UMA FERRAMENTA PARA CRIAÇÃO DE GRÁFICOS ESTATÍSTICOS UTILIZANDO FUSIONCHARTS

Robson B. Souza
Instituto Federal de Goiás
robson.souza@academico.ifg.edu.br

Luiz Fernando Batista Loja
Instituto Federal de Goiás
luiz@doutorado.ufu.br

Valdemar Vittorine Graciano Neto
Universidade Federal de Goiás
vgracianoneto@gmail.com

Renato de Sousa Gomide
Universidade Federal de Uberlândia
renato@doutorado.ufu.br

Edna Lucia Flôres
Universidade Federal de Uberlândia
edna@ufu.br

Rodrigo Pinto Lemos
Universidade Federal de Goiás
lemos@emc.ufg.br

***Resumo*—Este artigo descreve um trabalho de pesquisa, cujo o tema está relacionado à uma ferramenta utilizada na renderização de gráficos estatísticos. O *FusionCharts* é uma ferramenta que renderiza gráficos estatísticos, utilizando arquivos *eXtensive Mackup Language* (XML) como fonte dos dados a serem apresentados e as tecnologias *Flash* e *JavaScript* para apresentação visual dos gráficos. Porém, o uso dessa ferramenta em conjunto com a linguagem JAVA é composta por códigos que apresentam várias linguagens distintas, limitando assim a sua manutenibilidade e clareza. Portanto, este artigo apresenta uma solução para o problema utilizando a tecnologia JAVA. É apresentado o resultado da Ferramenta proposta com base em dados quantitativos.**

***Palavras-Chaves* - Ferramentas, FusionCharts, Gráficos Estatisticos, Aplicações Web**

A TOOL FOR STATISTICAL GRAPHICS GENERATION USING FUSIONCHARTS

***Resumo*—This paper describes a research to develop an integration between a tool used in the rendering of statistical graphics. FusionCharts is a tool that renders statistical graphs using files Extensive Mackup Language (XML) like source to be supplied and the Flash and JavaScript technologies for display graphics. However, the use of this tool in conjunction with the Java language consists of codes which have several different languages, thus limiting their clarity and maintainability. Therefore, this article presents a solution to the problem using the JAVA technology. It presented the results of the proposed tool based on quantitative data.**

***Key-Words* - Application Tools, FusionCharts, Statistical Graphics, Web Applications**

XIII CEEL - ISSN 2178-8308
12 a 16 de Outubro de 2015
Universidade Federal de Uberlândia - UFU
Uberlândia - Minas Gerais - Brasil

I. INTRODUÇÃO

Atualmente a representação gráfica é um importante recurso para a análise de dados e tratamento da informação [1]. Essa forma de representação é utilizada em relatórios e apresentações como forma de facilitar o entendimento do usuário da informação [2]. O modo de representação de dados estatísticos de maneira visual como gráfico de barra, pizza, histograma e pareto possibilita um novo ponto de vista na interpretação desse tipo de informação. Esses aspectos fazem com que o trabalho com gráfico se torne mais interessante e de fácil assimilação pelos usuários das demonstrações, haja vista o apelo visual e a maior facilidade de compreensão por parte de seus usuários [2].

Portanto, com o desenvolvimento de softwares capazes de realizar a conversão de informações em gráficos, o esforço para a interpretação dessas massas de dados foi reduzido. Com o aumento da capacidade computacional esse tipo de converção de dados foi incorporada nas aplicações *online*. Sendo assim, a criação de gráficos estatísticos por meio de ferramentas é uma situação presente em diversas aplicações que utilizam a arquitetura cliente/servidor.

O FusionCharts é uma ferramenta de criação e apresentação de gráficos disponibilizada em diversas versões (Flash, JavaScript e HTML5). Essa ferramenta tem a finalidade de auxiliar a apresentação de gráficos. Esse *software* é capaz de apresentar diferentes tipos de gráficos estatísticos, com base em dados codificados e estruturados em *eXtensive Mackup Language* (XML) e *JavaScript Object Notation* (JSON).

