UNIVERSIDADE FEDERAL DO RIO GRANDE

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Framework em PHP para criação de relatórios modelados no iReport

Anderson da Silva de Souza

Rio Grande

2013

Anderson da Silva de Souza

Framework em PHP para criação de relatórios modelados no iReport

Monografia do curso de graduação em Tecnologia em Análise e Desenvolvimento de Sistemas apresentados como requisito parcial para obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Prof. Marcio Torres

Rio Grande

2013

"A primeira regra de qualquer tecnologia utilizada nos negócios é que a automação aplicada a uma operação eficiente aumentará a eficiência. A segunda é que a automação aplicada a uma operação ineficiente aumentará a ineficiência."

— Bill Gates

FOLHA DE APROVAÇÃO

Monografia sob o título “Framework em PHP para criação de relatórios modelados no iReport”, defendida por Anderson da Silva de Souza e aprovada em _____, em Rio Grande, estado do Rio Grande do Sul, pela banca examinadora constituída pelos professores:

_____

Prof. Tecnl. Márcio Josué Ramos Torres
Orientador

_____

Prof. Msc. Leonardo Vianna Nascimento
IFRS – Campus Rio Grande

_____

Prof. Dr. Eng. Carlos Rodrigues Rocha
IFRS – Campus Rio Grande

AGRADECIMENTOS

Primeiramente a Deus por ter me dado força para chegar até aqui.

A minha família por ter me dado todo suporte necessário para eu poder estudar sem mais preocupações.

A minha esposa Marcia por estar sempre ao meu lado me apoiando e incentivando.

Ao meu professor e orientador Marcio Torres por ter se dedicado muito durante todo o projeto.

E ao meu amigo Bernardo Silva que me ajudou e auxiliou no desenvolvimento do projeto desde o início.

SUMÁRIO

RESUMO

Aplicações *web* ou *softwares*, normalmente disponibilizam algum tipo de relatório para seus usuários, pois é um modo agradável de visualizar um conjunto de informações para uma determinada atividade ou até mesmo para ajudar a tomar decisões importantes dentro da empresa.

Tendo em vista a falta de um framework escrito em PHP que possibilite desenvolver relatórios no formato PDF utilizando uma ferramenta visual e a dificuldade encontrada para suprir essa necessidade, é que foi desenvolvido o SimpleReport, um framework Open Source totalmente escrito em PHP.

Com ele, é possível modelar o relatório utilizando o iReport e depois converte-lo para o formato PDF, com poucas linhas de código e tudo escrito em PHP. Existe também a possibilidade de estendê-lo para implementar novos elementos do iReport e novas fontes de dados, conforme a necessidade.

Palavras-chave: Relatórios, PDF, PHP, iReport, Framework, Web

ABSTRACT

Web applications or software usually offer some kind of report to its users. It is a nice way to visualize a collection of information for a particular activity or even to help make important decisions within the company.

Given the lack of a framework written in PHP which allows us to develop reports in PDF format using a visual tool and the difficulty to meet this need, we developed the SimpleReport, an open source framework written entirely in PHP.

With it, we can model the report using iReport and then converts it to PDF format, with a few lines of code and everything written in PHP. We also have the flexibility to extend it to implement new elements of iReport and new data sources, as needed.

Keywords: Reports, PDF, PHP, iReport, Framework, Web

LISTA DE FIGURAS

LISTA DE LISTAGENS

1 INTRODUÇÃO

É tradicional nos *softwares* e nas aplicações *web* a possibilidade de gerar documentos ou relatórios e esta cada vez mais comum essa prática, pois a tendência dos documentos é se tornarem em sua maioria eletrônicos, como aconteceu, por exemplo, com o DACTE (Documento Auxiliar do Conhecimento de Transporte Eletrônico) e a DANFE (Documento Auxiliar da Nota Fiscal Eletrônica). Para garantir a melhor qualidade e a impressão idêntica em qualquer sistema operacional ou *browser,* o formato mais utilizado é o PDF, pois os outros formatos existentes nas aplicações *web*, como o HTML, por exemplo, são renderizados de forma diferentes, dependendo de fatores como o *browser* utilizado e o sistema operacional.

Se tratando de aplicações *web* temos o PHP. [PHP, 2013] Só que para gerar documentos no formato PDF em aplicações escritas em PHP existe uma dificuldade, causada pela falta de uma solução fácil e simples para desenhar o relatório e depois gerar o documento. A solução encontrada para resolver esse problema é utilizar um servidor Java para poder usar a biblioteca JasperReport. [JasperReport, 2013] Só que nesse modelo precisa-se de uma equipe que domine no mínimo duas linguagens, o que dificulta a manutenção e encarece o projeto.

Percebendo essa brecha no desenvolvimento de relatórios ou documentos de forma fácil e simples para o PHP, teve-se a ideia de construir um *framework*[1] para simplificar esse processo.

Com ele poderá ser usado o modelador de *layout* iReport e transformar esse "modelo" em documento PDF, tudo isso com poucas linhas de código e somente utilizando o PHP. [iReport, 2013] Desse modo terá a possibilidade de criar relatórios profissionais utilizando uma ferramenta consolidada no mercado.

Esse trabalho fala do processo de desenvolvimento do SimpleReport, desde a concepção até a sua implementação, com alguns erros, acertos e projetos para o futuro.

---

[1] Um conjunto de classes que colaboram para realizar uma responsabilidade para um domínio de um subsistema da aplicação.

1.1 OBJETIVO

Temos como objetivo simplificar o modo de como é gerado os relatórios e documentos no formato PDF utilizando o PHP, criando assim, um *framework* Open Source totalmente escrito em PHP.

1.2 JUSTIFICATIVA

Criar um relatório para uma aplicação *web* PHP é um desafio para qualquer programador, pois ele encontra poucas soluções que facilitam o seu trabalho e muitos deles acabam desenvolvendo o relatório em HTML+CSS mesmo sabendo das limitações do HTML quando se trata de recursos para impressão. Para fugir dessas limitações a melhor solução é utilizar o formato PDF, que garante a fidelidade do *layout*, ou seja, o que é projetado é impresso.

Só que existe uma dificuldade para desenvolver o *layout* desses relatórios, que é a falta de uma forma fácil e simples para projetá-los para que possam ser utilizados em aplicações PHP. O SimpleReport preenche essa lacuna, gerando o relatório no formato PDF utilizando o *layout* projetado no iReport.

1.3 ORGANIZAÇÃO DO TRABALHO

Esse trabalho está dividido em 5 capítulos, a seguir uma breve descrição de cada capítulo:

- O primeiro capítulo traz uma visão geral de toda problemática encontrada na geração de relatórios no formato PDF em aplicações PHP, juntamente com o objetivo e a motivação desse trabalho.
- No segundo capítulo comenta-se sobre a geração de relatórios em aplicações *web* e suas tecnologias envolvidas, assim como o problema encontrado.
- No terceiro capítulo veremos algumas soluções existentes e suas tecnologias envolvidas.

- No quarto capítulo apresenta-se o projeto e suas implementações, todo o processo de descoberta, decisões de projeto, tecnologias, licenças, princípios e padrões envolvidos.
- No quinto e último capítulo é descrito a conclusão do projeto, onde mostraremos as vantagens, desvantagens e quais são os planos para o futuro.

2 GERAÇÃO DE RELATÓRIOS EM APLICAÇÕES WEB

Nesse capítulo veremos o que são aplicações *web*, mostraremos alguns exemplos, as vantagens e desvantagens em relação a aplicações *desktop*, junto com algumas tecnologias e conceitos utilizados em aplicações *web*.

2.1 APLICAÇÕES WEB

É muito importante que as pessoas conheçam as diferenças entre um site e uma aplicação *web*, pois muitos confundem e acreditam que tudo que está na Internet é um site.

A verdade é que esses dois conceitos estão cada vez mais próximos desde o surgimento da *Web* 2.0. [SAMPAIO, 2007] Essa tecnologia permite que os sites se tornem mais "inteligentes" e configuráveis para o usuário final. Toda essa mudança pode ter passado despercebida pelos usuários comuns, mas por trás desses sites existem sistemas complexos com regras de negócios, bancos de dados, *logs*[2], preocupações com segurança, etc.

Alguns sites parecem simples, porem necessitam de um complexo sistema por trás. A lista de sites abaixo mostra alguns exemplos:

- Google
- Gmail
- Wikipédia
- Blogger

Mesmo assim esses sites são muitos fáceis de serem utilizados pelos usuários finais, pois existem muitos estudos na área da interação humano-computadores (IHC) em especial relacionados à usabilidade·.

Juntamente com essas aplicações *web* que mais parecem sites vem crescendo também os *softwares* na *web*. Essas aplicações são mais voltadas para o

[2] É uma expressão utilizada para descrever o processo de registro de eventos relevantes num sistema computacional.

lado empresarial e se parecem mais com os *softwares* tradicionais, os *desktops*, porém com algumas vantagens:

- **Não precisa de instalação:** O sistema já esta rodando e funcionando 24 horas por dia.
- **Não precisa de um computador potente:** O sistema agora não utiliza mais o computador do cliente para fazer o processamento, em aplicações *web* isso é feito em algum computador na internet.
- **Atualizações de software instantâneas:** O Sistema tem a capacidade de se atualizar sozinho, veja um exemplo: se uma empresa grande com algumas filiais em mais de uma cidade ou pais, por exemplo, tivesse que atualizar seu software para uma nova versão, isso seria muito custoso e demorado em aplicações *desktop*. Já em aplicações *web*, é totalmente diferente, pois esse processo é muito rápido, visto que o sistema está em um único computador e só ele precisa ser atualizado.
- **Compatibilidade entre sistemas operacionais:** Os sistemas *desktop* normalmente são feitos para funcionar em uma determinada plataforma ou sistema operacional, já os sistemas *web* rodam em qualquer sistema operacional desde que você tenha acesso a Internet e um *browser*.

As aplicações *web* também possuem algumas desvantagens em relação a aplicações *desktop*, como por exemplo, a dependência da Internet. Esses sistemas necessitam que o computador esteja conectado com a Internet para que os usuários possam utilizar o serviço. Mas o fato é que existem cada vez mais sistemas *web* e praticamente todos possuem relatórios, pois é uma forma de organizar visualmente a informação de acordo com as demandas dos clientes da aplicação.

