**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 14 de fevereiro de 2013, em Rio Grande, estado do Rio Grande do Sul, pela banca examinadora constituída pelos professores:

---

Prof. Tgo. Márcio Ramos Torres
Orientador

---

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

---

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

# AGREDECIMENTOS

Primeiramente gostaria de agradecer a Deus por ter conseguido chegar até aqui, depois a minha família por ter me dado todo o suporte necessário para eu poder estudar sem mais preocupações, e a minha esposa Marcia, por estar sempre do meu lado me apoiando e incentivando.

Também gostaria de agradecer meu Orientador Marcio Torres por ter se dedicado muito durante todo o projeto, e ao meu amigo Bernardo Silva que me ajudou e auxilio no desenvolvimento 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 nos possibilite desenvolver relatórios no formato PDF utilizando uma ferramenta visual e a dificuldade encontrada para suprir essa necessidade, é que desenvolvemos o SimpleReport, um framework Open Source totalmente escrito em PHP.

Com ele, podemos modelar o relatório utilizando o iReport e depois converte-lo para o formato PDF, com poucas linhas de código e tudo escrito em PHP. Temos também a flexibilidade de estendê-lo para implementar novos elementos do iReport e novas fontes de dados, conforme sua 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 design the report using iReport and then converts it to PDF for, 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 CÓDIGOS

# 1 INTRODUÇÃO

É tradicional nos *software*, à 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[1] e a DANFE[2]. E 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, como o HTML, por exemplo, poderá ser impresso um pouco diferente, pois depende de como o *browser* irá renderizar[3] o relatório.

Só que para gerar documentos no formato PDF em aplicações escritas em PHP existe uma grande 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, só que nesse modelo precisamos de uma equipe que domine no mínimo duas linguagens, isso 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*[4].

Com ele você pode usar o modelador de *layout* iReport e transformar esse "modelo" em documento PDF, tudo isso com poucas linhas de código e somente utilizando o PHP. Desse modo você terá a possibilidade de criar relatórios profissionais utilizando uma ferramenta consolidada no mercado e sem precisar treinar sua equipe para conhecer uma nova linguagem.

---

[1] Documento Auxiliar do Conhecimento de Transporte Eletrônico

[2] Documento Auxiliar da Nota Fiscal Eletrônica

[3] É processo pelo qual pode-se obter o produto final de um processamento digital qualquer.

[4] É um conjunto de classes implementadas em uma linguagem específica, usadas para auxiliar o desenvolvimento de software.

Veremos que 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.

## 1.1 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. A ideia é utilizar o modelador de *layout* iReport, para criar o relatório, e não depender do JasperReport para compilar e interpretar ele. Até mesmo porque essa solução necessita de um servidor Java.

## 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. E 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. (WYSIWYG[5])

Só que existe uma dificuldade para desenvolver o *layout* desses relatórios, pois não há uma forma fácil e simples para desenvolvê-los em que possa ser utilizado em aplicações PHP. O SimpleReport preenche essa lacuna, gerando o relatório no formato PDF utilizando o *layout* projetado no iReport.

[5] É o acrônimo da expressão em inglês "What You See Is What You Get"

## 1.3 ORGANIZAÇÃO DO TRABALHO

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

No primeiro capítulo trata-se da Introdução, objetivo e justificativa.

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.

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, mostraremos as vantagens e 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*. Vamos mostrar também 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*, 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, que é quando os sites começam a ficar mais “inteligentes” e configuráveis para o usuário final, toda essa mudança pode ter passado despercebido pelos usuários comuns, mas por trás desses sites existem sistemas complexos com regras de negócios, banco de dados, *logs*, preocupações com segurança e etc.

Para citar alguns exemplos que podem parecer sites simples, mas necessitam de um sistema por trás:

- Google
- Gmail
- Wikipédia
- Blogs em geral

A ideia realmente é tornar esses sistemas complexos muito fáceis de serem usados.

Existem muitos estudos em cima desse assunto, isso é denominado de usabilidade[6].

Juntamente com essas aplicações *web* que mais parecem sites vem crescendo também os *softwares* na *web*, essas aplicações são mais voltadas para o 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* é feito em algum computador na nuvem.

**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 esta em um único computador e só ele precisa ser atualizado.

**Compatibilidade entre sistemas operacionais:** Os sistemas *desktop* normalmente eram 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*.

Mas aplicações *web* também possuem algumas desvantagens em relação a aplicações *desktop*, como por exemplo, o uso 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 um modo fácil e agradável de visualizar grande quantidade de dados.

[6] Facilidade com a qual um equipamento ou programa pode ser usado.

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 vai ter de verba disponível para o orçamento do próximo ano.

E tratando de aplicações *web*, todos os textos e tabelas que são mostrados na tela do computador, é HTML. Pensando assim que durante muito tempo, os relatórios de sistemas *web* eram feitos em HTML, só que no HTML temos certo "problema", é bem complicado desenvolvermos um relatório que seja impresso sempre igual, pois cada navegador possui suas regras para renderizar os objetos e cada um interpreta o código de um jeito, então a solução para criarmos um relatório que tenhamos a garantia de que ele será impresso igual em qualquer computador, é utilizando o formato PDF (Portable Document Format), que serve para representar documentos de maneira independente do aplicativo, hardware e do sistema operacional. 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 **W**orld 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, obtém e processa o resultado. O resultado que será mostrado na tela do computador é sempre HTML.