Porém, o processo de criação e codificação desses gráficos é demorado e, na maioria das vezes, manual. Esse processo quando automatizado pode gerar uma codificação poluída que mescla diversas linguagens. Por exemplo, como os gráficos devem ser estruturados em XML para serem interpretados pelo FusionCharts o desenvolvedor Java mesclaria a linguagem Java com a linguagem XML no mesmo código. Além disso, quanto mais especifico o tipo de gráfico, maior o nível de conhecimento necessário para codificar os dados utilizando XML.

Portanto, para utilizar o FusionCharts é necessário o conhecimento de diversas linguagens e como implementar a interação entre essas tecnologias. Com o objetivo de diminuir esse esforço de conhecimento e codificação foi desenvolvida uma API (Aplication program Inerface) que permite a criação dos códigos interpretados pelo FusionCharts utilizando apenas a linguagem Java. Essa API possibilita a criação de códigos mais manuteníveis e menos poluidos, além de minimizar o conhecimento necessário para criar os gráficos.

Este trabalho está estruturado da seguinte forma: a Seção II descreve como foi realizada a criação da API proposta neste trabalho. A descrição da ferramenta FusionCharts é apresentado na seção III. Na Seção IV é descrita a estrutura da ferramenta desenvolvida. Uma comparação da geração de código utilizando a API e sem a API é apresentada na Seção V. Finalmente, a Seção VI descreve a conclusão desse trabalho e sugere trabalhos futuros.

II. Metodologia

A primeira etapa desta pesquisa consistiu em elucidar quais as ferramentas utilizadas para geração de gráficos *online*. Foram observadas três ferramentas de plotagem estatística: JSChart [3], RichChartLive [4] e *FusionCharts*. Após definir a ferramenta para qual seria construida a API, foi dado início a um estudo aprofundado sobre os fundamentos utilizados pela abordagem selecionada.

Ao final desse estudo, levantou-se os requisitos necessários para implementar a API. Logo após essa fase criou-se diversos diagramas utilizando Linguagem de Modelagem Unificada (UML) com a finalidade de melhorar entendimento dos requisitos e esquematizar o processo de desenvolvimento. Assim que todos os diagramas foram desenhados, deu-se início a fase de desenvolvimento.

Assim que a implementação foi finalizada varios testes de software foram realizados. Esses testes tinham o intuito de testar a robustes e manutenabilidade da API propsota. Finalmente um estudo de caso com dados quantitativos disponibilizados pelo Instituto Federal de Goiás foi realizado para a validar a solução.

III. FusionCharts XT

FusionCharts é uma ferramenta criada em Flash que pode ser utilizada para criação de gráficos animados. Esse programa pode ser utilizado em conjunto com qualquer linguagem de desenvolvimento para web como C#, Java, Ruby e PHP. Essa ferramenta utiliza o XML ou JSON como interface de dados. Os códigos criados nessas duas linguagens são interpretados pelo software que a partir deles cria gráficos interativos.

Entre as vantagens de utilizar essa ferramenta estão o suporte a diversas linguagens de programação, possibilidade de interação com os gráficos e redução do processamento na parte do servidor. Como a interface de comunicação é feita por meio de XML ou JSON o FusionCharts pode ser utilizado por qualquer linguagem de programação que possa representar os dados por meio desses dois padrões. Além disso, essa ferramenta possibilita a interação com os gráficos gerados permitindo operação de *drill-down* e detalhamento da informação. Finalmente, o custo de processamento do servidor é reduzido pois os gráficos são processados na máquina do cliente.

O FusionCharts apresenta três tipos de gráficos distintos, série simples, série múltipla e grafícos combinados. Os gráficos do tipo série simples representam apenas um valor para cada referência. Esse tipo de gráfico é apresentado na figura 1.

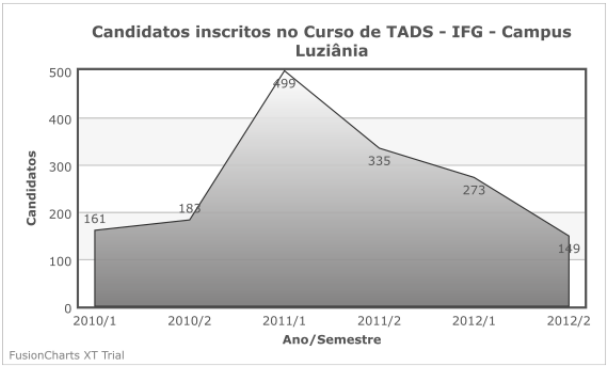

Figura 1. Apresentação de um gráfico do tipo Série Simples.