Não adianta nada utilizar, por exemplo, um sistema financeiro para controlar as despesas de uma empresa, se não conseguirmos ter a visão de quanto estamos gastando por dia ou quanto de verba disponível se vai ter para o orçamento do próximo ano. Tratando de aplicações *web*, todos os textos e tabelas que são mostrados na tela do computador, estão especificadas em HTML. Durante muito

tempo, os relatórios de sistemas *web* foram feitos em HTML. Infelizmente no HTML temos um certo “problema”: imprimir o relatório em qualquer *browser,* respeitando o *layout* projetado, pois cada navegador possui suas regras para *renderizar* os objetos e cada um interpreta o código de um jeito. Então a solução é utilizar o formato PDF (Portable Document Format), que serve para representar documentos de maneira independente do aplicativo, *browser*, hardware e do sistema operacional. Os arquivos PDF podem descrever documentos que contenham texto, gráficos e imagens em um formato independente de dispositivo e resolução.

2.2 TECNOLOGIA E CONCEITOS UTILIZADOS EM APLICAÇÕES WEB

As aplicações *web* fazem o uso de diversas tecnologias e conceitos, que as tornam confiáveis e eficientes. A seguir, algumas tecnologias e conceitos utilizados nessas aplicações.

2.2.1 World Wide Web

Mais conhecido como “www” ou simplesmente “*web*”, que traduzindo para o português seria “grande teia mundial”, é um sistema que é executado sobre a estrutura da Internet. Para visualizarmos o conteúdo contido nessa grande rede precisamos de um *browser,* que são *softwares* que se conectam aos servidores da rede, fazem uma requisição a um servidor utilizando o protocolo HTTP (*HyperText Transfer Protocol*), obtém e processa o resultado. O resultado que será mostrado na tela do computador é sempre HTML (*HyperText Markup Language*). Então podemos definir que a *web* é a junção do HTML mais o HTTP. A seguir veremos mais detalhes sobre essas tecnologias.

2.2.2 HTML

Abreviação para *HyperText Markup Language*, que significa *Linguagem de Marcação de Hipertexto*, é uma linguagem de marcação utilizada para formatar

páginas da *web*. Com ele temos a facilidade de criar páginas, e conversar com todo o mundo através de um *browser* [CARRIL, 2012]. Veja na Listagem 1 um exemplo simples de um código HTML.

```
<html>
      <head>
            <title>
            Aqui fica o título da página!
            </title>
      </head>
      <body>
      Aqui fica o conteúdo da página!
      </body>
</html>
```

Listagem 1 Exemplo de código HTML

Os documentos HTML são arquivos de texto simples e podem ser criados em qualquer editor de texto, como por exemplo, o bloco de notas, e possuem a extensão .html ou .htm.

2.2.3 HTTP

É a sigla de *HyperText Transfer Protocol* que em português significa "Protocolo de Transferência de Hipertexto". Protocolo que permite a transferência entre aplicações em redes.

É o protocolo utilizado na *web* desde 1990 para o tratamento de pedidos e respostas entre o cliente e o servidor. Quando uma requisição (pedido) de uma página *web* é feita a um servidor, ele retornará o conteúdo da página solicitada juntamente com um código de *status* HTTP em resposta à requisição. Esse código fornece informações sobre o *status* da requisição, alguns códigos de *status* comuns são:

- **200** - O servidor retornou a página com sucesso.
- **404** - A página solicitada não existe no servidor.
- **503** - O servidor está temporariamente indisponível.

Na Figura 1 temos um exemplo de como é feita essa comunicação entre o cliente e o servidor.

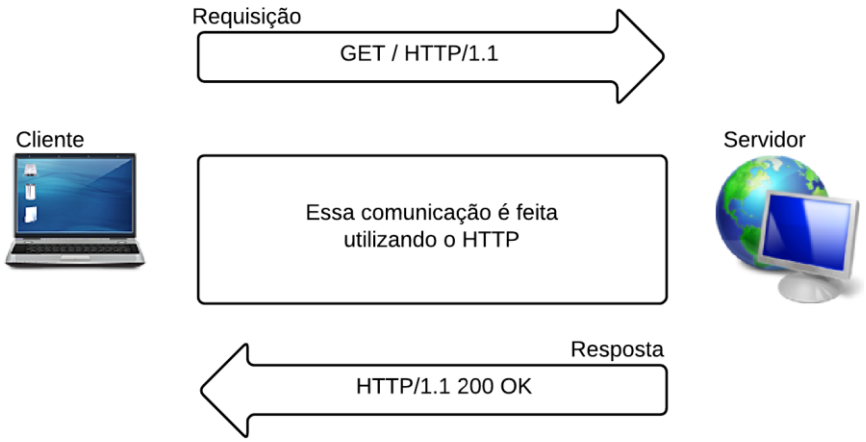

Figura 1 – Comunicação Cliente/Servidor utilizando o protocolo HTTP

2.2.4 Web 2.0

Na verdade há muitas discussões a respeito de quem realmente definiu o termo “web 2.0”. Oficialmente, esse nome foi adotado por Tim O’Reilly em 2003. [SAMPAIO, 2007]

A web 2.0 não significa uma mudança tecnológica, mas sim uma mudança no foco. Os desenvolvedores e pessoas relacionadas da área começaram a perceber que as páginas *web* deveriam se integrar mais, deixando de ser apenas páginas estáticas e começar a trocar conteúdos. Dentro desse contexto nasceram as redes sociais, blogs e o consumo de conteúdo produzido externamente, ou seja, por outros usuários. Podemos categorizar as características da *Web* 2.0 como:

- Uso da *Web* como plataforma de desenvolvimento, via APIs;
- Uso extensivo de estruturas de informação, como XML e JSON;

- Entrega assíncrona com o uso do AJAX;
- Aplicações compostas por outras aplicações - Mashups;

O importante é que não existe nenhuma grande inovação tecnológica na *Web* 2.0, apenas o reuso de tecnologias já consagradas, mas com um novo foco. [SAMPAIO, 2007]

2.2.5 Cloud Computing

A Cloud Computing, ou computação em nuvem, representa uma importante mudança no modo como nós guardamos dados e executamos aplicações. Ela possibilita que o usuário execute programas e acesse documentos de qualquer lugar do mundo e facilita a comunicação e colaboração de grupos em diferentes locais do mundo. Se pudermos resumir algumas de suas características principais, teríamos:

- Cria uma ilusão da possibilidade de recursos infinitos, acessíveis sobdemanda.
- Elimina a necessidade de adquirir e provisionar recursos antecipadamente.
- Oferece elasticidade, permitindo que as empresas usem os recursos necessários de forma dinâmica com base em sua demanda de negócios.
- O pagamento dos serviços é dado pela quantidade de recursos utilizados.

[TAURION, 2009]

2.2.6 Portable Document Format

É um formato de arquivo desenvolvido pela empresa Adobe System. A ideia é que o documento se mostre igual em qualquer computador, independente do programa ou sistema operacional.

Desenvolvido pela Adobe Systems e aperfeiçoado ao longo dos últimos 20 anos, agora o formato PDF é um padrão aberto para troca de

> *documentos eletrônicos mantido pela International Standards Organization (ISO). Quando você converte documentos, formulários, ilustrações e páginas da Web em PDF, eles ficam com a aparência exata que terão se forem impressos. Mas, ao contrário dos documentos impressos, os arquivos PDF podem conter links e botões em que você pode clicar, campos de formulário, vídeos e áudio. Também podem incluir uma lógica usada para automatizar processos corporativos de rotina. Um arquivo PDF compartilhado pode ser lido por todos com o software gratuito Adobe Reader® ou o aplicativo Adobe Reader para dispositivos móveis.* [ADOBE, 2012]

2.2.7 PHP

Significa “PHP: Hypertext Preprocessor", originalmente “Personal Home Page”. É uma linguagem interpretada de uso geral que é especialmente adequada para desenvolvimento *web*, embora exista uma extensão da linguagem chamada de PHP-GTK, para o desenvolvimento de *software desktop*. [PHP, 2013]

Com o avanço da linguagem, ela foi ganhando credibilidade entre os desenvolvedores e as empresas, principalmente com a sua versão 5 lançada em 2005, onde foram implementadas novas funcionalidades de orientação a objetos como, por exemplo, visibilidade de membros e métodos, tratamento de exceções e muitas outras novidades.

2.2.8 XML

Extensible Markup Language, é um formato de arquivo para a criação de documentos com dados organizados de forma hierárquica, que não depende de hardware ou software.

Para exemplificar a simplicidade e flexibilidade dos arquivos XML, veja o conteúdo de um arquivo XML, para armazenarmos um recado, por exemplo, na Listagem 2.

```
<?xml version="1.0"?>
<recado>
  <de>John</de>
  <para>Titor</para>
  <titulo>Information</titulo>
  <conteudo>There is something strange</conteudo>
</recado>
```

Listagem 2 - Representação de dados usando XML

Como pode ser observado no código XML apresentado na Listagem 2, trata-se de um texto simples que pode ser criado em qualquer editor de texto.

O formato XML não é recomendado para aplicações que necessitam de grandes fluxos de dados, pois ela se tornaria muito lenta e ocuparia muito espaço, por causa da sua formatação. Para aplicações com grandes fluxos existem outras formas para transferir e guardar esses dados. [Erik T. Ray, 2001]

2.2.9 Relatórios

Exposição escrita em que se descrevem todos os fatos de uma gerência, os dados colhidos numa sindicância, os trabalhos de uma comissão, etc. [Relatório, 2013]

Boa parte dos softwares geram relatórios, pois é a melhor maneira de visualizar grande quantidade de dados.

Existem vários tipos de relatórios, entre eles os operacionais e os gerenciais. Operacionais são utilizados para mostrar atividades a serem feitas por um funcionário da empresa, por exemplo. Os gerenciais servem para ajudar um gerente, por exemplo, a tomar alguma decisão a longo prazo ou fazer um planejamento para o próximo ano.

2.3 CRIAÇÃO VISUAL DE RELATÓRIOS

Como podemos perceber nesse capítulo, existem muitas tecnologias envolvidas nas aplicações *Web*, e ainda assim existe uma carência quando se trata de criar esses relatórios em PDF de forma visual, para ser utilizado em aplicações escritas em PHP.