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 **H**TML

Abreviação para *HyperText Markup Language*, que significa *Linguagem de Marcação de Hipertexto*, é uma linguagem de marcação, e não de programação, e é utilizada para formatar páginas da *web*. Com ele temos a facilidade de criação das páginas, e com a utilização de *browser* conseguimos conversar digitalmente com todo mundo. [CARRIL]

A seguir veremos 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>
```

**Código 1 - Exemplo 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 **H**TTP

É a sigla de *HyperText Transfer Protocol* que em português significa "Protocolo de Transferência de Hipertexto". É um protocolo de comunicação entre sistemas de informação que permite a transferência de dados entre redes de computadores.

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 retornara um código de *status* HTTP em resposta a 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.

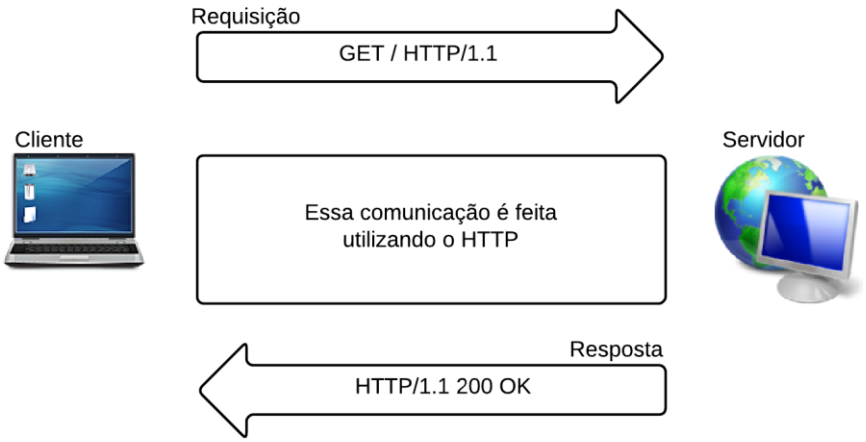


**Figura 1 - Protocolo HTTP**

### 2.2.4 **W**eb 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.

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 paginas estáticas e começaram a trocar conteúdos. Dentro desse contexto começaram a nascer as redes sócias, 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]

### 2.2.5 **C**loud 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]

### 2.2.6 **P**ortable Document Format

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

Uma das principais vantagens dos arquivos PDF é que eles são multi-plataformas, ou seja, podem ser visualizados e impressos em praticamente todas as plataformas entre elas Windows, Mac OS e até mesmo em plataformas móveis e ficará idêntico aos arquivos originais, caso que não ocorre com outros formatos, como por exemplo, arquivos HTML. Por isso a importância de desenvolver relatórios utilizando esse formato em especifico.

> *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]

### 2.2.7 **P**HP

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

Com o avanço da linguagem, ele 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 **X**ML

É um formato de arquivo para a criação de documentos com dados organizados de forma hierárquica. É também um arquivo 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.

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

**Código 2 - Representação de dados usando XML**

Como pode ser observado no código XML apresentado no Código 1, trata-se de um texto simples que pode ser criado em qualquer editor de texto, como por exemplo, o bloco de notas.

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]

### 2.2.9 **R**elatórios

É um conjunto de informações utilizadas para reportar resultados parciais ou totais de uma determinada atividade, experimento, projeto, ação, pesquisa, ou outro evento que esteja finalizado ou em andamento. [Relatório]

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 e serem feitas por um funcionário da empresa, por exemplo, já os gerencias, 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 sim 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 de a possibilidade de modelar *layouts* de forma visual (arrastando objetos) para gerar esse relatórios, o posicionamento dos objetos dentro do relatório tem que ser definido um por um, gerando uma grande perca de tempo. Vejamos exemplo de código PHP utilizando a biblioteca FPDF para gerar um relatório simples:

```
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");
```

**Código 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, só que em relatórios grandes, esse código fica muito complexo e de difícil manutenção. Então para resolver esse problema, a proposta é utilizar uma ferramenta que nos da à possibilidade de 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 a IDE que irá modelar os *layouts* dos relatórios, cuja seu nome é, 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]

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

Possui uma sintaxe semelhante a C/C++, possui também um sistema inteligente de desalocação de memória automático muito conhecido como coletor de lixo.

O Java deve toda sua portabilidade graças a sua maquina virtual que compila seus códigos para diversas plataformas.

O compilador do Java transforma o código fonte em código de maquina (bytecode), esse bytecode é lido e executado pelo JRE. Vale salientar que esse

processo de compilação e execução pode ser bem demorado em relação a outras linguagens como por exemplo o C.

Para darmos inicio ao desenvolvimento Java precisamos do JDK (Java Development Kit) que é um conjunto de classes do Java + as classes do JRE.

Mas se precisarmos apenas executar programas em Java como sites e jogos precisamos apenas do JRE.

## 3.2 IREPORT

É um programa Open Source, capaz de modelar visualmente os mais complexos relatórios para aplicações Java no formato da biblioteca JasperReports que é 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’ esse que é utilizado pelo JasperReport.

Já na realidade do SimpleReport, esse arquivo compilado vai ser descartado pois ele será responsável por interpretar o arquivo XML gerado pela ferramenta visual e montar o relatório em PDF.

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

### 3.2.1 **C**omo funciona

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

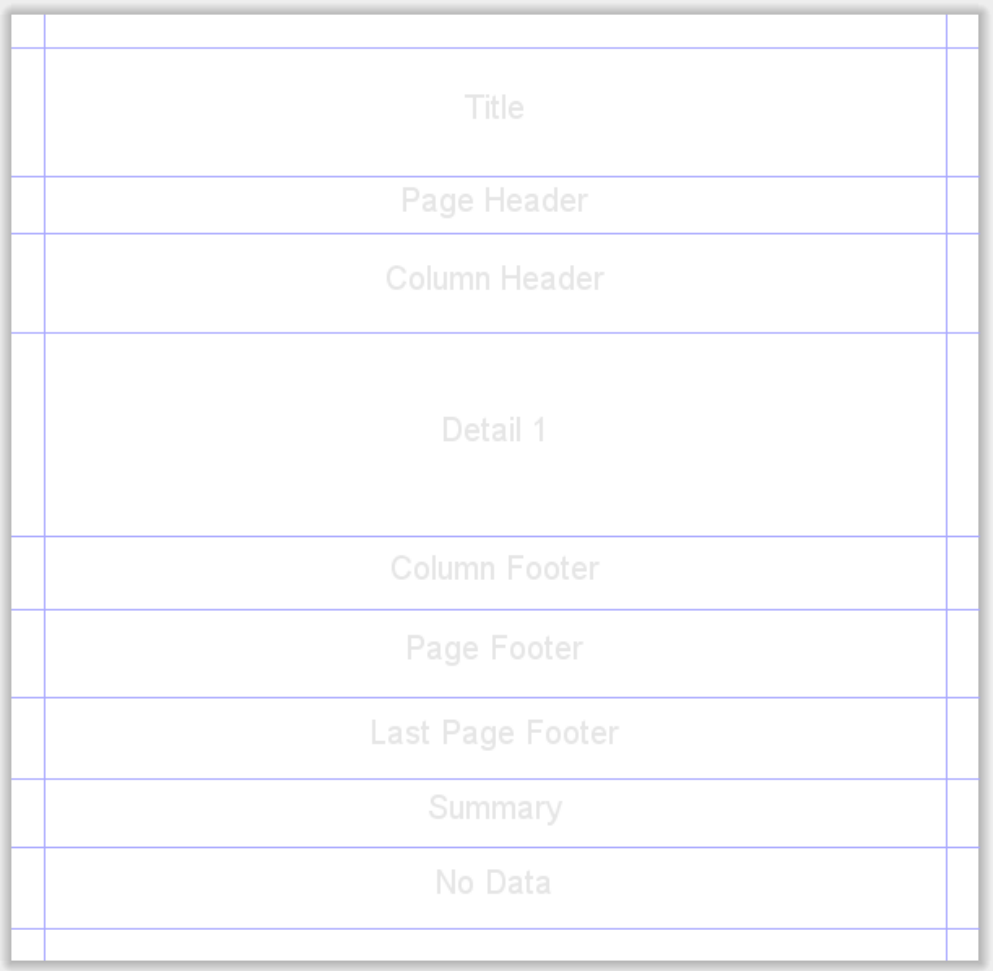


**Figura 2 - Seções disponíveis no iReport**

Cada banda possui um comportamento especifico, 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 a APÊNDICE A.

O relatório modelado no iReport além das bandas, é composto também de elementos, que são objetos que serão posicionado dentro do seu relatório, veja na Figura 3, quais são os elementos disponíveis para ser utilizado nos relatórios.

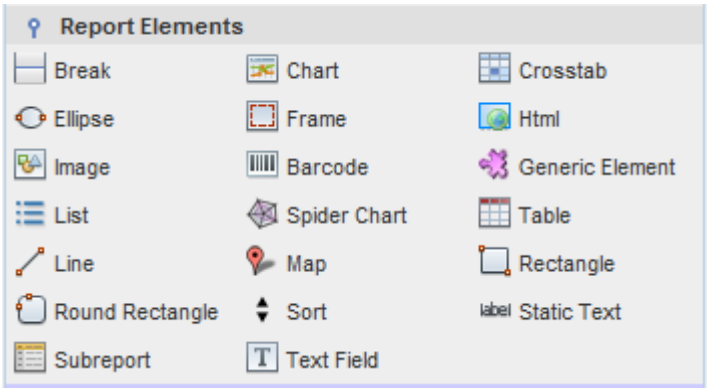


Figura 3 - Elementos disponíveis no iReport

Para ver mais detalhes sobre cada um dos elementos veja o APÊNDICE B.

## 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 **J**asperReport

É uma biblioteca escrita em Java, de código fonte 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]

Esse projeto nasce com o objetivo de otimizar o processo de criação de relatório gerando relatórios com base em arquivos XML bem estruturado, sendo no inicio escrito manualmente pelos programadores.

Com a necessidade de relatórios cada vez mais complexos começou a ficar inviável a construção desses arquivos manualmente, pois tinha a preocupação de posicionar cada elemento um a um e de informar cada propriedade do elemento e outras preocupações, foi então que em 09 de outubro de 2002 o italiano Giulio Toffoli desenvolveu um programa para gerar esses arquivos XML de forma visual chamado de iReport.

### 3.3.2 **U**tilizar um servidor Java

Solução utilizada para usar o JasperReport em aplicações PHP. Veja na Figura 4, como é feita a implementação dessa solução.

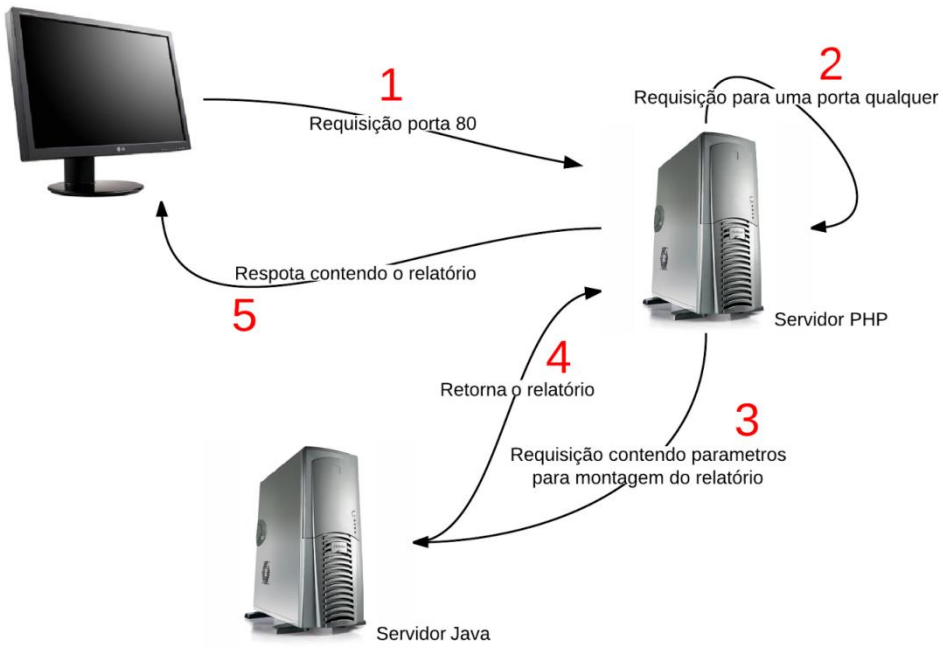


**Figura 4 - Como utilizar o JasperReport em aplicações PHP**

O Principal problema dessa solução e o programador ficar limitado ao protocolo HTTP para fazer a comunicação entre as linguagens. E obviamente para quem tem um sistema em PHP ter que manter um servidor Java simplesmente para responder requisições para a montagem do relatório não é muito agradável.

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, por exemplo, 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 **P**HP 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]

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";
?>
```