Os gráficos de multiplas séries aceitam mais de um valor para cada referência sendo agrupados por algum fator comum como característica, período, grupo, entre outros. Esse tipo de gráfico é ilustrado na figura 2.

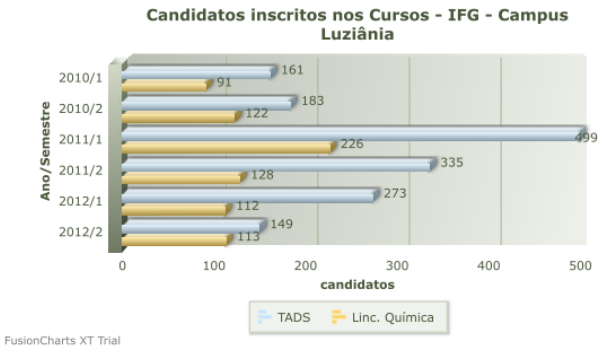

Figura 2. Apresentação de um gráfico do tipo Série Multipla.

Finalmente, o gráfico combinado possui características semelhantes aos gráficos de série múltipla com a diferença de serem capazes de apresentar de maneiras distintas os dados no mesmo gráfico. Esse tipo de gráfico é mostrado na figura 2.

O *FusionCharts* foi escolhido como ferramenta de

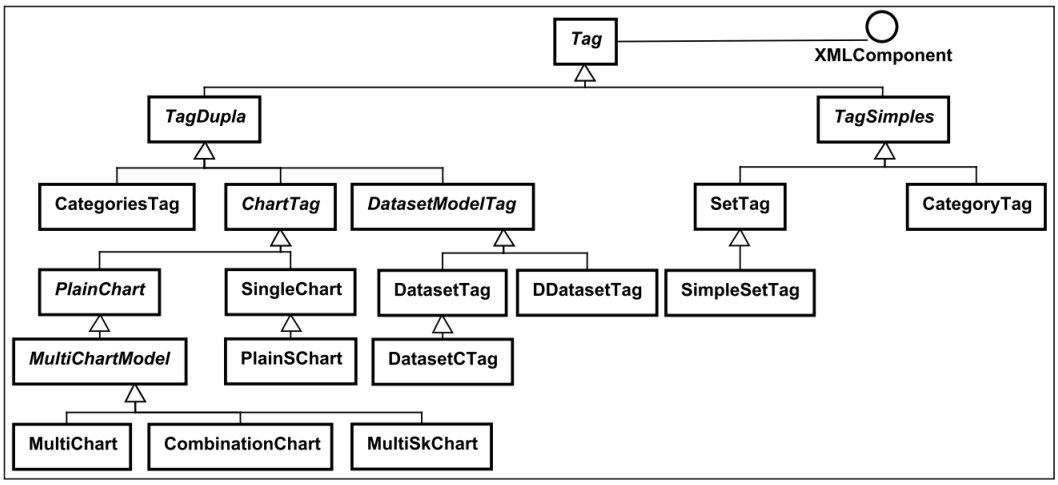

Figura 4. Estrutura base da API.

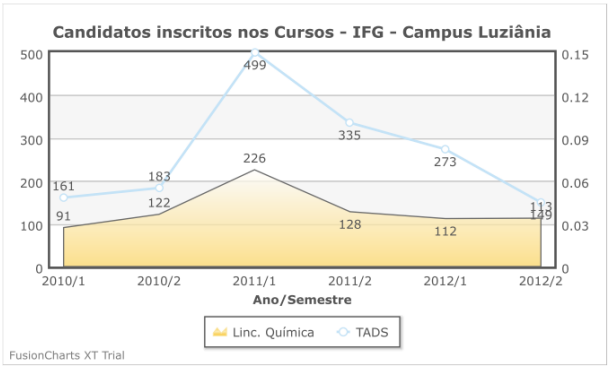

Figura 3. Apresentação de um gráfico do tipo Combinado.

gráficos neste trabalho em detrimento a outras abordagens porque possui uma documentação estruturada e bem definida. Além disso ele suporta uma grande variedade de gráficos e linguagens de programação.