Criar relatórios PDF utilizando PHP puro se torna cansativo, complexo e demorado, pois não existe uma ferramenta que dê a possibilidade de modelar *layouts* de forma visual (arrastando objetos) para gerar esses relatórios. O posicionamento dos objetos dentro do relatório tem que ser definido um por um, gerando uma grande perda de tempo. A Listagem 3 contém um exemplo que utiliza a biblioteca FPDF para gerar um relatório simples. [FPDF, 2012]

```
require_once("fpdf/fpdf.php");
define('FPDF_FONTPATH','fpdf/font/');
$pdf = new FPDF("P","cm",array(17.7,22));
$pdf->Open();
$pdf->AddPage();
$pdf->SetFont('Arial','',10);
$pdf->SetMargins(0,0,0);
$pdf->setY("2.25");
$pdf->setX("11.6");
$pdf->Cell(0, 0, "text");
$pdf->Output("arquivo","I");
```

Listagem 3 Exemplo de uso da Classe FPDF

Perceba que o posicionamento é feito manualmente informado a posição x e y de cada objeto dentro do relatório. Em relatórios grandes, esse código fica muito complexo e de difícil manutenção. Para resolver esse problema, a proposta é utilizar uma ferramenta que nos permite posicionar esses elementos de forma visual, arrastando e soltando componentes.

3 ESTUDO DE SOLUÇÕES EXISTENTES

Nesse capítulo veremos quais são as soluções existentes que programadores PHP utilizam para gerar seus relatórios em PDF com a possibilidade de poder utilizar um modelador de *layouts* para seus relatórios. Mas antes de conhecermos quais são essas soluções precisamos conhecer primeiro alguns conceitos básicos sobre cada tecnologia.

Primeiro veremos uma explicação sobre algumas tecnologias envolvidas nas soluções existentes. Vamos começar falando sobre a linguagem de programação Java, depois veremos o IDE[3] que irá modelar os *layouts* dos relatórios, chamado iReport. Depois disso veremos as soluções existentes e como utiliza-las:

- JasperReport
- PHP JasperXML

3.1 JAVA

É uma linguagem de programação e uma plataforma de computação lançada pela Sun Microsystems em 1995. [Java, 2012]

A linguagem Java é orientada a objetos e portável, pois independe de plataforma, ou seja, funciona em Windows, Linux, Mac OS, dispositivos móveis e até televisores.

Possui uma sintaxe semelhante a C/C++ e também um sistema inteligente para liberação de memória automática conhecida como coletor de lixo.

O Java deve toda sua portabilidade graças a sua máquina virtual disponível para diversas plataformas.

[3] *Integrated Development Environment*, no português, ambiente integrado para desenvolvimento de software.

O compilador do Java transforma o código fonte em bytecode que é lido e executado pelo JRE[4]. Vale salientar que esse processo de compilação e execução pode ser bem demorado em relação a outras linguagens como, por exemplo, a C.

Para darmos início ao desenvolvimento Java precisamos do JDK[5], que é um conjunto de classes do Java mais as classes do JRE e as ferramentas para desenvolvimento.

3.2 IREPORT

É um programa Open Source, capaz de modelar visualmente os mais complexos relatórios para aplicações Java no formato da biblioteca JasperReports, utilizando um arquivo XML com algumas particularidades.

Através de uma interface gráfica intuitiva, o desenvolvedor é capaz de criar qualquer tipo de relatório de forma simples e rápida. Essa ferramenta gera um arquivo com a extensão 'jrxml' que significa jasperReportXML e um outro arquivo compilado em Java com a extensão 'jasper', que é utilizado pelo JasperReport.

A partir de 2005 a JasperSoft (Empresa mantedora do JaspeReports) adotou o iReport como ferramenta oficial para o desenvolvimento de relatórios jasperReport. [GONÇALVES, 2009]

3.2.1 Como funciona

O relatório é dividido em diversas camadas, que no iReport são separadas por linhas horizontais, chamadas de Bands (Bandas em português), como representado na Figura 2.

[4] Java Runtime Environment
[5] Java Development Kit

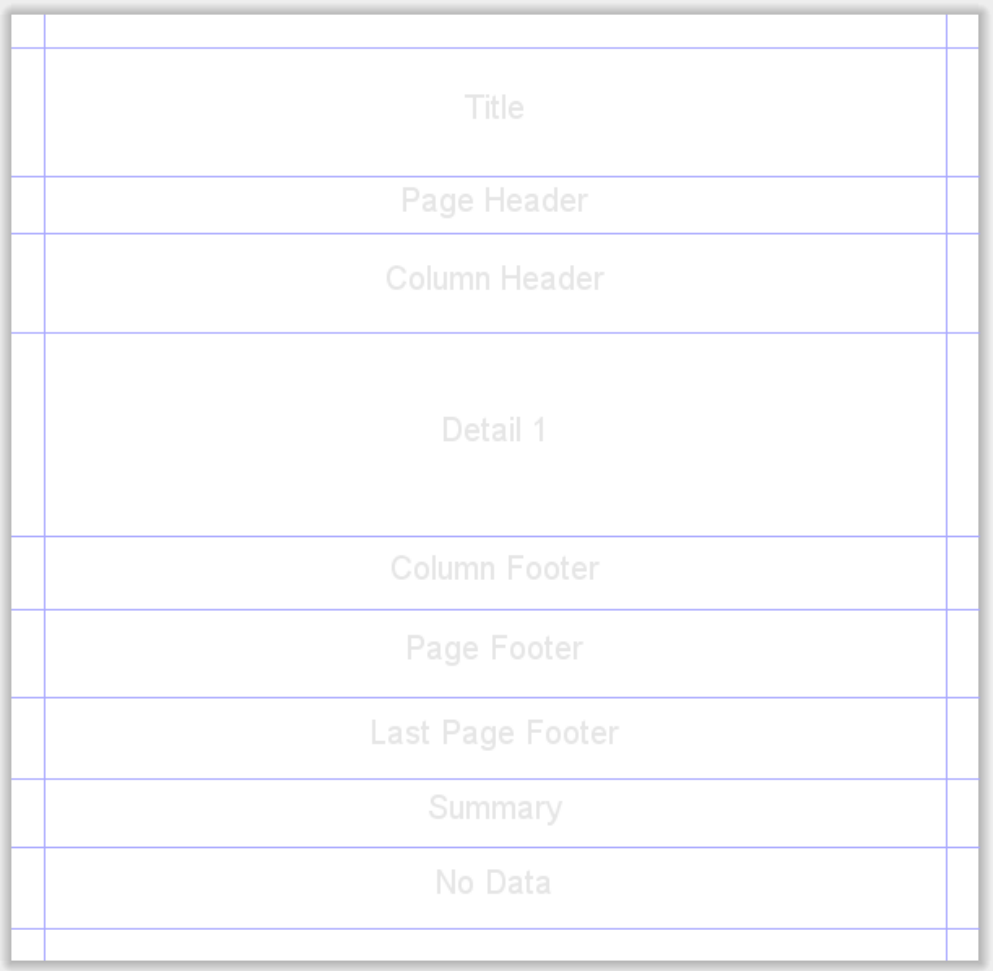

Figura 2 - Bandas disponíveis no iReport

Cada banda possui um comportamento específico e quando essas bandas se juntam com alguma fonte de dados para serem impressas, elas são impressas de diferentes maneiras e em tempos diferentes. Por exemplo, a banda ‘Title‘ é impressa apenas na primeira página de cada relatório já a banda ‘PageHeader’ será impressa no topo de todas as páginas de seu relatório. Para ver mais detalhes sobre cada banda veja o APÊNDICE A.

O relatório modelado no iReport é composto também de elementos, como o ‘Rectangle’ que é responsável por desenhar um retângulo e o ‘StaticText’ utilizado para pôr um texto. São objetos que serão posicionados dentro do relatório. Veja na Figura 3 todos os elementos disponíveis no iReport. Para ver mais detalhes sobre cada um dos elementos veja o APÊNDICE B.

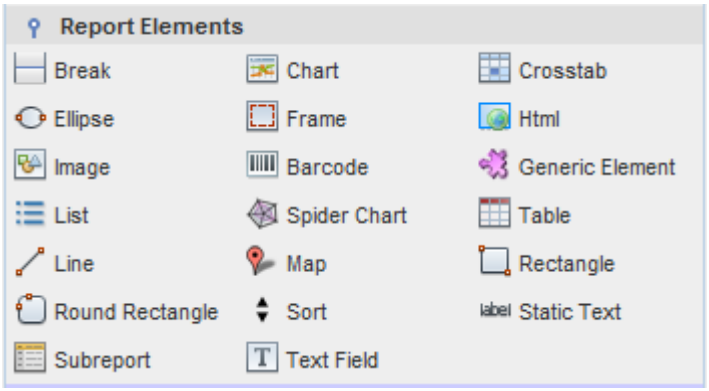

Figura 3 - Elementos disponíveis no iReport

3.3 SOLUÇÕES EXISTENTES

Existem algumas soluções para gerar relatórios em PHP de maneira visual, algumas mais simples e outras mais complexas. Essas soluções e as tecnologias envolvidas no processo serão abordadas a seguir.

3.3.1 JasperReport

É uma biblioteca escrita em Java, Open Source, projetada para ajudar o desenvolvedor com a tarefa de criar relatórios para aplicações, tanto *desktop* como *Web*, fornecendo uma API que facilita sua geração. [GONÇALVES, 2009]

Esse projeto nasceu com o objetivo de otimizar o processo de criação de relatórios gerando-os com base em arquivos XML bem estruturados, sendo no início escrito manualmente pelos programadores.

Com a necessidade de relatórios cada vez mais complexos começou a ficar inviável a construção desses arquivos manualmente, pois havia a necessidade de posicionar cada elemento um a um e de informar cada propriedade do elemento entre outras coisas. Foi então que em 09 de outubro de 2002 o italiano Giulio Toffoli desenvolveu o iReport.

3.3.2 Utilizar um servidor Java

Solução utilizada em aplicações PHP que possibilita o uso do JasperReport. Veja na Figura 4, como é feita a implementação dessa solução.

