UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

DIGRENDER – RENDERIZADOR DE INTERFACE GRÁFICA PARA WEB EM PHP COM ORIENTAÇÃO A OBJETOS

TRABALHO DE GRADUAÇÃO

Carlos Edmilson da Silva Maia

Santa Maria, RS, Brasil
2007

DIGRENDER – RENDERIZADOR DE INTERFACE GRÁFICA PARA WEB EM PHP COM ORIENTAÇÃO A OBJETOS

por

Carlos Edmilson da Silva Maia

Trabalho de Graduação apresentado ao curso de Ciência da Computação – Bacharelado, da Universidade Federal de Santa Maria (UFSM, RS) como requisito parcial para obtenção do grau de
Bacharel em Ciência da Computação

Orientadora: Prof. Dra. Roseclea Duarte Medina

Trabalho de Graduação nº 216
Santa Maria, RS, Brasil
2007

Universidade Federal de Santa Maria
Centro de Tecnologia
Curso de Ciência da Computação

A Comissão Examinadora, abaixo assinada, aprova o Trabalho de Graduação

DIGRENDER – RENDERIZADOR DE INTERFACE GRÁFICA PARA WEB EM PHP COM ORIENTAÇÃO A OBJETOS

elaborado por
Carlos Edmilson da Silva Maia

como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação

Comissão Examinadora

Prof. Dra. Roseclea Duarte Medina
(Orientadora)

Prof. João Carlos D. Lima

Prof. Oni Reasilvia de O. Sichonany

Santa Maria, 2 de março de 2007.

RESUMO

Trabalho de Graduação
Ciência da Computação
Universidade Federal de Santa Maria

DIGRENDER – RENDERIZADOR DE INTERFACE GRÁFICA PARA WEB EM PHP COM ORIENTAÇÃO A OBJETOS

Autor: Carlos Edmilson da Silva Maia
Orientadora: Prof. Dra. Roseclea Duarte Medina
Local e Data de Defesa: Santa Maria, 2 de março de 2007

O desenvolvimento de páginas dinâmicas para a *Web* requer, além do conhecimento de alguma linguagem de programação própria para tal, como *PHP: Hypertext Preprocessor* (PHP) ou *Active Server Pages* (ASP), o conhecimento da linguagem de apresentação necessária, como, por exemplo, a *Extensible HyperText Markup Language* (XHTML). O programador precisa escrever um programa na primeira linguagem que gere, para o usuário, a saída na segunda. O DigRender é apresentado como uma solução para esse problema, disponibilizando uma interface de programação que permite ao desenvolvedor criar documentos utilizando fundamentos de orientação a objetos. O DigRender é, então, capaz de gerar uma página, a partir do documento criado pelo desenvolvedor, no formato de saída escolhido pelo mesmo, utilizando, para tal, conceitos de renderização. A ferramenta também é capaz de suportar vários formatos de saída e descobrir qual desses formatos é o ideal para ser entregue ao cliente. Também é possível para um programador implementar suporte a um formato de saída específico, caso assim o deseje. A ferramenta suporta, inicialmente, saída no formato XHTML, garantindo, ao mesmo tempo, compatibilidade com as normas da *World Wide Web Consortium* (W3C).

Palavras-chave: web; php; xhtml; renderização; w3c

ABSTRACT

Trabalho de Graduação
Computer Science
Universide Federal de Santa Maria

DIGRENDER – WEB INTERFACE RENDER FOR OBJECT-ORIENTED PHP

Author: Carlos Edmilson da Silva Maia
Advisor: Prof. Dra. Roseclea Duarte Medina
Place and Date of Presentation: Santa Maria, March the 2nd, 2007

The development of dynamic pages for the web requires, besides the knowledge of some programming language with that purpose like PHP: Hypertext Preprocessor (PHP) or Active Server Pages (ASP), the knowledge of a presentation language like, for instance, the Extensible HyperText Markup Language (XHTML). The programmer needs to write a program in the former languages that generates, for the user, an output on the later. DigRender is presented as a solution to this problem, delivering a programming interface that allows the programmer to create documents using the object-oriented paradigm. DigRender is able to generate a page, based on the programmer-created document, in the output format chosen by the developer. To do so, it uses rendering concepts. The tool also supports several output formats and is able to discover which one of the formats should be used for a specific client. It is also possible for the programmer to implement support to any other output formats if he needs to. The tool supports, inittially, output in the XHTML format, ensuring, at the same time, compatibility with the World Wide Web Consortium (W3C) norms.

