introdução

Ao longo do livro, aprendemos a utilizar algumas das mais importantes bibliotecas para gerar relatórios em PHP e também diversas estratégias para gerar relatórios dos mais diversos tipos, desde os mais simples até os mais complexos. Até este ponto, construímos variados tipos de relatórios, realizamos consultas aos bancos de dados, formatamos os dados para apresentação em tabelas, gráficos, entre outros itens.

No dia a dia do desenvolvimento de aplicações, sobretudo quando falamos de aplicações de negócio, é necessário que tenhamos produtividade e esta possa ser alcançada de diversas maneiras. Quando desenvolvemos uma aplicação orientada a objetos, uma das maneiras mais eficientes de alcançarmos a produtividade é por meio da criação de classes que automatizam tarefas para o programador, de maneira que ele possa fazer cada vez mais com menos linhas de código. É por esse motivo que neste capítulo projetaremos algumas classes que simplificam a tarefa de gerar relatórios, tornando melhor a vida do programador.

Dentre todos os tipos de relatórios que vimos, os relatórios tabulares e os hierárquicos estão entre os mais solicitados pelos usuários de aplicações de negócio em termos

estruturação dos dados antes de sua apresentação. Essa classe permitirá ao desenvolvedor especificar o banco de dados, a consulta, adicionar colunas ao relatório, adicionar grupos e totalizações e, ao final, gerar o relatório em um dos três formatos disponíveis: HTML, PDF ou RTF.

A seleção do formato que será utilizado pela classe `TSimpleReport` para gerar o relatório será realizado pelo programador ao acionar o método `setReportWritter()`. Esse método receberá como parâmetro um objeto `TTableWriterHTML`, `TTableWriterPDF` ou `TTableWriterRTF`, o qual será armazenado no atributo `$writter`. Esse objeto será utilizado ao longo da classe `TSimpleReport` para formatar o relatório. Essa técnica nos permite selecionar um algoritmo, neste caso representado por uma das três classes durante a execução da aplicação. Essa técnica constitui uma variação do *design pattern* `Strategy`. A figura 7.1 procura representar a relação entre as classes nesse padrão.

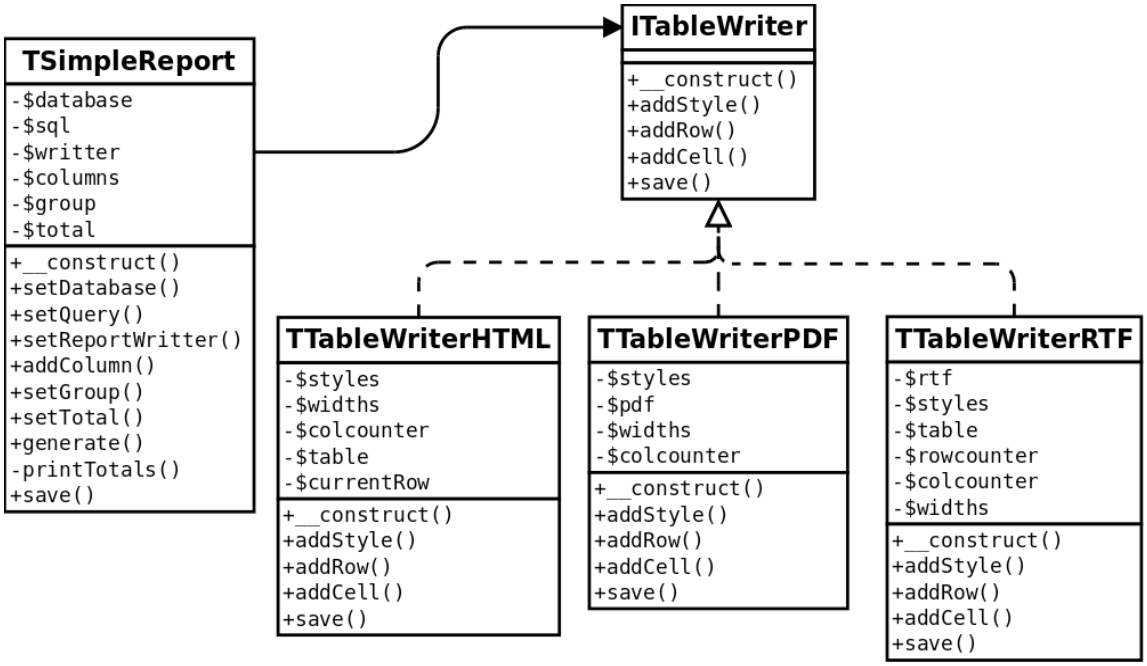


Figura 7.1 – Padrão de projeto Strategy.