IV. Ferramenta Proposta

A figura 4 apresenta a estrutura geral da API. A base dessa estrutura será explicada com detalhes. Para criar o código da fonte de dados que fomentam os gráficos o *FusionCharts* utiliza uma estrutura XML padrão. Sendo assim, fez-se necessário modelar os elementos que compõem o XML responsável pelos dados dos gráficos. Esse XML é formado basicamente de *tags*. Essas *tags* e sua disposição dentro desse XML constituem o domínio do problema. As *tags* apresentam duas configurações basicas, elas podem ser simples ou duplas.

A tag simples leva este nome porque é composta de apenas uma tag, enquanto que a tag dupla possui uma tag de abertura e outra tag de fechamento. Um exemplo de tag simples pode ser visualizado pela figura 5.

```
< set label='teste' value='23' />
```

Figura 5. Tag simples

A tag é iniciada por um parentese angular de maior "<", após este simbolo é apresentado o nome da tag, neste caso "set". Logo depois do nome da tag são exibidos os atributos "label" e "value" e seus respectivos valores "teste" e "23". Nota-se que existe um sinal de igual entre o atributo e seu valor. A tag é encerrada pelo sinal de "/" seguida do parentese angular de menor ">". No *FusionCharts*, as tags podem conter um número indetermidado de atributos, apesar deste exemplo conter apenas dois.

A tag dupla é similar a tag simples em quase todos os aspectos. A grande diferença entre as duas tags está no fato de que a tag dupla pode conter outras tags em sua composição. Para suportar esta composição a tag dupla não é finalizada após a declaração dos atributos. Portanto, ao finalizar a codificação dos atributos a tag dupla recebe apenas o parentese angular de menor ">". Após este sinal a tag pode conter inúmeras outras tags. Assim que todas as tags desejadas forem incluidas a tag dupla é finalizada por meio da adição de um parentese angular de maior "<" seguido do nome da tag e finalmente o sinal de barra "/" seguido do parentese angular de menor ">". A 6 apresenta uma tag dupla.

```
<chart caption='Média da Classe' subcaption='Notas
    ' >
  < set label='Aprovados' value='10' />
  < set label='Reprovados' value='3' />
  < set label='Recuperação' value='5' />
< chart />
```

Figura 6. Tag dupla

Na figura acima o nome da tag dupla é “chart”, os atributos são “caption” e “subcaption” e seus respectivos valores são “Média da Classe” e “Notas”. A tag chart contém três tags internas do tipo set. Após a declaração das tags internas a tag chart é finalizada.

Para modelar as tags utilizando o Paradigma de Orientação a Objetos foi necessário identificar quais propriedades eram comuns a ambas as tags. Notou-se que toda a tag é iniciada com parentese angular de maior seguido do nome da tag e a declaração dos seus atributos. Logo após a declaração dos atributos há uma distinção na composição das duas tags, como mostra a 7.

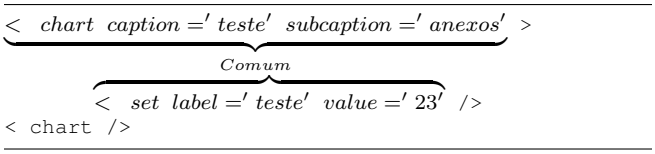

Figura 7. Área comum entre tags

Com a finalidade de agrupar as características comuns entre as duas tags, simples e dupla, foi criada a classe abstrata *Tag*, segundo o padrão de projeto [sec:composite]Composite. Essa classe possui três métodos fundamentais, toXML, getName e formatAttributes. O método toXML é responsável por converter a estrutura de classe em uma estrutura XML equivalente. O código XML gerado por este método é utilizado como fonte de dados para os gráficos do *FusionCharts*.

A classe Tag codifica o método toXML retornando um dado literal contendo apenas os dados primordiais para criação de uma tag. Esse método retorna, por meio desse literal, o parentese angular de início, seguido do nome da tag e seus atributos, ou seja, os dados que são comuns a toda tag. Porém, a classe Tag não especifica qual o nome da tag, tampouco quais atributos são apresentados.

Com o intuito de apresentar o nome da tag dinamicamente, modelou-se Tag como uma classe abstrata e o método getName, responsável por retornar o nome da tag foi codificado como abstrato. Essa abordagem força as classes concretas que herdam de Tag a sobrescreverem o método getName retornando o nome da tag que é apresentado na estrutura do XML. Essa técnica de implementação obedece o padrão de projeto [sec:template$_method]TemplateMethod$.