Keywords: web; php; xhtml; rendering; w3c

LISTA DE SIGLAS

API – *Application Programming Interface*

ASP – *Active Server Pages*

CGI – *Common Gateway Interface*

CSS – *Cascading Style Sheets*

MVC – *Model View Controller*

PHP – *PHP: Hypertext Preprocessor*

W3C – *World Wide Web Consortium*

WWW – *World Wide Web*

XHTML – *eXtensible HyperText Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A *World Wide Web* (WWW) é uma forma de visualizar toda a informação *online* disponível na Internet como um ambiente navegável e contínuo (BERNERS-LEE et al., 1993). Tendo em vista a velocidade com a qual as informações são disponibilizadas e alteradas na Internet, além da necessidade de acesso a sistemas remotos utilizando-se a WWW, cada vez mais sites na *Web* são desenvolvidos para serem gerados de forma dinâmica, baseados em informações guardadas em bancos de dados, na requisição feita pelo usuário ou por quaisquer outros dados que sejam relevantes para uma determinada página.

Existem linguagens de programação que suportam ou são adequadas para a tarefa de gerar páginas de Internet, sendo que algumas dessas linguagens foram criadas especificamente com esse fim. Entre as linguagens que oferecem esse suporte podemos citar ASP, PHP, ColdFusion, Java e Perl. No momento em que se almeja criar dinamicamente um documento formatado para ser visualizado em um *Web Browser*, surge o requisito de ter o desenvolvedor do site o conhecimento de como utilizar a linguagem de apresentação para o usuário, seja ela HTML, XHTML, WML ou qualquer que seja a linguagem que deve ser entregue ao navegador do usuário para que seja mostrada a informação desejada. Com isso, além da qualificação e conhecimento do desenvolvedor na linguagem de programação escolhida, torna-se também necessário um bom conhecimento na linguagem de apresentação que deve ser gerada para a página em questão.

A boa utilização de um formato de estruturação do documento para ser mostrado em um *browser* é imprescindível para que o resultado seja compatível com os navegadores que devam suportá-lo. É preciso seguir normas e ter conhecimento dos parâmetros a serem utilizados para conseguir os resultados esperados. A única forma de se garantir que as páginas desenvolvidas funcionem em qualquer navegador é seguir as normas vigentes para a autoria das páginas (ZELDMAN, 2001).

No caso do desenvolvedor da página ou site desejar suportar mais de uma linguagem de apresentação, o problema torna-se ainda maior. As diferentes linguagens que o desenvolvedor possa querer como resultado possuem suas próprias peculiaridades, suportando diferentes estruturas e parâmetros e tendo funcionalidades com comportamento diverso. Em

muitos casos, seria obrigatório escrever uma mesma página em todas as linguagens que se deseja suportar, o que tornaria o desenvolvimento do site mais demorado, mais caro e com um produto mais difícil de se fazer manutenção e oferecer suporte.

Tendo em vista esse quadro, torna-se desejável a disponibilidade de uma ferramenta que seja capaz de resolver esses problemas para o programador, retirando do mesmo a responsabilidade de se preocupar com detalhes específicos de como deve ser gerada a página para o usuário, questões de compatibilidade e suporte a linguagens diferentes.

Uma solução possível para esse problema seria um sistema que disponibilizaria uma *Application Programming Interface* (API) que permita a um programador descrever, de forma programacional (isto é, utilizando-se fundamentos de orientação a objetos), a página a ser exibida, cabendo a esse sistema gerar o código final dos documentos a serem entregues ao usuário.

1.1 Objetivos deste trabalho

O principal objetivo deste trabalho é pesquisar, planejar e desenvolver uma solução para a geração de código de apresentação de páginas de forma dinâmica e automática a partir de uma entrada descrita de forma programacional por um desenvolvedor PHP. A idéia é disponibilizar uma API com orientação a objetos que permita ao programador criar e acessar objetos que irão representar elementos da estrutura de apresentação do documento como figuras, *links*, tabelas e assemelhados. Dessa forma, o desenvolvedor precisará somente conhecer os elementos suportados pela ferramenta, sem precisar ter conhecimento aprofundado acerca das linguagens de saída que deseja que sua aplicação suporte.