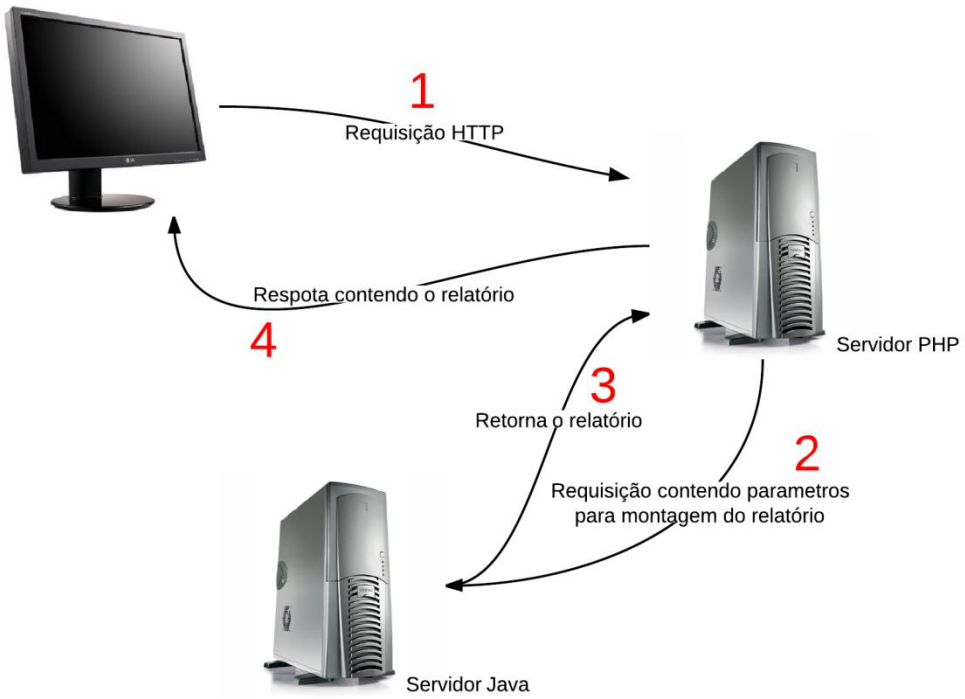

Figura 4 - Como utilizar o JasperReport em aplicações PHP

O principal problema dessa solução é que o programador fica limitado ao protocolo HTTP para fazer a comunicação entre as linguagens. E para quem tem um sistema em PHP ter que manter um servidor Java somente para responder requisições para a montagem do relatório, gera um gasto de recurso desnecessário.

Mas a grande vantagem dessa solução é poder usar 100% dos recursos disponíveis na ferramenta iReport.

Vale salientar que não é necessário ter 2 servidores físicos como mostrado na Figura 4. Essa solução pode ser implementada com apenas um servidor físico, e dois servidores *web* rodando: um para trabalhar com PHP (HTTPD LITE) e outro para trabalhar com o Java (TOMCAT), ambos respondendo em portas diferentes.

Nesse formato basta o servidor PHP fazer uma requisição para o servidor Java e obter a resposta.

3.3.3 PHP JasperXML

É uma solução de código aberto de uma empresa da Malasya chamada de Sim IT. Resumindo é uma classe escrita em PHP para facilitar a criação de relatórios. Esses relatórios são gerados em PHP e XLS, pois independem de plataforma para sua visualização e impressão.

Segundo a própria empresa esse projeto encontra-se parado, ou seja, sem desenvolvedores. [PHP JasperXML, 2012]

Para utiliza-lo é necessário primeiro modelar seu relatório no iReport e depois editar o arquivo “setting.php” para definir algumas configurações:

```
<?php
$server="localhost";
$db="phpjasperxml";
$user="root";
$pass="root";
$version="0.6d";
?>
```

Listagem 4 Arquivo de configuração do PHPJasperXML

Depois basta instanciar a classe `PHPJasperXML` e definir algumas propriedades, veja o exemplo na Listagem 5.

```
<?php
require_once('class/fpdf/FPDF.php');
require_once('class/PHPJasperXML.inc');
require_once('setting.php');
```

```
$xml = simplexml_load_file('report.jrxml'); // arquivo criado no iReport
$PHPJasperXML = new PHPJasperXML();

$PHPJasperXML->arrayParameter = array("descricao"=>$_GET["descricao"]);
$PHPJasperXML->xml_dismantle($xml);
$PHPJasperXML->connect($server,$user,$pass,$db);
$PHPJasperXML->transferDBtoArray($server,$user,$pass,$db);
$PHPJasperXML->outpage("I");
?>
```

Listagem 5 Exemplo do PHPJasperXML

4 PROJETO E IMPLEMENTAÇÃO

Nesse capítulo iremos falar sobre o processo de descoberta, decisões de projeto, análise e desenvolvimento. Falaremos também quais tecnologias, licença, princípios e padrões foram utilizados e a justificativa do uso de cada uma.

4.1 TECNOLOGIAS, PRINCÍPIOS E PADRÕES.

O SimpleReport foi desenvolvido utilizando o IDE Eclipse Índigo, o Google Code para armazenar, centralizar e disponibilizar o projeto. Como servidor *web* o Apache 2.2, com a linguagem PHP 5.4.3, o banco de dados MySQL Server 5.5.14 e para desenhar os relatórios foi utilizado o iReport versão 5.0.0. A seguir uma breve descrição sobre alguns princípios, padrões e tecnologias.

4.1.1 Design Patterns

Em português significa padrões de projeto. Eles nomeiam, abstraem e identificam os aspectos chave de uma estrutura de projeto comum para torná-lo útil para a criação de um projeto orientado a objetos reutilizáveis. [GAMMA e ERICH, 2000]

4.1.2 Convention Over Configuration

É um modelo de desenvolvimento de software (também conhecido como Convenção sobre configuração) que visa diminuir o esforço do programador, ganhando simplicidade no código, mas sem perder a flexibilidade. A ideia é que o programador codifique apenas o mínimo necessário para concluir seu objetivo, sem tem que se preocupar com o resto do problema.

4.1.3 Iterator

É um padrão de projeto que oferece uma forma de acessarmos sequencialmente os elementos de uma coleção de dados, sem expor sua representação interna.

4.1.4 MySQL

É um SGBD (Sistema de Gerenciamento de Banco de Dados), que utiliza a linguagem SQL (Linguagem de Consulta Estruturada, do inglês Structured Query Language) como interface. É o mais popular banco de dados em código aberto. Veja algumas de suas características:

- Fácil de usar
- Boa escabilidade e desempenho
- Suporte a produção

[MySQL, 2013]

4.1.5 FPDF

É uma classe escrita em PHP que permite gerar arquivos PDF. Essa classe possui dezenas de funções para facilitar e agilizar o desenvolvimento de arquivos PDF e possui algumas vantagens tais como:

- Escolha da medida de formato de página da unidade, e as margens;
- Quebra de página automática;
- Quebra de linha automática e justificação do texto;
- Suporte de imagem (JPEG, PNG e GIF);
- Cores;
- Links;

[FPDF]

4.1.6 LGPL

Significa GNU Lesser General Public License. Foi utilizada essa licença, pois ela possui algumas liberdades a mais que a GPL. Veja mais detalhes no ANEXO A.

4.2 PROJETO

O SimpleReport foi todo projetado para dar maior flexibilidade à quem for utiliza-lo. Ele suporta desde estruturas muitos simples até estruturas complexas.

Assim como o JasperReport, o SimpleReport foi projetado para funcionar sem a necessidade de um modelador de *layouts*, ou seja, não é necessário ter um arquivo XML para gerar seu relatório. Obviamente não fará muito sentido o seu uso sem uma ferramenta que facilite o seu desenvolvimento, mas saiba que é possível.

É nítida a semelhança entre essas duas bibliotecas, pois foi aproveitada a mesma ideia, principalmente para facilitar o uso de quem já esta acostumado com a outra biblioteca. Veja nas figuras 5 e 6 essa semelhança.

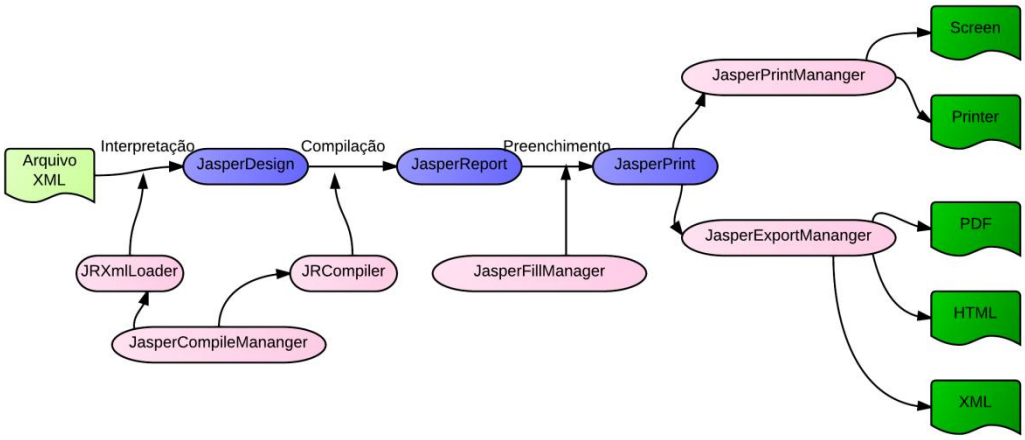

Figura 5- Estrutura do JasperReport

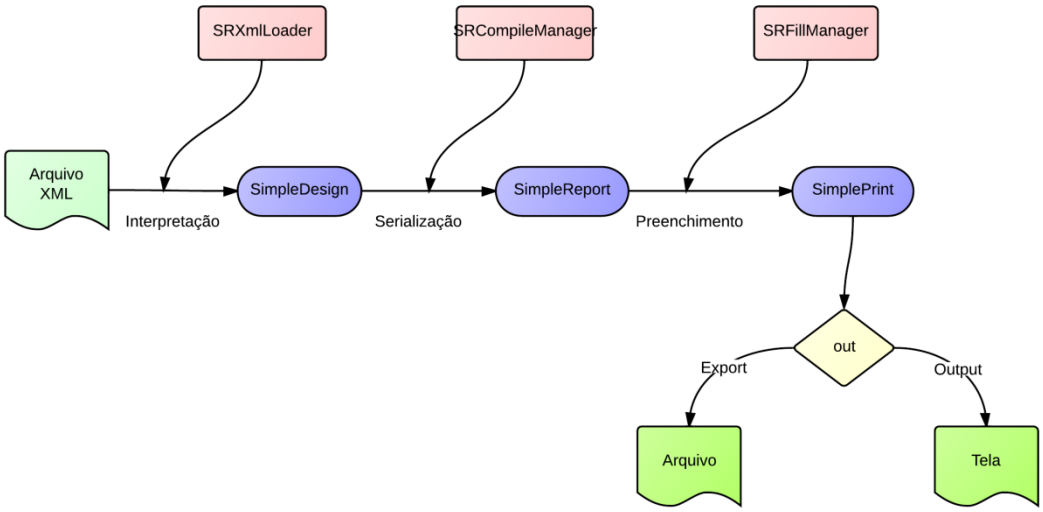

Figura 6 - Estrutura do SimpleReport

O SimpleReport é composto por três classes principais, são elas;

- **SimpleDesign:** Essa classe é a representação do modelo do relatório. Nela serão definida todas as propriedades, bandas, elementos e etc.
- **SimpleReport:** Essa classe representa o SimpleDesign serializado, apenas com as propriedades definidas e já interpretadas.
- **JasperPrint:** Essa classe é responsável por gerar o relatório, ela precisa de uma instância da SimpleReport e/ou uma fonte de dados. Ela possui dois métodos para gerar o relatório um é o *output* que mostra o relatório na tela e o outro é o *export* que faz um download do arquivo.