Um *design pattern* `Strategy` é representado por uma família de algoritmos encapsulados em uma hierarquia de objetos, de forma que possamos facilmente trocar de algoritmo. Para implementar o *design pattern* `Strategy`, normalmente declaramos uma classe abstrata contendo os métodos que esse algoritmo deve prover e um conjunto de classes-filha concretas desta, que implementam cada versão diferente desse algoritmo. A decisão de qual algoritmo tomar fica por conta do programador que só terá de escolher uma determinada classe a instanciar. Independentemente da classe escolhida, todas as classes-filha implementarão a mesma interface.

```
            $content = ob_get_clean();

            file_put_contents($filename, $content);
            return TRUE;
        }
    }
    ?>
```

Obs.: A versão sem cortes desse código encontra-se para download no site da editora Novatec.

Escritor de PDF

A próxima classe "escritora" que construiremos é a `TTableWriterPDF`, que será responsável pela escrita de tabelas em arquivos PDF. Esta classe fará uso da classe FPDF para construir o arquivo final em PDF. Em seu método construtor, a classe `TTableWriterPDF` receberá como parâmetro um vetor contendo as larguras das colunas da tabela e armazenará esse vetor no atributo `$this->widths`. Além disso, o método construtor inicializa o vetor `$this->styles` e cria o objeto `$this->pdf`, instância da classe FPDF, que será responsável por escrever a tabela no formato PDF. Em seguida são executados os métodos `Open()` e `AddPage()` para abertura do documento PDF.

TTableWriterPDF.class.php

```
<?php
/**
 * Escreve tabelas no formato PDF
 */
class TTableWriterPDF implements ITableWriter
{
    private $styles;
    private $pdf;
    private $widths;
    private $colcounter;

    /**
     * Método construtor
     * @param $widths vetor contendo as larguras das colunas
     */
    public function __construct($widths)
    {
        // armazena as larguras
        $this->widths = $widths;
        // inicializa atributos
        $this->styles = array();
```

```
    // exibe um atalho para o usuário realizar download do arquivo
    echo "Relatório no formato {$formato} gerado com sucesso<br>";
    echo "<a href='saida1.{$formato}'>Clique aqui para efetuar o download</a><br><br>";
}
?>
```

Obs.: A versão sem cortes desse código encontra-se para download no site da editora Novatec.

Utilizando recursos visuais

No último exemplo, vimos como utilizar as classes "escritoras" de tabelas por meio de um exemplo bastante simples que gerava ao mesmo tempo três arquivos com o mesmo conteúdo, mas em três formatos diferentes: html, pdf e rtf.

O objetivo deste novo exemplo, que pode ser visto na figura 7.3, é explorar as potencialidades das classes escritoras, criando um documento com mais detalhes, explorando recursos como estilos, cores, fontes, alinhamentos e a possibilidade de criar células que ocupem mais de uma coluna (`colspan`) para proporcionar um relatório com efeito de quebras (agrupamentos) e totalizações.