Baseado na estrutura descrita pelo desenvolvedor, a ferramenta é capaz de gerar uma saída para visualização, baseada em parâmetros, configurações e definições específicas por linguagem de saída. Cada linguagem que se pretende oferecer suporte pode ser descrita com a implementação de uma interface de saída que será acessada pela ferramenta. Essa interface terá uma estrutura descrita pelo sistema, permitindo que qualquer programador PHP com experiência na linguagem de destino possa estender o sistema, implementando uma interface que ofereça suporte a uma nova linguagem de saída. Essa interface irá aproveitar as

funcionalidades de uma linguagem orientada a objetos para a sua implementação.

A ferramenta, denominada DigRender, também assume a tarefa de preocupar-se com os padrões e normas para a geração das páginas, além de ser capaz de detectar, de acordo com os parâmetros enviados pelo navegador que acessa a página, qual seria o formato ideal de geração da página. Os padrões e normas serão seguidos por cada interface que seja implementada para servir como linguagem de saída, permitindo facilmente que diferentes exigências sejam atendidas para diferentes linguagens. Isso permitirá que uma mesma página seja acessada por um número maior de pessoas, que podem estar utilizando diversos navegadores diferentes para acessar o conteúdo disponível.

Inicialmente o projeto teve o foco no desenvolvimento da ferramenta em si e na criação de uma interface capaz de gerar a saída em XHTML. A validação da saída nesse formato, levando em consideração o respeito às normas e a compatibilidade com navegadores, foi utilizada para testar e avaliar a funcionalidade do sistema.

A intenção principal do DigRender é ser utilizado para facilitar e acelerar o processo de desenvolvimento de sistemas *Web* na Decadium Studios, empresa gaúcha fundada em 2004 que trabalha no ramo de desenvolvimento de sites e sistemas para a Internet e no desenvolvimento de entretenimento digital. Espera-se que a utilização desta ferramenta possibilite uma redução no custo de desenvolvimento de conteúdo para a WWW, além de facilitar a manutenção e expansibilidade dos sistemas feitos com a mesma.

1.2 Organização do texto

Na seção 2 serão explicadas linguagens de apresentação, de geração de páginas e renderização para contextualizar o trabalho acerca dos assuntos relevantes ao mesmo. Na seção 2.4 será explorada a importância de se buscar compatibilidade com os navegadores e a influência das conformidade com as normas nesse objetivo, além de mostrar qual a importância disso para este trabalho. Na seção 3, serão discutidas as soluções já existentes para os problemas que o DigRender propõe-se a resolver, dissertando sobre os motivos da busca de uma forma diferente de se resolver os mesmos. Na seção 4 será explicado, de forma detalhada, o funcionamento da ferramenta que será implementada. Por fim, na seção 5 serão

apresentadas as conclusões acerca do projeto.

2 SISTEMAS WEB, LINGUAGENS DE APRESENTAÇÃO DE CONTEÚDO, LINGUAGENS DE GERAÇÃO DE PÁGINAS, RENDERIZAÇÃO, COMPATIBILIDADE E NORMAS

A *World Wide Web* funciona baseada na comunicação e transmissão de dados entre computadores. Um usuário, utilizando um navegador ou ferramenta semelhante – que pode ser um programa instalado em um computador ou embarcado em um dispositivo móvel, como um telefone celular – faz uma requisição a um servidor para visualizar uma determinada página. O servidor, por sua vez, entrega a página de volta ao navegador, que passa a fazer requisições adicionais, se necessário, para carregar figuras, imagens, animações, scripts e outros itens descritos no documento como sendo necessários para a visualização correta do mesmo. A entrega do documento pelo servidor para o cliente pode ser simples como acessar um arquivo no sistema de arquivos e enviar para o navegador ou complexo como entregar a requisição para um programa ou sistema que gera uma saída a ser entregue ao usuário de acordo com a requisição feita. A figura 1 apresenta um diagrama simples que mostra o processo descrito.