O método formatAttributes é responsável por formatar os atributos das tags. Os atributos apresentados nas tags são os próprios atributos das classes que herdam de tag. Por exemplo, a classe Set tem um atributo chamado label e outro chamado value, esses campos e seus respectivos valores são listados na tag assim que o método toXML for chamado. Sendo assim, o método formatAttributes pesquisa recursivamente quais os nomes e os valores de cada campo pertecente a classe e os lista no retorno do método toXML. Como a classe Tag não tem conhecimento dos atributos das suas subclasses fez-se necessário a utilização de técnicas de metaprogramação para codificar este método.

Sendo assim, ao executar o método toXML em uma classe concreta as técnicas de reflexão identificam os atributos dessa classe e os retornam formatados para que sejam apresentados corretamente no XML. Essa formatação segue a seguinte estrutura: nome do atributo,sinal de igual, aspas duplas, valor do atributo e fecha aspas duplas.

Vale ressaltar que houve uma preocupação na tipagem de cada atributo. Portanto, se o domínio do atributo da tag fosse inteiro, o atributo declarado no código java era do tipo int. Caso o atributo aceitasse um conjunto de valores restrito o atributo era declarado como um Enum. Esse Enum possuiria apenas os valores validos relacionados ao domínio do atributo. Outro ponto importante foi a validação de atributos nulos, todos atributos destituídos de valores não são listados na tag. A 8 mostra a responsabilidade de cada método na formatação da tag XML.

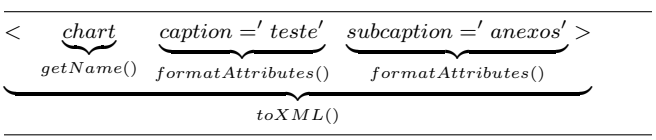

Figura 8. Responsabilidades de cada elemento presente em uma tag, em função dos métodos

Após definir a classe abstrata *Tag* foram implementadas as classes abstratas *TagSimples* e *TagDupla*. Essas duas classes especificam as características particulares de cada tipo de tag, simples e dupla.

A *TagSimples* herda de tag e sobrescreve o método toXML implementado apenas o fechamento da tag adicionado o sinal de barra “/” e o parentese angular de menor “>”. Essa codificação finaliza a implementação da classe TagSimples.

A codificação da classe *TagDupla*, assim como TagSimples, sobrescreve o método toXML. Como especificado anteriormente uma tag dupla pode conter outras tags, tanto simples quanto dupla. Para modelar este comportamento utilizou-se o padrão composite. Sendo assim, adicionou-se uma coleção de objetos denominada *internalsTags*. Essa coleção é reponsável por referênciar as *tags* que são adicionadas *TagDupla*. Portanto, assim que o método toXML é invocado a coleção de tags é percorrida. Para cada tag interna o método toXML é chamado convertendo a Tag em estruturas XML. Após listar as tags o método toXML fecha a tag dupla adicionando parentense angular de maior “<” , o nome da tag, barra “/” e finalmente parentese angular de menor “>”.

Com a finalidade aumentar a extensibilidade da API adicionou-se a interface *XMLComponent*. Essa interface permite que elementos que não sejam de fato tags sejam adicionados a estrutura do XML. Um exemplo deste tipo de elemento é o comentário XML. O comentário não é de fato

uma tag, mas pode ser adicionado a estrutura do XML. A XMLComponent possui o método toXML que é o método fundamental para classes pertencentes a API.

A partir da estrutura principal, definida anteriormente, a API extende-se conforme apresentado na figura {fig:estrutura. A classe *ChartTag* representa a tag "chart", que contém os dados gerais dos gráficos, por esse mótivo possui o maior conjunto de atributos. Os Atributos da tag "chart" vão desde nome do gráfico até sombra do invólucro do gráfico. O método *getName* é sobrescrito para retornar o nome da tag, neste caso "chart". A classe ChartTag origina duas subclasses *SimpleChart* e *PlainChart*.