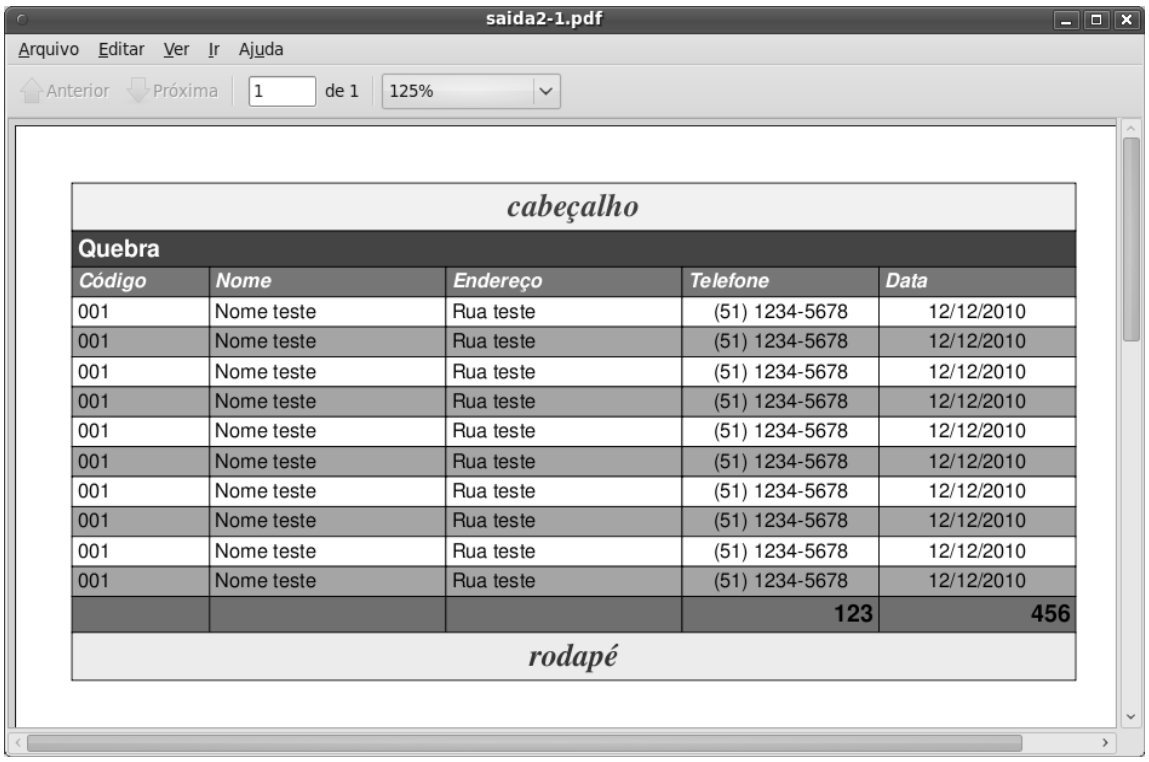


Figura 7.3 – Relatório gerado no formato PDF.

Assim como foi realizado no exemplo anterior, também neste o bloco principal do programa estará dentro de um laço de repetições cujos comandos serão executados uma vez para cada um dos formatos de arquivos: html, pdf e rtf. Ao final da execução

É importante notar que a coluna que contém o nome do fabricante não será exibida nos detalhes do relatório por não ter sido adicionada pelo método `addColumn()`. Dessa forma, o nome do fabricante somente será exibido quando alterar seu valor, configurando uma quebra no relatório e sendo exibida em uma linha de grupo. Além disso, é importante dizer que ambos parâmetros dos métodos `setGroup()` e `setTotal()` devem corresponder exatamente aos nomes que os campos têm na consulta SQL enviada ao banco de dados. Para garantir que os campos tenham esses nomes, podemos utilizar recursos como aliases de campos, por meio da cláusula "`as`" da linguagem SQL. Isso é especialmente útil quando campos de tabelas diferentes têm o mesmo nome.

Ao final, o método `generate()` é acionado, a consulta à base de dados é realizada, o relatório é gerado em memória e o método `save()` armazena o relatório no arquivo *saida5.pdf*. Caso alguma exceção ocorra durante a geração do relatório, o programa será direcionado para o bloco `catch` e a mensagem de exceção será exibida na tela. Por fim, um *link* é exibido ao usuário, por meio do qual ele poderá efetuar o *download* do arquivo contendo o relatório, que pode ser visualizado na figura 7.6.

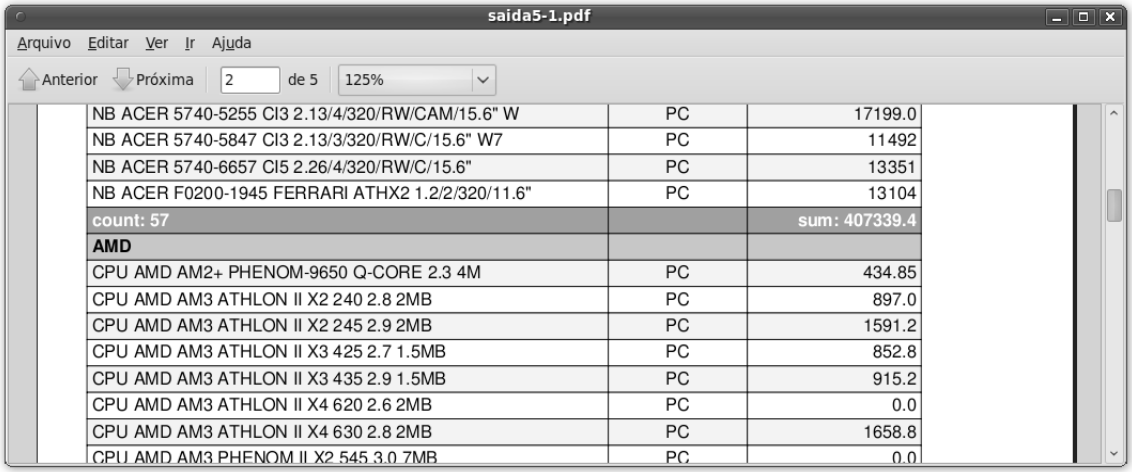


Figura 7.6 – Relatório gerado no formato PDF.

relatorio5.php

```
<?php
// inclui a biblioteca FPDF
require_once 'app.util/pdf/fpdf.php';

// inclui as bibliotecas para escrita de tabelas
require_once 'app.reports/ITableWriter.iface.php';
require_once 'app.reports/TTableWriterPDF.class.php';

// inclui a classe para geração de relatórios tabulares
require_once 'app.reports/TSimpleReport2.class.php';

// inclui as bibliotecas de acesso à base de dados
require_once 'app.ado/TConnection.class.php';
```