Figura 1: Comunicação entre cliente e servidor na requisição de um documento Web

2.1 XHTML

O navegador, ao fazer a requisição do conteúdo ao servidor, envia diversos dados adicionais, informando, inclusive, em que formato ele espera a resposta do servidor. Existem diversos formatos para a entrega desse resultado pelo servidor. O conteúdo requisitado poderia ser, por exemplo, um arquivo executável, um filme ou uma música. Neste trabalho, a preocupação é com a entrega do resultado como um documento formatado.

Dos tipos de documentos formatados, este trabalho irá se concentrar principalmente no formato XHTML. Esse formato é o atualmente recomendado pela *World Wide Web Consortium* (W3C) para se desenvolver páginas para a *Web*. De acordo com a W3C (W3C, 2006), XHTML é uma família de módulos atuais e futuros que reproduzem, organizam e estendem o HTML, reformulando-o em XML. A razão pela qual este trabalho irá se concentrar, pelo menos inicialmente, com a geração de resultados no formato XHTML se deve a essa recomendação da W3C. A W3C foi criada em 1994 para levar a *Web* para o seu potencial máximo, através do desenvolvimento de protocolos comuns e fóruns abertos que promovem sua evolução e asseguram a sua interoperabilidade. O W3C desenvolve tecnologias, denominadas *Web Standards* (ou Padrões *Web*) para a criação e a interpretação dos conteúdos para *Web* (WIKIPÉDIA - W3C). A importância em seguir essa recomendação, bem como uma explicação mais extensa acerca da W3C, será mostrada na seção 2.4.

2.2 PHP

Conforme citado anteriormente, a requisição feita ao servidor pode ser entregue a um programa especializado ao invés de simplesmente se entregar um arquivo encontrado no sistema de arquivos. Esse programa pode, então, decidir o que deverá ser entregue ao usuário, baseando-se em dados de entrada da requisição como qual o navegador que fez o pedido, que tipo de resposta o navegador espera, parâmetros variados etc.

Existem diversas linguagens de programação que foram desenvolvidas ou adaptadas para essa finalidade, dentre as quais podemos citar PHP, ASP, ColdFusion e Java. Este

trabalho terá sua funcionalidade desenvolvida na versão 5 do PHP, utilizando-se o suporte da mesma para orientação a objetos.

Segundo o manual oficial do PHP (ACHOUR et al, 2006), o PHP (acrônimo recursivo para *PHP: Hypertext Preprocessor*) é uma linguagem de script Open Source amplamente utilizada e de uso genérico, sendo especialmente apta para o desenvolvimento de páginas para a Web.

Entre as razões para a escolha da ferramenta PHP para o desenvolvimento deste trabalho, podemos destacar:

- Vasta experiência e familiaridade do autor com a linguagem;
- O fato de quase a totalidade dos projetos *Web* da Decadium Studios já serem desenvolvidos em PHP e que essa é a linguagem de escolha da empresa para esse tipo de desenvolvimento, o que possibilitaria a agregação e integração do DigRender com projetos já existentes e futuros;
- O PHP é a mais amplamente utilizada linguagem de programação utilizada na Web, com mais de 40% das aplicações *Web* utilizando-a. Está instalada em mais de 22 milhões de servidores e possui mais de 2,5 milhões de programadores (LERDORF; TATROE; MACINTYRE, 2006). Isso tudo faz com que o desenvolvimento de um novo projeto utilizando esta linguagem possibilite uma maior aplicação comercial do mesmo devido à maior quantidade de desenvolvedores familiarizados com PHP e de servidores com suporte a este tipo de tecnologia. Segundo a experiência do autor, servidores com suporte a PHP são, em geral, mais baratos e em maior número do que os de outras linguagens para programação *Web*.

2.3 Renderização

De acordo com Arms (ARMS, 2000), renderizar é transformar informação digital para ser mostrada na tela de um computador ou em outra forma de apresentação para o usuário. Ou seja, a renderização é gerar uma saída compreensível para o usuário a partir de uma representação lógica da informação.