**Código 4 - Arquivo de configuração do PHPJasperXML**

Depois basta instanciar a classe `PHPJasperXML` e definir algumas propriedades, veja o exemplo a seguir:

```
<?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");
?>
```

**Código 5 - Exemplo do PHPJasperXML**

# 4 PROJETO E IMPLEMENTAÇÃO

Nesse capítulo iremos falar sobre o processo de descoberta, decisões de projeto, analise 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 todo desenvolvido utilizando como plataforma de desenvolvimento o Eclipse índigo, foi usado também o Google code para armazenar, centralizar e disponibilizar o projeto, no desenvolvimento também foi utilizado como servidor *web* o Apache 2.2, PHP 5.4.3, MyQL 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 e padrões.

### 4.1.1 **D**esign patterns

No 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]

Fazer um sistema orientado a objetos pode ser um pouco complexo, mas fazer um projeto utilizando padrões de projeto com o código totalmente reutilizável é ainda mais complexo, por isso depende de uma equipe mais treinada e experiente, mas concerteza garante que o projeto estará muito bem estrutura e consolidado para mudanças no futuro, isso tudo claro se você utilizar os padrões certos para os problemas certos.

### 4.1.2 **C**onvention over configuration (coc)

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. Uma das vantagens do framework Rails é utilizar esse modelo.

### 4.1.3 **I**terator

Esse padrão de projeto prove uma forma de acessarmos sequencialmente os elementos de uma coleção de dados, sem expor sua representação interna.

### 4.1.4 **M**ySQL

É um SGBD [7], 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]

### 4.1.5 **F**PDF

É uma classe escrita em PHP que permite gerar arquivos PDF. Essa classe possui dezenas de funções para facilitar e agilizar no desenvolvimento do seu arquivo PDF e possui algumas vantagens tais como:

- Escolha da medida de formato de página da unidade, e as margens;
- Cabeçalho e rodapé gestão;

---

**1.1.1** [7] Sistema de Gerencimanto de Banco de Dados

- Quebra de página automática;
- Quebra de linha automática e justificação do texto;
- Suporte de imagem (JPEG, PNG e GIF);
- Cores;
- Links;
- TrueType, Type1 e suporte à codificação;

[FPDF]

### 4.1.6 **F**ail-Fast

Sistemas com essa propriedade tende a comunicar a interface, assim que encontra um comportamento estranho que tende a gerar um erro no código.

### 4.1.7 **L**GPL

Significa GNU Lesser General Public License. Essa licença é um conjunto de permissões adicionais adicionados a versão 3 da GNU General Public License (GPL).

Em termos gerais, a GPL baseia-se em 4 liberdades:

1. A liberdade de executar o programa, para qualquer propósito (liberdade nº 0).
2. A liberdade de estudar como o programa funciona e adaptá-lo para as suas necessidades (liberdade nº 1). O acesso ao código-fonte é um pré-requisito para esta liberdade.
3. A liberdade de redistribuir cópias de modo que você possa ajudar ao seu próximo (liberdade nº 2).
4. A liberdade de aperfeiçoar o programa, e liberar os seus aperfeiçoamentos, de modo que toda a comunidade se beneficie deles (liberdade nº 3). O acesso ao código-fonte é um pré-requisito para esta liberdade.

Com a garantia destas liberdades, a GPL permite que os programas sejam distribuídos e reaproveitados, mantendo, porém, os direitos do autor por forma a não

permitir que essa informação seja usada de uma maneira que limite as liberdades originais.

Os principais motivos de termos utilizados a LGPL ao invés da GPL é que a LGPL visa à regulamentação do uso de bibliotecas de código e permite que sua biblioteca seja incorporada em programas proprietários.
[GNU]

## 4.2 PROJETO

O SimpleReport foi todo projeto para dar maior flexibilidade á quem for utiliza-lo, ele suporta 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.