A classe *SimpleChart* representa a categoria "Simple charts" presente no *FusionCharts*. Sendo assim, essa classe possui uma coleção de classes do tipo *SimpleSetTag*. A *SimpleSetTag* representa a tag "set" responsável por informar os dados do gráfico. A classe *SimpleSetTag* possui basicamente dois atributos *label* e *value*.

A coleção de *SimpleSetTag* é inicializada pelo *SimpleChart* utilizando o padrão de projeto *especialista*. Seguir este padrão de projeto permitiu criar um método que recebe apenas o *label* e o *valor* que serão utilizados na criação da *SimpleSetTag*. Esse tipo de implementação facilita a utilização da API, pois aumenta a performance de desenvolvimento retirando do programador a responsabilidade de instanciar cada *SimpleSetTag*.

O *PlainChart* agrupa os elementos comuns aos gráficos que adotam o uso de uma área de plotagem. Para utilizar a área de plotagem modelou-se um grande conjunto de atributos. Esses atributos permitem a manipulação da área de plogatem do gráfico. A classe *PlainSChart* é uma variação da classe *SimpleChart* com a adição dos atributos especiais. Essa abordagem é utilizada porque categoria *SimpleChart* reune gráficos que não possuem área de plotagem. Alguns desses atributos são: *canvasbgColor* (cor de fundo da área de plotagem), *yAxisName* e *xAxisName* (nomes apresentados nos eixos x e y respectivamente) entre outros.

A classe *PlainChart* origina a subclasse *MultiChartModel* que representa o modelo inicial da categoria "Série Multiplas" do *FusionCharts*. Essa classe foi criada com objetivo de agrupar os elementos comuns entre as categorias Multi Chart e Combination Chart.

Com o objetivo de representar as categorias dos gráficos do tipo "multi-series" adicionou-se a classe *MultiChartModel* uma coleção de *Category*. Essa coleção é representada pela classe *CategoriesTag* que representa a tag dupla "categories". *CategoriesTag* possui uma coleção de *Category* responsável por representar as categorias pertencentes a todos os chart to tipo "multi-series". A partir da classe *MultiChartModel* implementou-se duas subclasses *MultiChart* e *CombinationCharts*.

A classe MultiChart possui uma lista de *Dataset*, sendo que cada dataset tem um lista de *SetTags*. *Dataset* é a classe agrupada pelo ArrayList *datasets* e essa classe é formada por um conjunto de *SetTags*. O set que faz parte de *Dataset* é a **superclasse** de *SimpleSetTag* e *SetTag*. Isso ocorre pois, diferente da classe usada por *SimpleChart*, a tag "set" das categorias Multi-series e Combination series não utilizam o atributo 'label'. A *SetTag* extende de TagSimples e implementa o método getName, de maneira que retorne "set" e como atribudo possui value que é do tipo Number.

V. Comparação entre Códigos com e sem a API

A API proposta neste trabalho apresenta uma abordagem diferenciada de geração dos códigos XML para fomentar os gráficos. Para gerar um gráfico utilizando apenas o FusionCharts o desenvolvedor teria que utilizar a linguagem Java junto com a tecnologia XML no mesmo código. A Figura 9 apresenta uma forma simples de codificar um gráfico do tipo Série Simples utilizando *FusionCharts* e a linguagem de programação Java. Neste exemplo, a classe BasicRenderData possui diversos atributos, entre eles o atributo "xml". Esse atributo é do tipo literal e recebe o código XML que alimenta os dados do gráfico. Para criar esse XML é necessário o conhecimento de três tecnologias, são elas, XML, Java e *FusionCharts*.

```
package com.fusioncharts.sampledata;
public class BasicRenderData {
 protected String xml;
 protected String chartId = "basicChart";
 protected String width = "600";
 protected String height = "300";
 protected String swfFilename = ChartType.COLUMN3D
     .getFileName();
 protected String uniqueId = "";
 public BasicRenderData() {
  xml = "<chart caption='Monthly Unit Sales'
  xAxisName='Month' yAxisName='Units' showValues
      ='0'
  formatNumberScale='0' showBorder='1'>";
  xml += "<set label='Jan' value='462' />";
  xml += "<set label='Feb' value='857' />";
  xml += "<set label='Mar' value='671' />";
  xml += "<set label='Apr' value='494' />";
  xml += "<set label='May' value='761' />";
  xml += "<set label='Jun' value='960' />";
  xml += "<set label='Jul' value='629' />";
  xml += "<set label='Aug' value='622' />";
  xml += "<set label='Sep' value='376' />";
  xml += "<set label='Oct' value='494' />";
  xml += "<set label='Nov' value='761' />";
  xml += "<set label='Dec' value='960' />";
  xml += "</chart>";
 }
}
```

Figura 9. Exemplo básico de uso segundo a documentação do *FusionCharts*(*FusionCharts*,2013).