Para este trabalho, renderizar será transformar a representação lógica de um

documento, definida com a ajuda da ferramenta DigRender, em um documento formatado que esteja de acordo com o que o navegador do usuário espera como resposta, para que a informação seja apresentada da forma devida.

2.4 Compatibilidade e normas

Um dos aspectos e exigências mais importantes no desenvolvimento de páginas, sites e sistemas para a *Web* é a compatibilidade com navegadores e outras formas de visualização do documento. Em geral, um documento publicado na Internet tem a intenção de poder ser visualizado por qualquer pessoa interessada no mesmo. A partir do momento em que um determinado documento não pode ser visualizado pelas pessoas que seriam o público-alvo porque o navegador utilizado não reconhece a página como documento válido, estamos perdendo um acesso por problemas de compatibilidade. Isso pode significar uma idéia desperdiçada, um cliente perdido ou uma exigência que não é mais atendida.

A melhor forma, no momento, de se conseguir uma maior compatibilidade com o software existente e com o que será desenvolvido futuramente é seguir as normas e padrões da W3C. Zeldman (ZELDMAN, 2001) confirma isso. Segundo ele, a W3C criou recomendações como XHTML e CSS para que a Web não se fragmentasse em dispositivos e navegadores cada vez mais incompatíveis entre si, mas que funcionasse para todo mundo. Esse autor também afirma que a única forma que os desenvolvedores podem ajudar a alcançar esse objetivo é criando conteúdo seguindo essas recomendações.

A *World Wide Web Consortium* (W3C) é um consórcio internacional onde suas organizações membro, uma equipe em tempo integral e o público trabalham juntos para o desenvolvimento de padrões para a *Web*. A missão do consórcio é liderar a Web ao seu potencial máximo, desenvolvendo protocolos e orientações para garantir o crescimento, a longo prazo, da Web (JACOBS, 2006).

Conforme foi citado na introdução, quando um desenvolvedor precisa gerar uma página ou documento para a Internet de forma dinâmica (por exemplo, apresentando informações obtidas a partir de um banco de dados), é necessário que ele seja qualificado com a linguagem que ele usará para tal (como PHP). Além disso, ele tem que estar familiarizado

com a linguagem de destino do documento gerado, como XHTML. E, para garantir uma maior compatibilidade da página criada, é necessário também que o desenvolvedor conheça as normas definidas pela W3C no momento de escrever o código que será gerado pelo seu programa.

No momento em que o DigRender assume a responsabilidade de gerar o conteúdo a partir da estrutura criada pelo programador usando o mesmo, passa a ser responsabilidade também do DigRender garantir que o código que ele estará gerando siga as normas necessárias. Mais especificamente, isso é responsabilidade da interface que foi implementada para a linguagem de saída sendo usada no momento de gerar a saída. O sistema de interface, assim como a estrutura do DigRender, será explicado na seção 4 deste documento, que é dedicada a descrever o sistema de forma detalhada.

3 SOLUÇÕES PARA GERENCIAMENTO DE LAYOUT

O problema de gerar a saída para uma página *Web* não é um fato novo. Desde os primeiros programas escritos para funcionar em *Common Gateway Interface* (CGI) já surgem soluções das mais variadas para resolver esse problema, e a partir dessas soluções surgiram novas linguagens destinadas a ter a solução do problema como funcionalidade padrão, e sobre essas linguagens surgiram novos sistemas que resolve o problema de forma mais completa ou fácil de utilizar. O DigRender é um desses sistemas, que tem seu funcionamento implementado sobre o PHP, uma linguagem desenvolvida para gerar páginas de Internet de forma dinâmica.

As soluções mais utilizadas atualmente implementam ou utilizam conceitos de MVC ou são baseadas em sistemas de *templates*, itens discutidos a seguir.

3.1 MVC

O *Model View Controller* (MVC) é um padrão de arquitetura de aplicações que consiste em três tipos de objetos: o *Model* (modelo), objeto de aplicação, a *View* (visualização), apresentação na tela, e o *Controller* (controlador), que define como a interface reage à entrada do usuário (GAMMA et al., 1995). Esse padrão permite que a mesma lógica de negócios possa ser acessada e visualizada por várias interfaces.