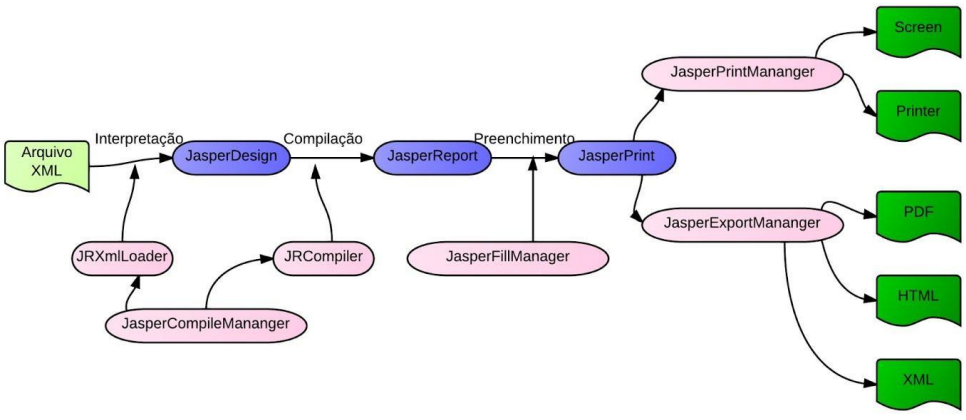


Figura 5- Estrutura do SimpleReport

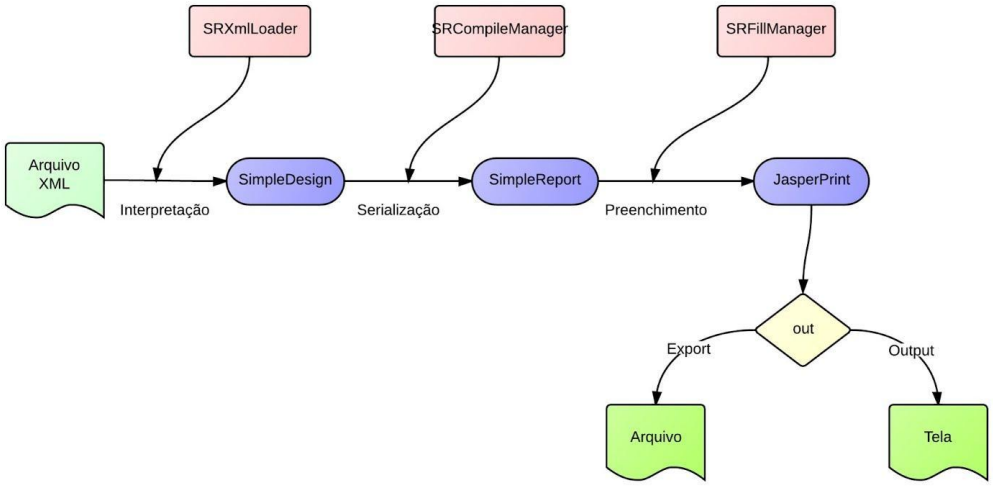


Figura 6 - Estrutura do JasperReport

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

**SimpleDesing**

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 esta estruturado os diretórios do projeto.

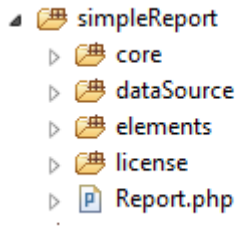


**Figura 7 - Estrutura dos diretórios do SimpleReport**

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

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 padrões como o CoC *(Convention over Configuration)* para facilitar e simplificar o uso da biblioteca, outro padrão utilizado foi o *Iterator,* foi preciso utiliza-lo 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 da total liberdade a que for usa-la, pode ser utilizada até mesmo em softwares proprietários.

## 4.3 IMPLEMENTAÇÃO

Para implementar o SimpleReport foi utilizado 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();
```

**Código 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 da ao usuário a possibilidade de estender o SimpleReport e criar suas próprias fontes 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 seguinte:

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

**Código 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);
	}
}
```

**Código 8 - Classe SRDataSource**

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

```
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();
```

Um princípio utilizado foi o *Fail-Fast*, que qualquer erro encontrado, tanto no processo de interpretação, serialização ou preenchimento do relatório, o código falha, essa foi uma decisão tomada por nós, pois acreditamos que seja a melhor prática para evitar erros no futuro.

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.

```
$fileName = substr('nomeDoRelatorio.jrxml', 0, -6);
// verifica se já existe um arquivo SR
if (file_exists($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);
}
```

Por padrão o SimpleReport já verifica se existe um arquivo de cache para utiliza-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 core 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 por preencher-se, ou seja, somente o elemento sabe se preencher. E o método `draw` é responsável por desenhar-se, ou seja, somente o elemento sabe se desenhar. Desse modo cada elemento é responsável por se preencher e se desenhar, veja a seguir um exemplo do Elemento “Image” implementado.

```
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);
	}
}
```

**Código 9 - Classe Image**

Depois de criada a classe, deve-se colocar dentro do diretório “/elements”.