No código apresentado pela figura V nota-se claramente a mesclagem dos códigos Java junto ao XML. Esse tipo de abordagem pode tornar o código confuso além de dificultar a manutenção. Outro ponto deficiente neste tipo de codificação é o tamanho das Strings de XML. As vezes esses literais

podem ser muito extensos e apresentar alta complexidade dificultando a validação e manutenção do código.

Utilizando a API o trabalho de criação dos gráficos é reduzido como apresentado pela figura 10. Essa figura mostra o mesmo gráfico gerado pela figura9. Nota-se que a API substitui todo o código XML. Isso evita possíveis problemas, como: qualquer erro de sintaxe ou atribuir valores de tipos não suportados por um atributo específico. Além disso, as documentações das classes Java auxiliam o desenvolvedor a criar o gráfico, pois, se o desenvolvedor estiver usando algum tipo de IDE ele poderá consultar a documentação do *FusionCharts* integrada a API enquanto codifica.

```
public class BasicRenderData {
 String xml;
 protected String chartId = "basicChart";
 protected String width = "600";
 protected String height = "300";
 public BasicRenderData() {
  Column3DChart chart = new Column3DChart();
  chart.setCaption("Monthly Unit Sales");
  chart.setXAxisName("Month");
  chart.setYAxisName("Units");
  chart.setShowValues(false);
  chart.setFormatNumberScale(0);
  chart.setShowBorder(true);
  String[] months = { "Jan", "Fev", "Mar", "Abr",
      "May","Jun", "Jul", "Aug", "Sep", "Oct", "
      Nov", "Dez" };
  int[] units = { 462, 857, 671, 494, 761, 960,
      929, 622, 376, 494, 761, 960 };
  chart.addSet(months, units);
  xml = chart.toXML();
 }
}
```

Figura 10. Exemplo básico de uso segundo a API desenvolvida para o uso do *FusionCharts*.

VI. CONSIDERAÇÕES FINAIS

Com a crescente tendência de desenvolvimento de aplicações cliente/servidor o FusionCharts se mostra uma ferramenta promissora na criação de gráficos estatísticos [5]. Porém a codificação necessária a aplicação apresenta elementos que tornam o código confuso e extenso. A API desenvolvida neste trabalho torna o código mais robusto e manutenível, assim como, diminui o conhecimento necessário para geração de gráficos utilizando FusionCharts. Outro ponto importante é que com a utilização da API o desenvolvedor tem mais tempo para aprimorar áreas críticas da aplicação ao invés de tratar questões relacionadas a apresentação dos gráficos.

Como trabalhos futuros pretende-se otimizar as interfaces de geração de gráficos e disponibilizar essa API para comunicadade como um projeto de código fonte aberto.

REFERÊNCIAS

[1] C. R. FLORES and M. T. MORETTI, "O funcionamento cognitivo e semiótico das representações gráficas: ponto de análise para a aprendizagem matemática," *REUNIÃO ANUAL DA ANPED, GT19: EDUCAÇÃO MATEMÁTICA*, vol. 28, pp. 1–13, 2005.

[2] L. C. Miranda, A. da Silva Vieira, U. C. T. Lagioia, and M. T. de Castro Vasconcelos, "Utilização de gráficos em demonstrações contábeis," *Revista de Educação e Pesquisa em Contabilidade (REPeC)*, vol. 2, no. 3, pp. 16–35, 2009.

[3] Jscharts.com, "Js charts - free javascript charts," 2015. [Online]. Available: http://www.jscharts.com/

[4] Richchartlive.com, "Rich chart live - create enjoyable and captivating flash charts from your web browser for free," 2015. [Online]. Available: http://www.richchartlive.com/RichChartLive/

[5] Z. Ting, "Application of fusioncharts in web system," *Proceedings of the 2nd International Symposium on Computer, Communication, Control and Automation*, 2013.