O padrão MVC é comumente utilizado no desenvolvimento de sistemas *Web*. O DigRender não implementa esse padrão, mas, da mesma forma que os sistemas de *templates*, pode ser utilizado para o desenvolvimento da interface do usuário, permitindo a separação desta dos outros componentes descritos pelo padrão.

Existem ferramentas mais completas que implementam o MVC de forma mais abrangente, como o *Struts*, framework da *Apache Foundation*, o *Spring Framework*, um *framework* de aplicação para Java, o *Google Web Toolkit*, um framework de desenvolvimento de software também em Java, e o *CakePHP*, um framework que implementa MVC em PHP.

Esse tipo de ferramenta oferece uma solução mais completa para o desenvolvimento de um site de grande porte.

As razões pelas quais o DigRender foi desenvolvido como ferramenta à parte das já existentes incluem a necessidade de uma solução em PHP pequena e modular que possa ser acoplada a projetos já existentes com pouco esforço, além de suportar um conjunto específico de funcionalidades (suporte a várias linguagens de saída, detecção do formato ideal de entrega) sem a necessidade forçada de adoção do padrão MVC que, em projetos pequenos, poderia acrescentar uma complexidade desnecessária ao processo. A principal intenção do DigRender é ser uma ferramenta pequena, de fácil utilização, que não exija adoção de padrões específicos de desenvolvimento, permitindo que projetos pequenos possam ser desenvolvidos, em vários casos, de forma mais rápida e menos custosa.

As soluções que mais se aproximam da proposta pelo DigRender são melhor representadas pelos já citados sistemas de *templates*, que serão explanados na seção seguinte.

3.2 Sistemas de Templates

Em geral, as soluções partem do pressuposto de que existe um código que deve ser entregue ao navegador – geralmente HTML – e um código que deve ser usado para gerar ou controlar essa apresentação – no caso deste trabalho, um programa escrito em PHP. Com isso em mente, surgiu a idéia inicial de desenvolver formas que separem a lógica de apresentação da lógica de negócio. A idéia, nesse caso, seria ter uma parte do programa, independente, que, baseado nos dados processados e entregue pela lógica de negócio, iria montar, dinamicamente, o código de uma página a ser entregue para o usuário. A outra parte do programa, que teria a lógica do negócio, iria, após realizar todo o processamento necessário relacionado ao pedido do usuário, entregar os dados necessários para a parte que, com a lógica de apresentação, iria gerar o documento a ser entregue para o usuário.

Sistemas que implementam essa funcionalidade são chamados de sistemas de *templates*. Esse nome advém do fato deles serem baseados em arquivos que definem a forma como as páginas devem ser apresentadas (denominados *templates*), sendo essa a parte com a lógica de apresentação no sistema. Geralmente os arquivos de *templates* nesses sistemas são o

código de apresentação em si (por exemplo, em HTML) com informações de pseudo-código embutidos para execução de lógica de apresentação, o que inclui estruturas simples de programação como IF, WHILE, FOR etc.

De forma resumida, pode-se explicar o funcionamento do sistema de templates da seguinte forma: conteúdo (por exemplo, de um banco de dados) e "especificações de apresentação" (em um template web) são combinados (através do sistema de templates) para produção em massa de documentos *Web* (WIKIPEDIA - Web template).

O diagrama apresentado na figura 2 explica, de forma simplificada, como funciona um sistema de templates comum.