E 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);
	}
}
```

**Código 10 - Classe MySQL**

Depois de criar a classe, deve-se colocar 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.

- Colocar um elemento dentro do outro;
- 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 suportar ele 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`, ela não nós permite utilizarmos outra fonte de dados que não seja um SGBD. Esse problema ainda tem que pensar uma solução.

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

## 4.6 COMO USAR O SIMPLEREPORT

Primeiro é necessário obter o SimpleReport em http://code.google.com/p/simplereport .

Será necessário também o iReport, para fazer o download acesso o seguinte link http://sourceforge.net/projects/ireport/files/iReport/iReport-5.0.0/, utilizaremos 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 9.

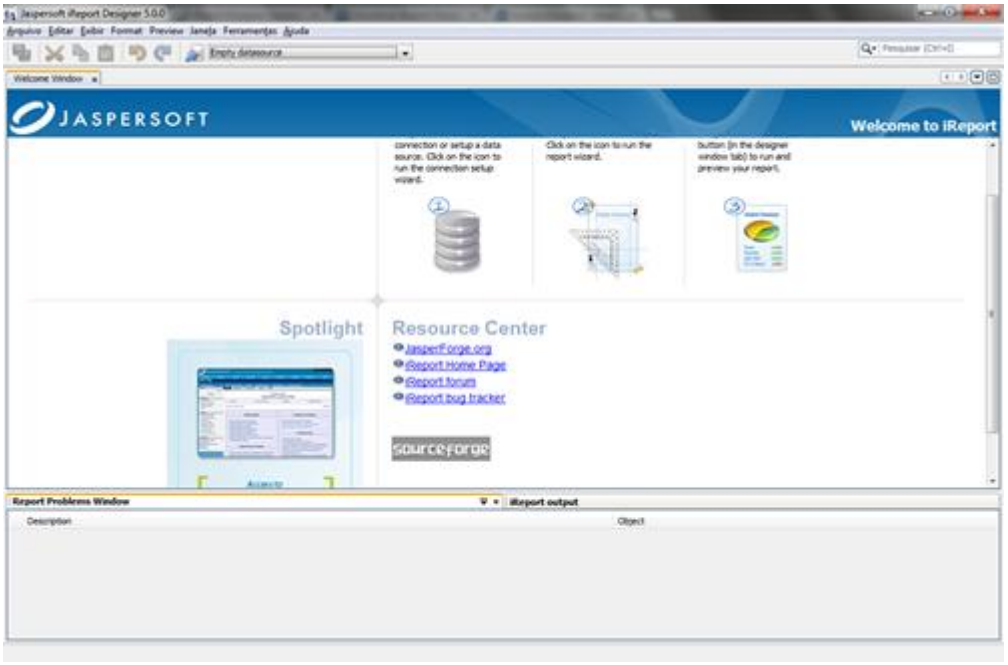


Figura 8 - Tela inicial do iReport

Para criar um novo projeto deve-se acessar o menu Arquivo>New, irá abrir uma janela com varias opções, selecione o arquivo **Blank A4** para começarmos um arquivo novo, depois clique no botão ‘Open this template’. Depois disso irá abrir uma nova janela para informar o nome do relatório e o local dele, clique em próximo e depois Finalizar.

Desse modo o projeto já esta criado, agora deverá aparecer uma tela igual à Figura 10.

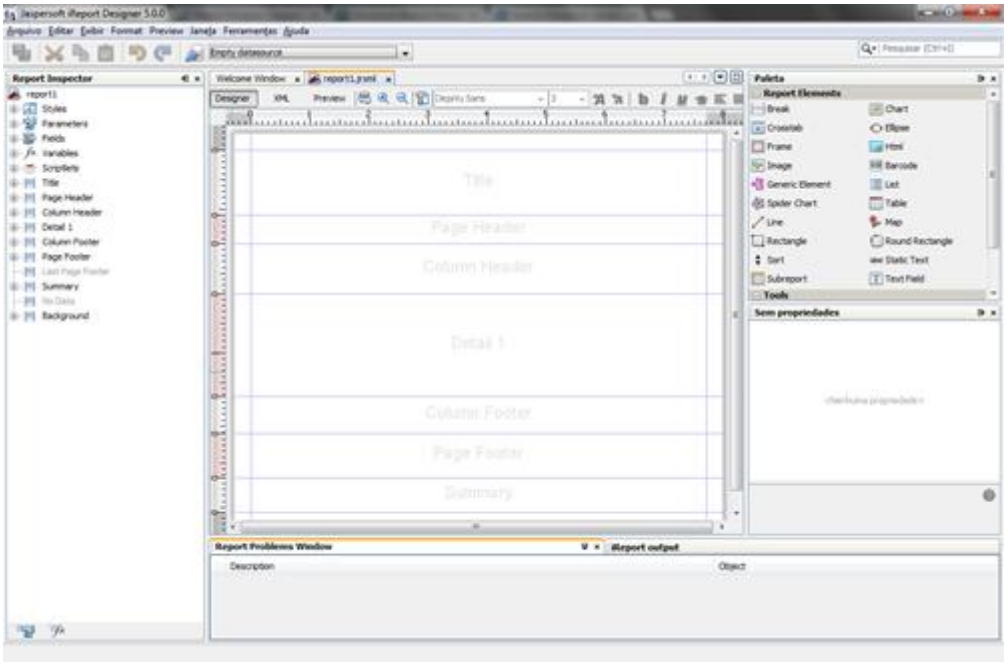

Figura 9 - iReport com novo relatório vazio

Agora já pode começar a desenhar o relatório, No menu ‘Report Inspector’ a esquerda temos os *parameters*, *fields* e as *bands*, se clicarmos em algum item, irá aparecer as 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. Mas 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 algo parecido com isso.

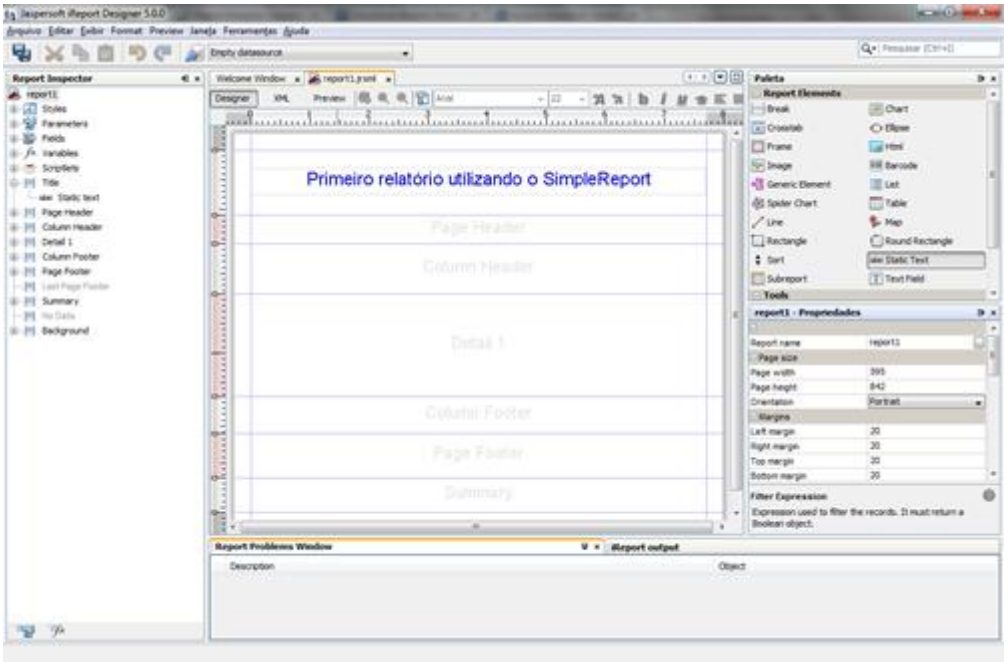


Figura 10 - Exemplo no iReport com elemento static text

Perceba que foi alterado o tamanho e a cor da fonte, e o texto esta centralizado, dentro da banda *Title*.

Vamos supor que o relatório já esta todo desenhado com todos os textos que serão necessários, e agora vamos utilizar o SimpleReport para transforma-lo em PDF.

Depois de fazer o download do SimpleReport e colocar dentro do projeto, então crie um arquivo PHP e faça o seguinte código.

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

**Código 11 - Transformado o *layout* do iReport em PDF**

O *report.jrxml* é o arquivo gerado pelo iReport, feito isso, será gerado o relatório PDF utilizando o SimpleReport, mas esse exemplo foi muito simples, vamos fazer 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 ou POST ou até mesmo um valor vindo do bando 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,* aparecerá uma tela assim:

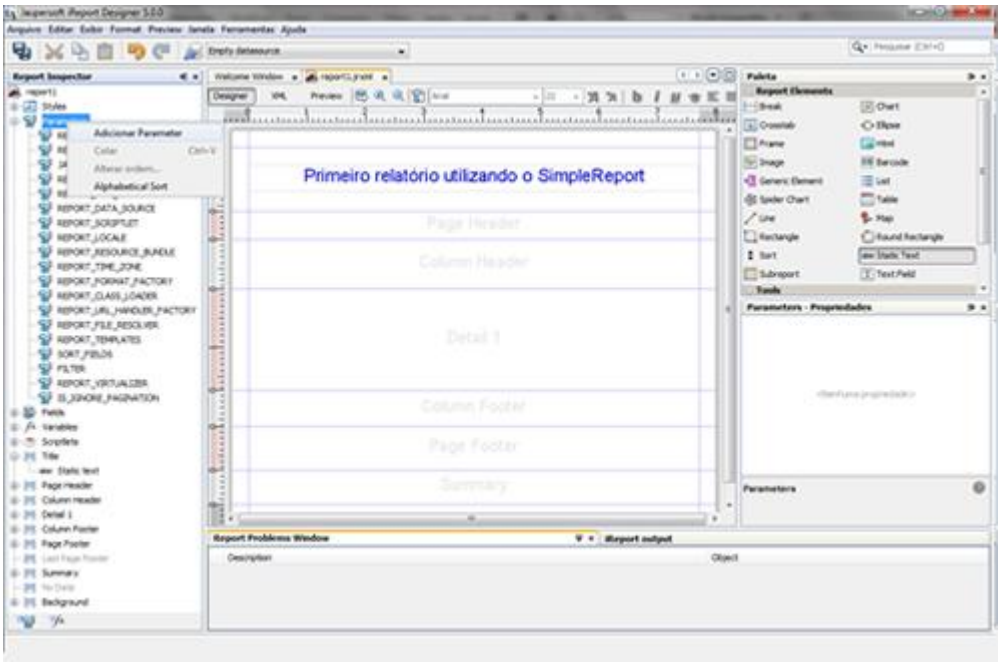


Figura 11 - Criando parâmetros no iReport

Então crie um novo parâmetro chamado de DATA, por exemplo, e arraste ele para o lugar desejado no seu relatório, veja exemplo na Figura 13.

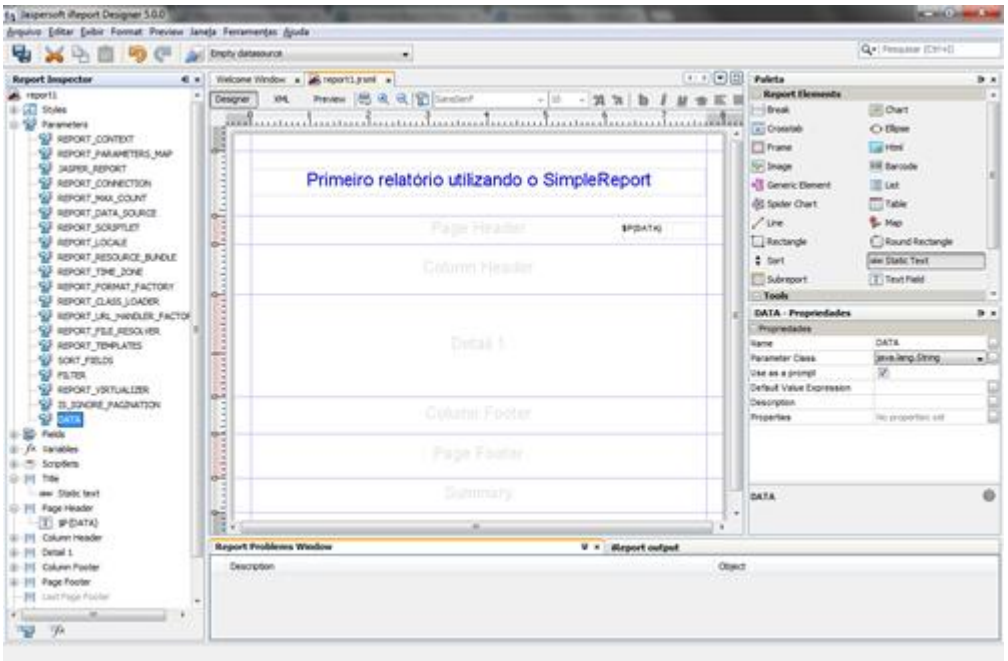


Figura 12 - Relatório no iReport com parâmetro

Feito isso, basta adicionar uma nova linha no seu arquivo php, para substituir o parâmetro criado dentro do iReport pelo valor desejado, o código ficará assim:

```
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();
```

**Código 12 - Exemplo SimpleReport com parâmetros**

Agora vamos desenvolver um relatório um pouco mais interessante, iremos trabalhar com 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 inserirmos a *SQL (*Linguagem de Consulta Estruturada*)*, mais 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 parecido com a criação de um *parameters*, então para esse exemplo vamos criar três *fields*, são eles, o codigo, nome e endereco. Depois de criar eles, acomode eles dentro do seu relatório dentro da banda *detail*.

Para colocarmos nomes nas colunas da listagem do relatório, utilizaremos a banda *Column Header*, perceba que ao colocarmos os *field* 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*, 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 14.

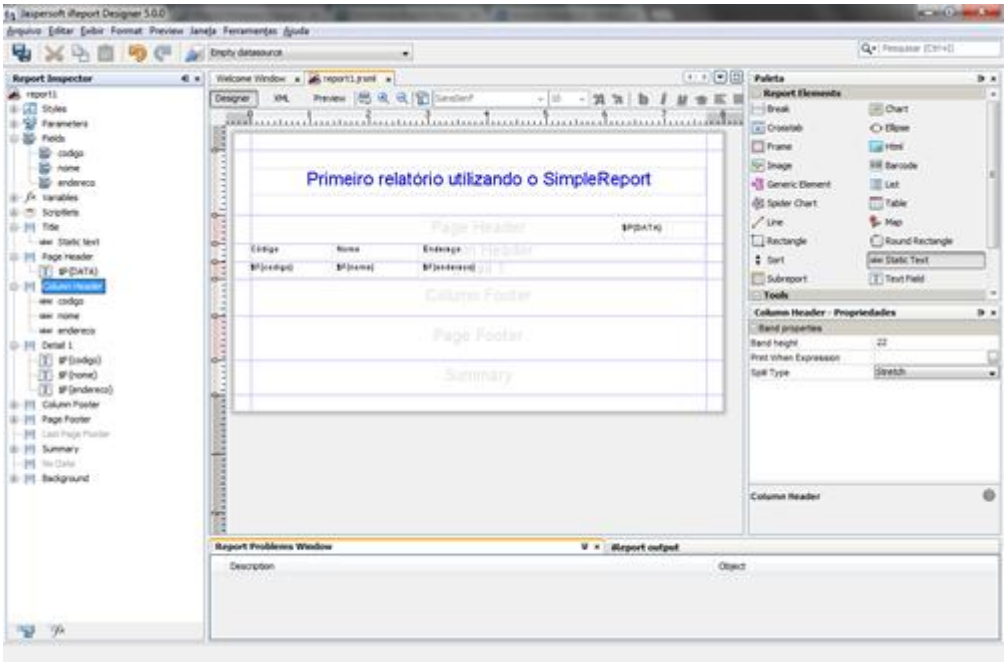


Figura 13 - Relatório no iReport com fields

O relatório já esta desenhando para trabalhar com dados, agora precisamos informar para o SimpleReport qual SQL ele deverá utilizar, para isso, precisamos utilizar a classe `SRDataSource`, ela possui um método estático que recebe dois parâmetros, o primeiro é qual fonte de dados irá utilizar e o segundo é uma *queryResult.* O código ficará assim:

```
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();
```

Código 13 - Exemplo SimpleReport utilizando MySQL

Feito isso, será impresso no *browser* o relatório que tinha sido projeto 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`.

# 5 CONCLUSÃO

Percebemos que o SimpleReport é uma ideia inovadora e funciona muito bem, pois 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 disponibilizar ele e incentivar novas pessoas a contribuir no projeto, através de blogs e grupos na Internet.

Queremos no futuro, desenvolver nosso próprio modelador de telas, e perder o vinculo do iReport, desse modo o projeto irá se tornar independente e poderemos criar nossos próprios elementos e regras.

Se alguém estiver motivado com o projeto, e quiser virar um colaborador ou criar um *fork* [8] do projeto, sinta-se a vontade, basta acessar o site code.google.com/p/simplereport/.

[8] Significa bifurcação, ou seja, pegar o código-fonte e desenvolver a partir dele.

# REFERÊNCIAS


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.

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

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.

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

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

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

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

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

GNU. Disponivel em http://www.gnu.org. Acesso em 20 jan 2013.

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

Relatório. Disponível em http://pt.wikipedia.org/wiki/Relatório. Acesso em 31 de jan 2013.

# APÊNDICE A: Descrição detalhada sobre as seções 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 inicio 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)[9]

**Column Header**
Essa banda só aparece no começo de cada interação com a banda 'detail', Normalmente é utilizada para por 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, e 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 coma 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 ultima página, onde é impresso a banda Last Page

[9] Não serão implementados componentes redimensionáveis

Footer em seu lugar. É muito utilizada para informações como o numero de páginas, uma data ou até mesmo o logo se sua empresa.

**Last Page Footer**
Essa banda substitui o rodapé das páginas regulares na ultima página se seu relatório, se você precisar em um rodapé diferenciado no final de todas as páginas do seu relátorio você vai precisar utilizar essa banda. Se você não definir uma altura para ela, não será interpretada e todos os rodapés de seu relatório será igual os definido na banda Page Footer.

**Summary**
Essa banda é conhecida como o rodapé do relatório, ela será impressa apenas na ultima página logo abaixo da banda Column Footer, é muito comum utilizar essa banda para por algumas informações como um total geral do relatório ou até mesmo um gráfico.

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

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

# APÊNDICE B: Descrição detalhada sobre os elementos 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 relatório gerencias.

**Elipse**
Desenha elipses no relatório

**Frame**
Cria quadros que pode 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 (preenchida 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 ser conectado em determinado momento da exportação do relatório

**List**
Possibilita criar outra query no mesmo relatório, ou seja, uma nova consulta.

**Spider chart**
Gráfico é 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 bando 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.