No APENDICE C você encontrará a descrição de outras classes da estrutura do SimpleReport. Veja na Figura 7 como estão estruturados os diretórios do projeto.

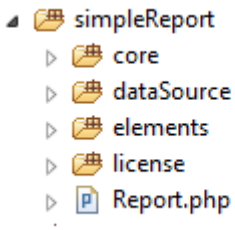

Figura 7 - Estrutura dos diretórios do SimpleReport

- **core**: Classes da estrutura da biblioteca
- **dataSource**: Classes que serão utilizadas como fontes de dados
- **elements**: Classes dos elementos
- **license**: Arquivo com dados referentes a licença da biblioteca

O JasperReport tem três classes principais, como pode-se observar na Figura 5. De forma similar, o SimpleReport também tem três classes principais com responsabilidades análogas às do JasperReport.

A Figura 6, demonstra que o JasperReport também possui três classes principais, cuja suas responsabilidades são muito parecidas com as do SimpleReport.

- **JasperDesign**: Representa a definição do relatório. A partir de um template XML é criado um JasperDesign.
- **JasperReport**: Representa o JasperDesign compilado. O processo de compilação verifica o design do relatório e compila o design em um JasperReport.
- **JasperPrint**: Representa o relatório gerado. É criado um JasperPrint a partir de um JasperReport, contendo o relatório preenchido.

Mais detalhes sobre as classes do JasperReport estão no Apêndice D.

Nesse projeto foi utilizado alguns princípios como o CoC para facilitar e simplificar o uso da biblioteca com algumas configurações pré definidas. Também foi utilizado o padrão *Iterator,* para dar suporte a extensão para novas fonte dados. Desse modo o SimpleReport se preocupa em fazer a iteração com uma coleção de

dados, sem saber na verdade como o usuário irá criar essa coleção de dados. Outra decisão tomada foi a licença que seria utilizada no projeto, foi escolhida a LGPL, pois dá total liberdade a quem for usa-la, podendo ser utilizada até mesmo em softwares proprietários.

4.3 IMPLEMENTAÇÃO

Para implementar o SimpleReport foram utilizados alguns padrões de projeto e boas práticas de programação, como por exemplo *Convention Over Configuratio*n, *Iterator*, *Factory static*, *Fail/Fast*.

O exemplo de *Convention Over Configuration* em nosso projeto é a facilidade de criar seu relatório, veja no código abaixo o exemplo de uso do SimpleReport e perceba como é simples e livre de qualquer configuração:

```
require_once 'simpleReport/Report.php';
Report::from('report.jrxml')->outPut();
```

Listagem 6 Exemplo para mostrar a simplicidade do SimpleReport

Com apenas duas linhas de código, o SimpleReport será capaz de reproduzir seu relatório modelado no iReport.

Para o uso do padrão *Iterator*, foi criada uma interface chamada de `ISRIterator`, onde dá ao usuário a possibilidade de estender o SimpleReport e criar suas próprias fontes de dados.

Tendo essa estrutura definida, ficou mais fácil para manipular os dados. Desse modo no momento de ler os dados para montar o relatório, basta fazer o que está na Listagem 7.

```
// $mysql = instância da classe MySQL que implementa a ISRIterator
while($r = $mysql->next()){
	// $r = array associativo com o nome do campo e o valor
}
```

Listagem 7 Implementação do SimpleReport para ler as fontes de dados

Para simplificar o uso de sua classe, utilizamos outro padrão, o Static Factory. Para isso é necessário utilizar a classe `SRDataSource`. Ela é responsável por retornar uma instância de sua classe através do nome dela, utilizando o método estático `getInstance`.

```
class SRDataSource{
	public static function getInstance($class, $result){
		$file = 'simpleReport/dataSource/'.$class.'.php';

		if(!file_exists($file))
			die('Arquivo não encontrado: '.$file);

		require_once $file;
		return new $class($result);
	}
}
```

Listagem 8 Classe SRDataSource

Veja abaixo um exemplo do uso dessa classe para criar relatórios com dados na Listagem 9.

```
require_once 'simpleReport/core/SRDataSource.php';
require_once 'simpleReport/Report.php';
$link_identifier = mysql_connect('localhost', 'user', 'password');
$db_selected = mysql_select_db('tads', $link_identifier);
$result = mysql_query('select * from alunos', $link_identifier);

Report::from('docs/example2.jrxml',
SRDataSource::getInstance('MySQL', $result))->outPut();
```

Listagem 9 Exemplo do SimpleReport com banco de dados.

Um princípio utilizado foi o *Fail-Fast*, onde qualquer erro encontrado, tanto no processo de interpretação, serialização ou preenchimento do relatório, causa uma falha explícita no código.

O SimpleReport possui uma estrutura de cache, para não ter que interpretar todo arquivo XML cada vez que um novo relatório for gerado. Quando é executado pela primeira vez é criado no mesmo diretório no arquivo XML um arquivo SR. É fortemente recomendado que seja utilizado esse arquivo nas próximas vezes que for

gerado esse relatório, pois toda a estrutura do relatório já foi interpretada. Veja um exemplo que como utilizar a estrutura de cache na Listagem 10.

```
$fileName = substr('nomeDoRelatorio.jrxml', 0, -6);
// verifica se já existe um arquivo SR
if (file_exists($fileName.'.sr')) {
$this->report=SRInstanceManager::getInstance(file_get_contents($fileName.'.sr');
} else {
$load = new SRXmlLoader();
$this->design = $load->load($sourceFileName);
$this->report = SRCompileManager::compile($this->design, $fileName);
}
```

Listagem 10 Exemplo para utilizar o cache do SimpleReport.

Por padrão o SimpleReport já verifica se existe um arquivo de cache para utilizá-lo.

4.4 COMO ESTENDER O SIMPLEREPORT

O SimpleReport dá para o usuário a possibilidade e estende-lo facilmente. Ele terá a liberdade de criar novos elementos e novas fontes de dados, sem precisar alterar uma linha no cerne da biblioteca.

Para implementar novos elementos, deve-se criar uma classe com o mesmo nome do elemento no iReport, estender a classe `SRElements`, implementar os métodos `fill` e `draw`. O método `fill` é responsável pelo preenchimento do elemento. O método `draw` é responsável pelo desenho do elemento. Desse modo cada elemento é responsável por se preencher e se desenhar. Veja a seguir um exemplo do elemento "Image" implementado na Listagem 11.

```
require_once 'simpleReport/core/SRElements.php';
class Image extends SRElements{
	public $imageExpression;
	public $extension;
	public function fill(SimpleXMLElement $xml){
		foreach ($xml as $elementName => $element){
			switch($elementName){
```

```
                        case 'reportElement':
                               $this->x = (String)$element['x'];
                               $this->y = (String)$element['y'];
                               $this->width = (String)$element['width'];
                               $this->height = (String)$element['height'];
                               break;
                        case 'imageExpression':
                               $nameFile = str_replace('"', '',
(String)$element);
                               $this->extension = substr($nameFile, -3);
                               $this->imageExpression = $nameFile;
                               break;
                    }
             }
       }
       public function draw(&$pdf){
       $pdf->Image($this->imageExpression,$this->x,$this->y,
                    $this->width,$this->heigh,$this->extension);
       }
}
```

Listagem 11 Classe Image

Depois de criada a classe, deve-se colocá-lo dentro do diretório “/elements”. Para criar novas fontes de deve-se implementar a interface `ISRIterator` e escrever o método `next` de modo que você quiser. Veja um exemplo da classe MySQL para fazer iteração com a SimpleReport.

```
require_once 'simpleReport/core/ISRIterator.php';
class MySQL implements ISRIterator{
       var $result;
       function __construct($result){
              $this->result = $result;
       }
       function next(){
              return mysql_fetch_array($this->result, MYSQL_ASSOC);
       }
}
```

Listagem 12 Classe MySQL

Depois de criar a classe, deve-se colocá-lo dentro do diretório “/datasource”.

Mas nem todos os elementos e fontes de dados serão capazes de ser implementados. Veja a seguir alguns problemas encontrados e algumas implementações pendentes.

4.5 IMPLEMENTAÇÕES PENDENTES

Durante o processo de implementação, perceberam-se algumas falhas no projeto que nos impediu de implementar certas funcionalidades, como por exemplo:

- Sub-elementos (ou seja, elementos que compõem elementos);
- Obter a própria query interna do iReport;
- Utilizar fontes de dados como XML e CSV;

No iReport existem elementos, como o *Frame,* que são agrupadores de outros elementos, e esse é um exemplo de elemento não suportado pelo SimpleReport. Para suportá-lo deveríamos ter utilizado um padrão chamado de *Decorator.* Com esse padrão teríamos por exemplo, a capacidade de inserir elementos dentro de elementos com um método, chamado de `addElements`, e quando fossemos desenhar um elemento, teríamos que primeiro desenhar todos os filhos, e se os filhos tivessem outros filhos dentro, seria a mesma postura, ou seja, deveria ser recursivo, tanto para desenhar o elemento quanto para preencher ele.