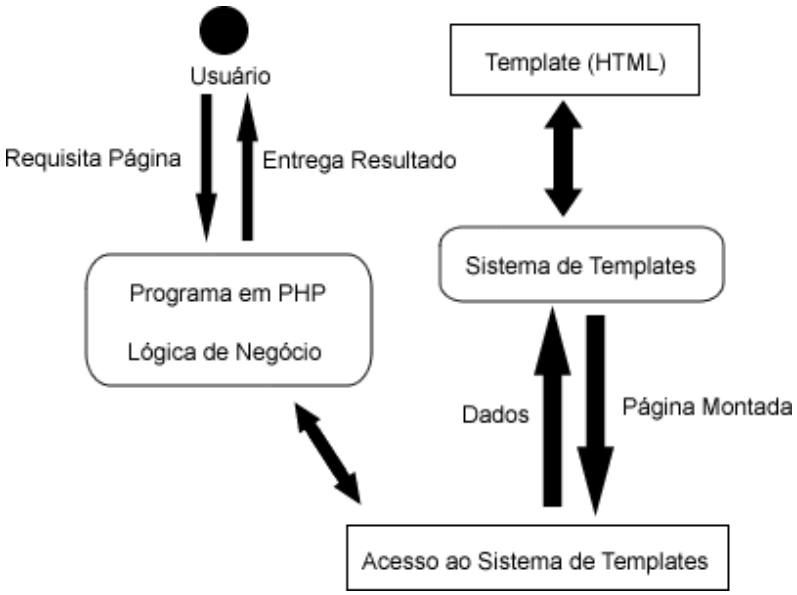

Figura 2: Sistema de Templates

Como pode ser visto na figura, um programa em PHP que utiliza um sistema de templates faz acesso ao mesmo para receber a página pronta. Ele entrega os dados que devem ser usados para preencher os *templates*. O sistema, por sua vez, carrega o *template* especificado pelo programa em PHP e executa o pseudo-código encontrado no mesmo, preenchendo assim a página final com os dados entregues pelo programa. Por fim, o sistema de *templates* entrega o resultado final para o programa ou diretamente para o usuário. O arquivo de *template*, nesse caso, deve ser escrito pelo programador juntamente com a escrita da lógica de negócio, definindo para o sistema de *templaes* o que deve ser feito com as variáveis e dados geradas pelo código do programador no momento do processamento do *template*.

Cada página que deve ser gerada utilizando as funcionalidades desse sistema deve ser escrita em duas partes separadas: a primeira, referente à lógica de negócio; a segunda, o *template* na linguagem de apresentação com a inclusão de pseudo-código ou algo semelhante. Esse tipo de sistema, porém, apresenta vários problemas e desvantagens.

Por exemplo, esse sistema obriga o desenvolvedor a manter dois códigos para cada página a ser gerada, geralmente representando dois arquivos. Esses códigos são, obviamente, o código da lógica de negócio e o código da lógica de apresentação. Manter dois arquivos para cada página pode apresentar várias vantagens, e não é isso que é mostrado aqui como desvantagem, mas sim a obrigatoriedade de que seja feito dessa forma. Não é possível utilizar-se das vantagens e funcionalidades de um sistema de *template* sem ter uma página dividida em pelo menos dois arquivos.

Esse tipo de abordagem na separação da lógica do negócio da lógica de apresentação leva a mais problemas. Por exemplo, o suporte a mais de uma linguagem exige a criação de ainda mais arquivos para cada página. Isto é, para que uma determinada página possa ser gerada, por exemplo, em HTML, WML e em texto puro, caso essa seja uma funcionalidade desejada do sistema, torna-se necessário que haja um arquivo de *template* para cada uma das linguagens às quais se quer oferecer suporte. Neste exemplo, seriam necessários três arquivos de *template* além do arquivo com a lógica de programa. Para um site que tenha, por exemplo, 30 páginas diferentes pelas quais o usuário pode navegar, isso significaria um mínimo de 90 arquivos de *template* para suportar essas linguagens, sem levar em consideração os arquivos para a execução do sistema em si. Uma quantidade desse tamanho de arquivos leva a uma maior dificuldade no gerenciamento do site, dificultando muitas vezes a manutenção do mesmo. Um outro problema que surge é a possibilidade de se descobrir que em alguma das linguagens ocorreu algum erro de escrita que tenha se repetido em todas os templates daquela linguagem. Em um caso desse tipo, seria necessário então corrigir todos os arquivos daquela linguagem, o que, dependendo do erro e das ferramentas, pode ser um processo bastante custoso.

Outra desvantagem é o custo para suportar uma nova linguagem de apresentação. Se o desenvolvedor do site decidir por passar a suportar uma nova linguagem como formato de saída, será necessário escrever um novo arquivo de *template* para cada uma das páginas do sistema que utilizam *templates*.