Também no iReport temos a possibilidade de inserir uma *query*, só que isso não é suportado pelo SImpleReport, pois não é compatível com a estrutura criada do *Iterator,* pois do modo que está projeto e implementado, o SimpleReport não é responsável por fazer a conexão e executar a *query*. Essa responsabilidade é do usuário. Ele simplesmente recebe um *result* de uma SQL. Para solucionar esse problema deveríamos ter utilizado a classe PDO (PHP Data Object). Com ela, teríamos a possibilidade de executar uma query independentemente do SGBD utilizado pelo usuário.

E outro erro no projeto foi a classe `SRDataSource`, que não nos permite utilizar outra fonte de dados que não seja um SGBD. Esse problema ainda carece de

solução. Todos esses erros de projeto só foram percebidos no momento da implementação.

4.6 COMO USAR O SIMPLEREPORT

Para ver detalhes de como usar o SimpleReport veja o APÊNDICE E.

5 CONCLUSÃO

Percebemos que o SimpleReport é uma ideia muito proeminente e funciona razoavelmente bem e existe essa brecha no desenvolvimento visual de relatórios PDF para a linguagem PHP. Vamos continuar a implementação dele e corrigir os erros de projeto, para dar maior flexibilidade e segurança para os usuários. Podemos citar algumas vantagens e desvantagens do SimpleReport:

Vantagens

- Desenvolver relatório em PDF com iReport.
- Rodar nativamente em PHP.
- Código aberto.
- Enviar parâmetros para o relatório facilmente.

Desvantagens

- Não suporta todas as funcionalidades do JasperReport
- Não interpreta código Java incorporado dentro do iReport

A ideia é de disponibiliza-lo e incentivar novas pessoas a contribuir no projeto, através de blogs e grupos na Internet.

Queremos no futuro, desenvolver o próprio modelador de telas, e perder o vinculo do iReport. Desse modo o projeto irá se tornar independente e poderão ser criados novos elementos e regras.

Se alguém estiver motivado com o projeto. Basta acessar o site code.google.com/p/simplereport/.

REFERÊNCIAS

ADOBE. Disponível em http://www.adobe.com/br/products/acrobat/adobepdf.html. Acesso em: 11 jun 2012.

CARRIL, Marly. **HTML Passo a Passo,** 2012.

Erik T. Ray. **Learning XML.** First Edition, January 2001.

FPDF. Disponível em http://www.fpdf.org. Acesso em 14 jun 2012.

GAMMA, Erich et al. **Padrões de projetos, soluções reutilizáveis de software orientado a objetos**. Porto Alegre Bookman, 2000.

GONÇALVES, Edson. **Desenvolvendo Relatórios Profisionais com iReport para Netbeans IDE**. Rio de Janeiro: Editora Ciência Moderna Ltda, 2009.

iReport. Disponível em http://community.jaspersoft.com/project/ireport-designer. Acesso em 12 de fev 2013.

JasperReport. Disponível em http://community.jaspersoft.com/project/jasperreports-library. Acesso em 12 de fev 2013.

Java. Disponível em http://www.java.com/pt_BR/download/faq/whatis_java.xml. Acesso em 14 jun 2012.

MySQL. Disponível em http://www.oracle.com/br/products/mysql/index.html. Acesso em 20 de jan 2013.

PHP. Disponível em http://php.net . Acesso em 12 de fev 2013.

PHP JasperXML. Disponível em http://www.simit.com.my/?q=PHPJasperXML. Acesso em 14 jun 2012.

Relatório. Disponível em http://www.priberam.pt/dlpo/Default.aspx?pal=relat%C3%B3rio. Acesso em 31 de jan 2013.

SAMPAIO, Cleuton. **Web 2.0 e Mashups - Reinventando a Internet.** Editora Brasport, 2007.

TAURION, Cezar. **Cloud Computing - Computação em Nuvem.** Rio de Janeiro : Brasport, 2009.

APÊNDICE A: Bandas de relatório do iReport.

- **Title:** Essa é a primeira banda visível na impressão do seu relatório, e aparecerá apenas na primeira página.

- **Page Header**: Essa banda aparece em todas as páginas, sempre no início do seu relatório, exceto na primeira página, que ficará abaixo da banda Title. A altura especificada durante a fase de desenho do seu relatório não vai ser alterada (exceto que haja componentes redimensionáveis, como por exemplo, um sub-relatorio ou um texto muito longo)[6]

- **Column Header**: Essa banda só aparece no começo de cada interação com a banda 'detail'. Normalmente é utilizada para pôr nome nas colunas do seu relatório.

- **Detail**: Essa banda é onde ficará a exibição de seus dados. Ela vai se repetir até preencher toda a página, levando em consideração os espaços das próximas bandas. Se a página chegar no final e não foram impressos todos os dados, uma nova página será criada e o relatório começará a imprimir tudo novamente começando todo o fluxo novamente.

- **Column Footer**: Essa banda só aparece no final de cada interação como a banda 'detail', ela tem comportamento semelhante com a banda column header.

- **Page Footer**: Essa banda representa o rodapé da página do seu relatório, ela será impressa no final de cada página, exceto na última página, onde é impresso a banda Last Page Footer em seu lugar. É muito utilizada para informações como o número de páginas, uma data ou até mesmo o logo de sua empresa.

- **Last Page Footer**: Essa banda substitui o rodapé das páginas regulares na última página de seu relatório. Se você precisar de um rodapé diferenciado no final de todas as páginas do seu relatório, você vai precisar utilizar essa

[6] Não serão implementados componentes redimensionáveis

banda. Se você não definir uma altura para ela, não será interpretada e todos os rodapés de seu relatório serão iguais aos definidos na banda Page Footer.

- **Summary**: Essa banda é conhecida como o rodapé do relatório. Ela será impressa apenas na última página logo abaixo da banda Column Footer. É muito comum utilizar essa banda para pôr algumas informações como um total geral do relatório ou até mesmo um gráfico.

- **Background**: Essa banda dá a possibilidade de você por uma marca d'agua em seu relatório, ou qualquer outra coisa de efeito similar.

- **noData**: Essa banda é utilizada quando a fonte de dados utilizada no seu relatório está vazia. Você pode por uma mensagem personalizada, por exemplo, em caso de não houver dados. Você pode por imagens e o que quiser nessa banda.

APÊNDICE B: Elementos de relatório do iReport.

- **Break**: Quebra uma página a partir dele.
- **Chart**: Gera um gráfico a partir de valores determinados.
- **CrossTab**: Desenvolve o que chamamos em programas de planilhas eletrônicas de tabelas dinâmicas (referencia cruzada). É muito comum utilizar em relatórios gerenciais.
- **Elipse**: Desenha elipses no relatório.
- **Frame**: Cria quadros que podem conter outros elementos em seu interior. É muito utilizado para criar blocos dinâmicos em seu relatório.
- **HTML**: Permite utilizar marcações HTML em seu relatório.
- **Image**: Utilizado para exibir imagens em seu relatório. Podem ser imagens dinâmicas (preenchidas com alguma fonte de dados, por exemplo) ou estáticas, definindo diretamente o caminho do diretório da imagem.
- **BarCode**: Permite utilizarmos códigos de barras nos relatórios
- **GenericElement**: Permitem manipuladores personalizados para serem conectados em determinado momento da exportação do relatório
- **List**: Possibilita criar outra query no mesmo relatório, ou seja, uma nova consulta.
- **Spider chart**: Gráfico processado usando a biblioteca JFreeChart
- **Table**: Usado para criar tabelas.
- **Line**: Usado para inserir linhas no relatório.
- **Map**: Usado para mostrar mapas do Google Maps. Esse elemento possui dois parâmetros de configuração bem específicos que são latitude e longitude.

- **Rectangle**: Usado para criar retângulos. Contem uma cor no background e uma cor na borda. Normalmente é utilizado para separar elementos ou dar destaque em alguma informação.

- **Round Rectangle**: Possui as mesmas características do Rectangle, porem suas bordas são arredondadas.

- **Sort**: Usado para ordenar uma coluna dinamicamente.

- **Static Text**: Representa um texto estático no relatório, pode ser texto de uma linha ou de múltiplas linhas, mas sempre estático.

- **SubReport**: Cria um relatório dentro do outro. Isso é mais conhecido como *master-detail,* onde possuímos um relatório pai e ‘n’ relatórios filhos.

- **TextField**: Utilizado para criar os campos dinâmicos dos relatórios. É neste elemento que você se conecta a um determinado campo do banco de dados para exibir suas informações, por exemplo.

APÊNDICE C: Descrição detalhada de algumas classes importantes da estrutura do SimpleReport

- **Report**: Essa classe é utilizada para facilitar a criação do seu relatório.

- **SRXmlLoader**: Responsável por interpretar o arquivo XML e retornar uma instância da classe SimpleDesign.

- **SRInstanceManager**: Essa classe transforma um arquivo do simpleReport interpretado, cuja sua extensão é '.sr', em uma instância da classe SimpleReport.

- **SRFillManager**: Essa classe é utilizada para gerar instâncias da classe SimplePrint. Em seu processamento, ela utiliza uma fonte de dados (SRDataSource) e uma instância da classe SimpleReport.

- **SRCompileManager**: Responsável por criar instâncias das classes SimpleReport e de criar um arquivo com o conteúdo serializado, que será utilizado como cache, para otimizar o processo de geração do relatório.

- **SRElements**: Classe padrão para todos os elementos. Todos deverão estender essa classe.

- **SRParameter**: Classe responsável por gravar e obter os parâmetros no momento da geração do relatório.

APÊNDICE D: Descrição detalhada de algumas classes importantes da estrutura do JasperReport.

- **dori.jasper.engine.design.JasperDesign**: Instâncias dessa classe representam o relatório no seu formato mais primitivo. São resultados de um processamento sobre o arquivo XML.

- **dori.jasper.engine.JasperReport**: Instâncias dessa classe representam relatórios compilados. Nesse estágio, toda a análise sintática nas expressões existentes no XML já foi realizada. Objetos dessa classe podem ser armazenados em arquivos.*jasper*.

- **dori.jasper.engine.JasperCompileManager**: Responsável por criar instâncias das classes JasperReport e JasperDesign.

- **dori.jasper.engine.JasperPrint**: Consiste no JasperReport com todos os campos preenchidos. Um objeto dessa classe pode ser visualizado diretamente utilizando visualizadores internos do JasperReport, como também pode ser transformado em formatos mais populares como HTML, XML ou PDF.

- **dori.jasper.engine.JRDataSource**: Essa interface padroniza o comportamento das classes que manipulam as fontes de dados necessárias durante o preenchimento dos campos existentes no JasperReport. Dessa forma, várias fontes podem ser utilizadas, seja um banco de dados, ou mesmo um arquivo XML.

- **dori.jasper.engine.JasperFillManager**: Essa classe é utilizada para gerar instâncias da classe JasperPrint. Em seu processamento, ela utiliza uma fonte de dados (JRDataSource) e uma instância da classe JasperReport.

- **dori.jasper.engine.JasperPrintManager**: Permite imprimir o relatório (o conteúdo do relatório é enviado para impressora, uma janela de requisição de impressão é aberta). Tanto é possível imprimir todo o relatório como também páginas do mesmo. Além disso, é possível imprimir o relatório como uma imagem (utilizando o método printPageToImage).

- **dori.jasper.engine.JasperExportManager**: Permite gerar documentos nos formatos PDF, HTML e XML (versão 1.0). Com o tempo, novos formatos serão incorporados.

APÊNDICE E: Como usar o SimpleReport.

Primeiro é necessário obter o SimpleReport no Google Code. Para isso, acesse o seguinte link http://code.google.com/p/simplereport .

Será necessário também o iReport. Para fazer o download acesse o seguinte link http://sourceforge.net/projects/ireport/files/iReport/iReport-5.0.0/. Foi utilizado a ultima versão até o momento que é 5.0.0, mas o SimpleReport funciona em versões anteriores.

Assim que terminar a instalação do iReport e abri-lo, terá a Figura 8.

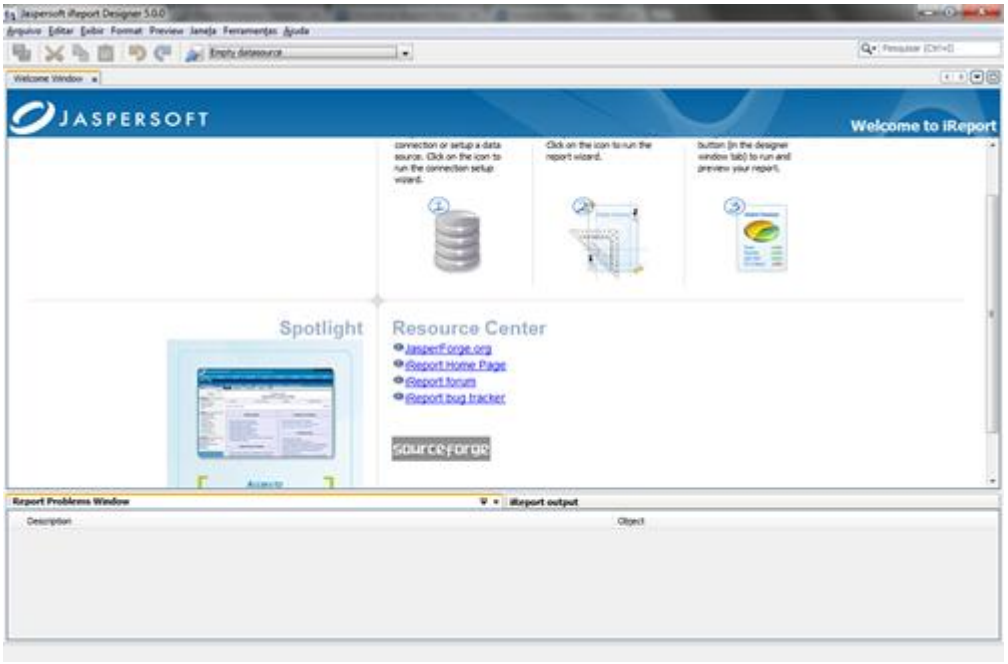

Figura 8 - Tela inicial do iReport

Para criar um novo projeto deve-se acessar o menu Arquivo>Novo. Essa opção irá abrir uma janela com varias opções. Selecione o arquivo “Blank A4” para começarmos um relatório novo. Depois clique no botão ‘Open this template’, que irá abrir uma nova janela para informar o nome do relatório e o local. Preencha essas informações e clique em próximo e depois Finalizar.

Desse modo o projeto já esta criado, agora deverá aparecer uma tela igual à Figura 9.

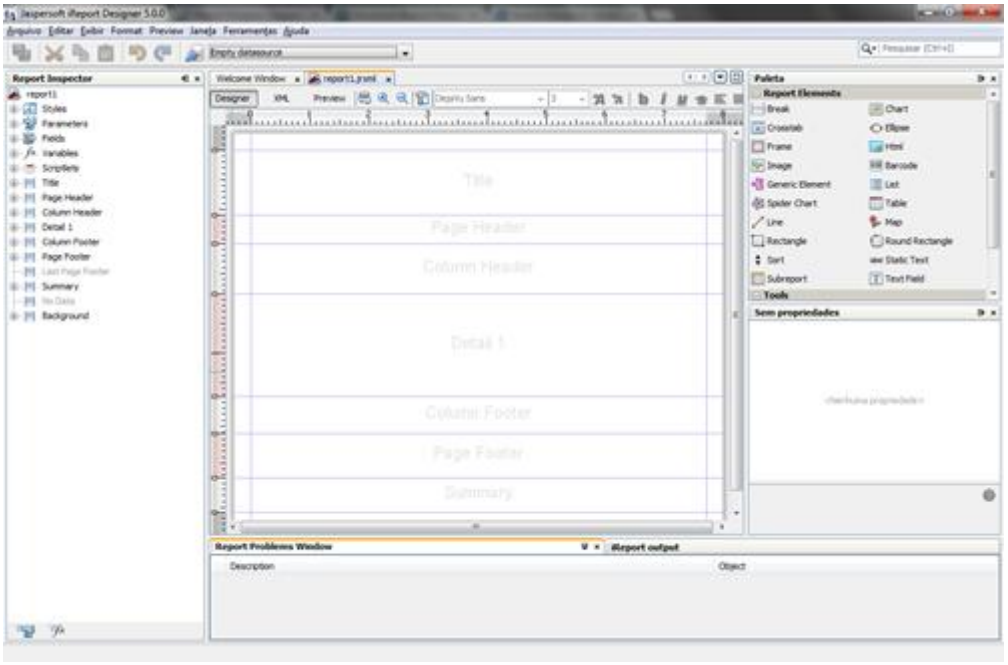

Figura 9 - iReport com novo relatório vazio

Se começar a modelar o relatório. No menu ‘Report Inspector’ à esquerda temos os *parameters*, *fields* e as *bands*. Se clicarmos em algum item, irão aparecer às propriedades dele no canto inferior direito no menu ‘Propriedades’. Já no canto direito superior temos a paleta com os elementos disponíveis. Vale ressaltar que o SimpleReport ainda não suporta todos os elementos disponíveis no iReport.

Para começar o primeiro exemplo bem simples, iremos utilizar o elemento *Static Text.* Basta clicar em cima dele e arrastar para posição desejada no relatório. Então teremos uma imagem parecida com a Figura 10.

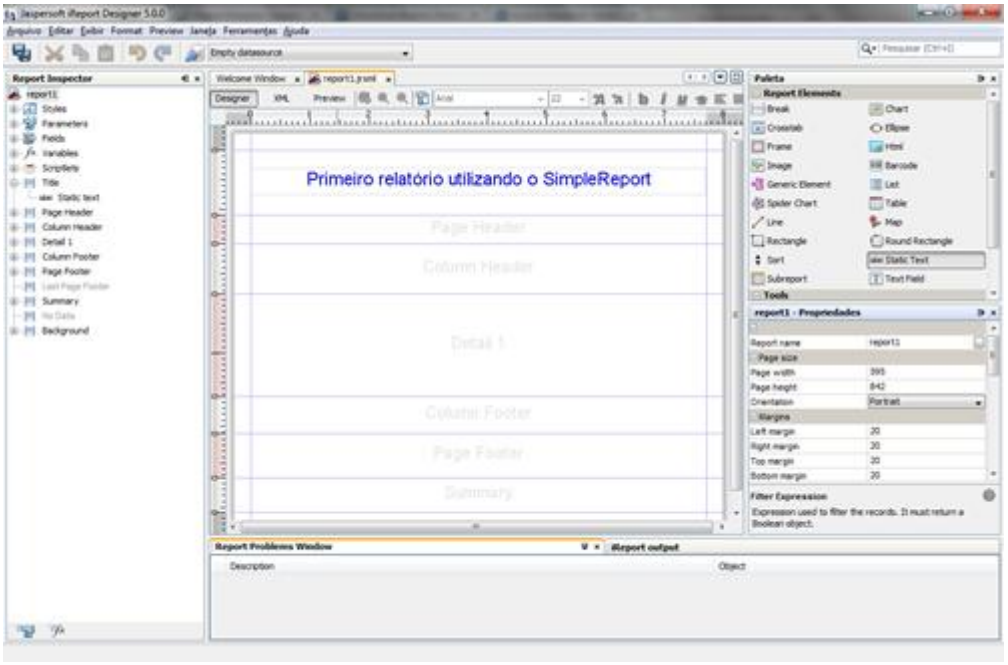

Figura 10 - Exemplo no iReport com elemento static text

Perceba que foi alterado o tamanho e a cor da fonte, e o texto está centralizado, dentro da banda *Title*. Vamos supor que o relatório já esta todo modelado com todos os textos que serão necessários, e agora vamos utilizar o SimpleReport para transforma-lo em PDF.

Para transformar o relatório modelado no iReport em relatório em PDF, deve-se criar um novo arquivo php e pôr o mesmo conteúdo da Listagem 13, onde o arquivo no formato ".jrxml", é o modelo do relatório criado pelo iReport.

```
require_once 'simpleReport/Report.php';
Report::from('report.jrxml')->outPut();
```

Listagem 13 Transformado o *layout* do iReport em PDF

Feito isso, será gerado o relatório PDF utilizando o SimpleReport. Mas esse exemplo foi simples. Faremos um relatório um pouco mais elaborado agora. Vamos supor que desejamos colocar um texto dinâmico no relatório, como por exemplo, uma data ou um valor passado por GET, POST ou até mesmo do banco de dados. Uma boa forma para fazer isso é utilizarmos os parâmetros. Para adicionar um novo parâmetro ao relatório, utilize o item *Parameters,* clique com o botão direito e clique em *Adicionar Parameter,* assim como a Figura 11.

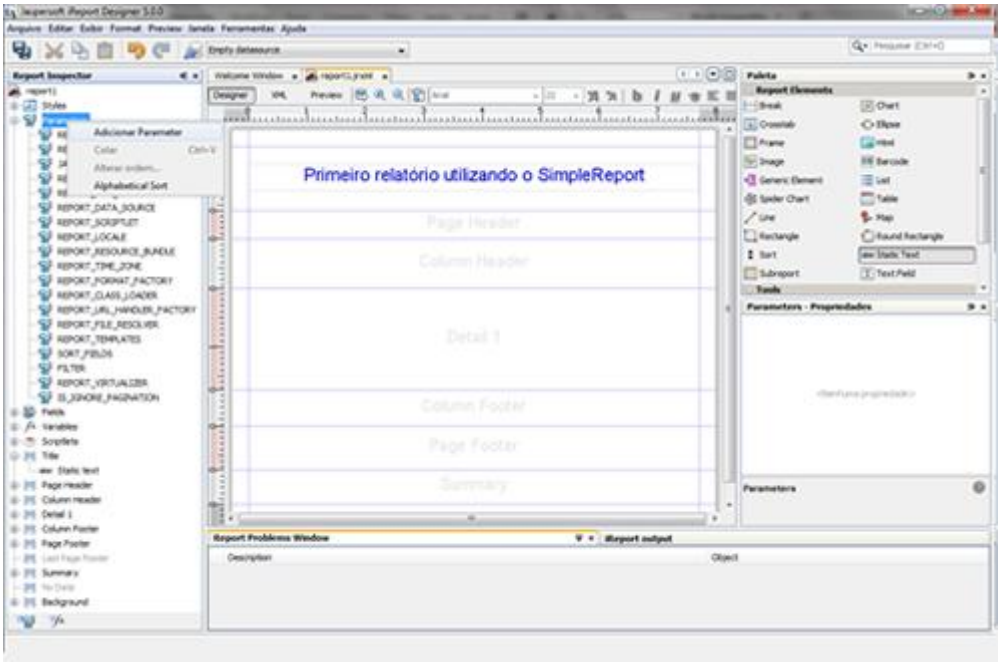

Figura 11 - Criando parâmetros no iReport

Então crie um novo parâmetro chamado de DATA, por exemplo, e arraste ele para o lugar desejado no seu relatório. Veja exemplo na Figura 12.

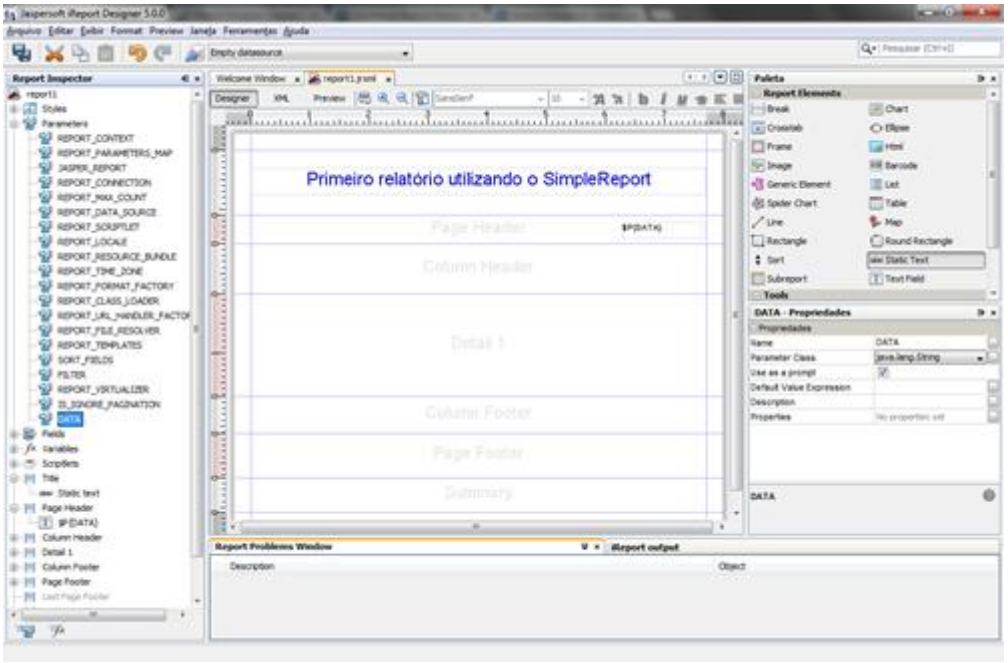

Figura 12 - Relatório no iReport com parâmetro

Feito isso, basta adicionar uma nova linha no seu arquivo php, para substituir o parâmetro criado dentro do iReport pelo valor desejado, conforme a Listagem 14.

```
require_once 'simpleReport/Report.php';
require_once 'simpleReport/core/SRParameter.php';
SRParameter::set('DATA', date('d/m/Y H:i:s'));
Report::from('report.jrxml')->outPut();
```

Listagem 14 Exemplo SimpleReport com parâmetros

Agora vamos desenvolver um relatório que utilize o banco de dados, e para isso, precisamos utilizar a banda *detail*. Ela irá se repetir até não existir mais dados para serem exibidos.

Vale salientar que, dentro do iReport temos a possibilidade de inserir a *SQL (*Linguagem de Consulta Estruturada*)*, mas isso ainda não esta funcionando no SimpleReport. Então precisamos informar a *query* no código mesmo. Veremos isso daqui a pouco. Agora precisamos criar os *fields* (campos) da nossa *query*. A criação de um *field* é semelhante a criação de um *parameter*. Para esse exemplo vamos criar três *fields*, são eles: codigo, nome e endereco. Depois de cria-los, acomode-os dentro da banda *detail*.

Para colocarmos nomes nas colunas da listagem do relatório, utilizaremos a banda *Column Header*. Perceba que ao colocarmos os *fields* na banda *detail* o iReport já cria automaticamente o nome da coluna referente ao *field* selecionado com um *static text*. Diminua um pouco a banda *detail*, pois a altura dela vai ser o espaço que será replicado a cada registro de sua SQL. Depois de adicionar esses três *fields*, terá um relatório parecido com a Figura 13.

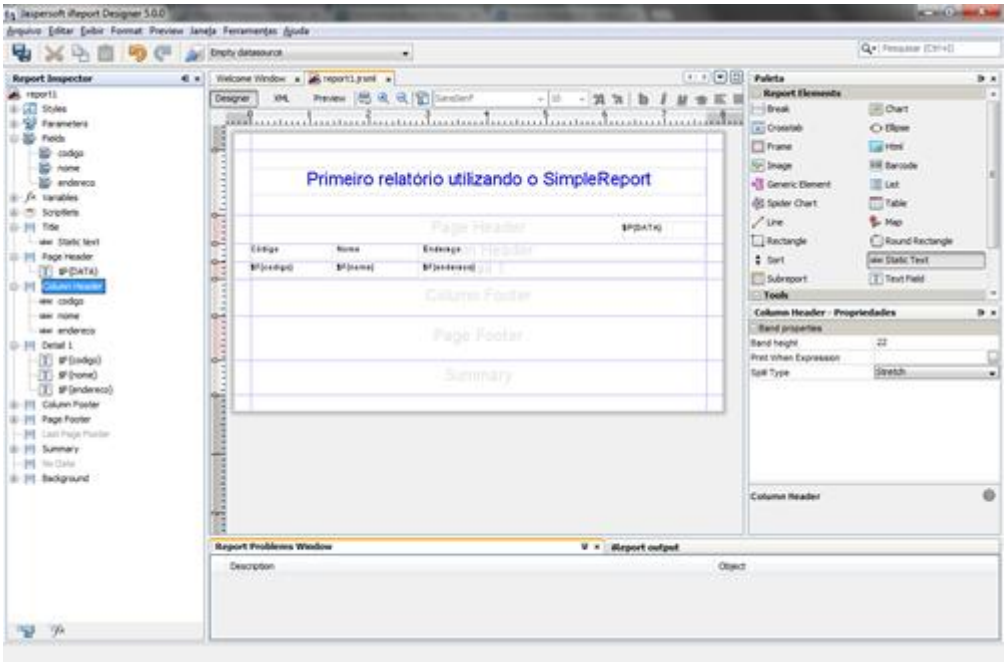

Figura 13 - Relatório no iReport com fields

O relatório já esta projetado para trabalhar com banco de dados. Agora precisamos informar para o SimpleReport qual SQL ele deverá utilizar. Para isso, precisamos utilizar a classe `SRDataSource`, que possui um método estático que recebe dois parâmetros. O primeiro é qual fonte de dados será utilizada e o segundo é uma *queryResult.* Veja o exemplo na Listagem 15.

```
require_once 'simpleReport/Report.php';
require_once 'simpleReport/core/SRParameter.php';
require_once 'simpleReport/core/SRDataSource.php';

$link_identifier = mysql_connect('localhost', 'root', 'root');
$db_selected = mysql_select_db('simpleReport', $link_identifier);
$result = mysql_query('select * from pessoas', $link_identifier);

SRParameter::set('DATA', date('d/m/Y H:i:s'));
Report::from('report.jrxml',SRDataSource::getInstance('MySQL', $result))->outPut();
```

Listagem 15 Exemplo SimpleReport utilizando MySQL

Feito isso, será impresso no *browser* o relatório modelado no iReport. Mas se for necessário forçar o download do relatório ao invés de mostra-lo em tela, é necessário utilizar o método `export` no lugar do `outPut`.

ANEXO A: GNU LESSER GENERAL PUBLIC LICENSE

Version 3, 29 June 2007

Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

Additional Definitions.

As used herein, “this License” refers to version 3 of the GNU Lesser General Public License, and the “GNU GPL” refers to version 3 of the GNU General Public License.

“The Library” refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An “Application” is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A “Combined Work” is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the “Linked Version”.

The “Minimal Corresponding Source” for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The “Corresponding Application Code” for a Combined Work means the object code and/or source code for the Application, including any data and utility programs

needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or

b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the object code with a copy of the GNU GPL and this license document.

Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.

b) Accompany the Combined Work with a copy of the GNU GPL and this license document.

c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.

b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License “or any later version” applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.