Existe também um erro conceitual em como os sistemas de *templates* foram criados e

como passaram a ser utilizados. Isso é bem explicado por Lozier (LOZIER, 2003), que já utilizou vários sistemas de *templates* (incluindo fastTemplate, Smarty e HTML::Template) e desenvolveu o seu próprio, o bTemplate. Segundo ele, os sistemas atuais para separação de código e *layout* em *templates* falham basicamente em ter a noção errada de que um sistema de *templates* é o mais recomendável para separar a lógica da apresentação. Ele justifica essa afirmação explicando que, apesar de resolver alguns problemas como a separação em si e a possibilidade de ter um código mais complicado em PHP separado do *layout* em HTML, esses sistemas introduzem uma complexidade adicional que seria desnecessária ao sistema, aumentando a quantidade de arquivos a serem controlados e criando processamento adicional que consumiria mais recursos do sistema. Ele também afirma, após ter desenvolvido o seu próprio sistema de *template*, que sistemas de *templates* para PHP não apresentam utilidade real, mesmo porque a própria linguagem já uma *engine* de *templates* por si mesma, e apresenta um exemplo simples de como fazer o que os sistemas de *template* existentes já fazem utilizando-se somente de funcionalidades padrão do PHP.

O objetivo deste trabalho é apresentar uma nova forma de se resolver o problema, criando um sistema para PHP que seja capaz de permitir que um programador desenvolva, de forma mais simples e rápida, um sistema que gere o resultado no formato desejado, sem ter que ter conhecimento do formato em si ou as normas que regem o mesmo. Esse novo sistema também permitirá que se separe a lógica de apresentação da lógica de negócio em arquivos separados sem, contudo, tornar essa prática obrigatória para o uso do sistema. Muito embora essa funcionalidade não seja parte integral do sistema, será perfeitamente possível utilizar o mesmo dessa forma. Isso será citado na seção 4, dedicada a discutir aprofundadamente a proposta deste trabalho.

4 DIGRENDER

O DigRender é uma ferramenta que resolve o problema da geração dinâmica do código para páginas *Web* utilizando conceitos de renderização. A idéia é disponibilizar um conjunto simplificado de ferramentas e classes que possibilitem a construção de um documento lógico a partir de instruções simples executadas por um programa escrito em PHP. A partir desse documento, a ferramenta é capaz de gerar a saída no formato desejado. A figura 3 apresenta um diagrama simplificado do funcionamento da ferramenta, e a figura 4 apresenta um diagrama de classes resumido que foi utilizado para o desenvolvimento da ferramenta.

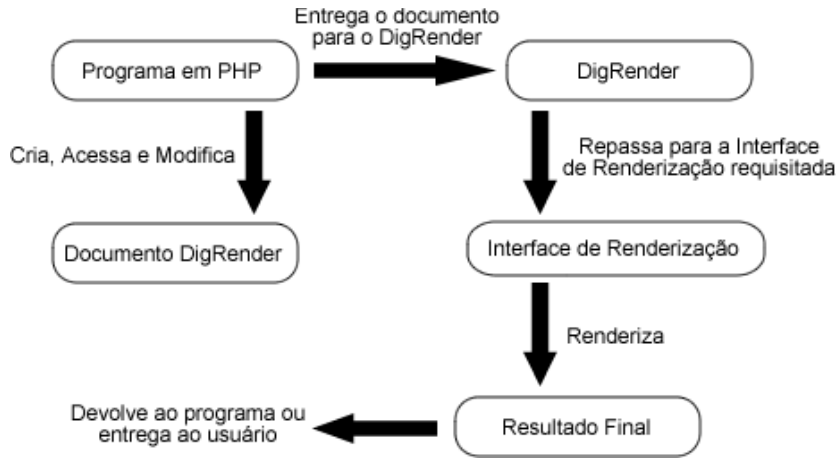

Figura 3: Funcionamento do DigRender

Na figura 3 podemos ver a idéia geral do funcionamento do DigRender. O programa escrito em PHP que deseja utilizar a ferramenta cria um objeto do tipo Documento e passa a adicionar informações e alterar o mesmo conforme for necessário. Quando o uso desse objeto pelo programa estiver completo, o programa pode então entregar o mesmo ao DigRender que, por sua vez, entrega o mesmo à Interface de Renderização adequada – seja ela escolhida pelo programa ou pela própria ferramenta. A Interface, por fim, renderiza o resultado final a ser entregue ao usuário.

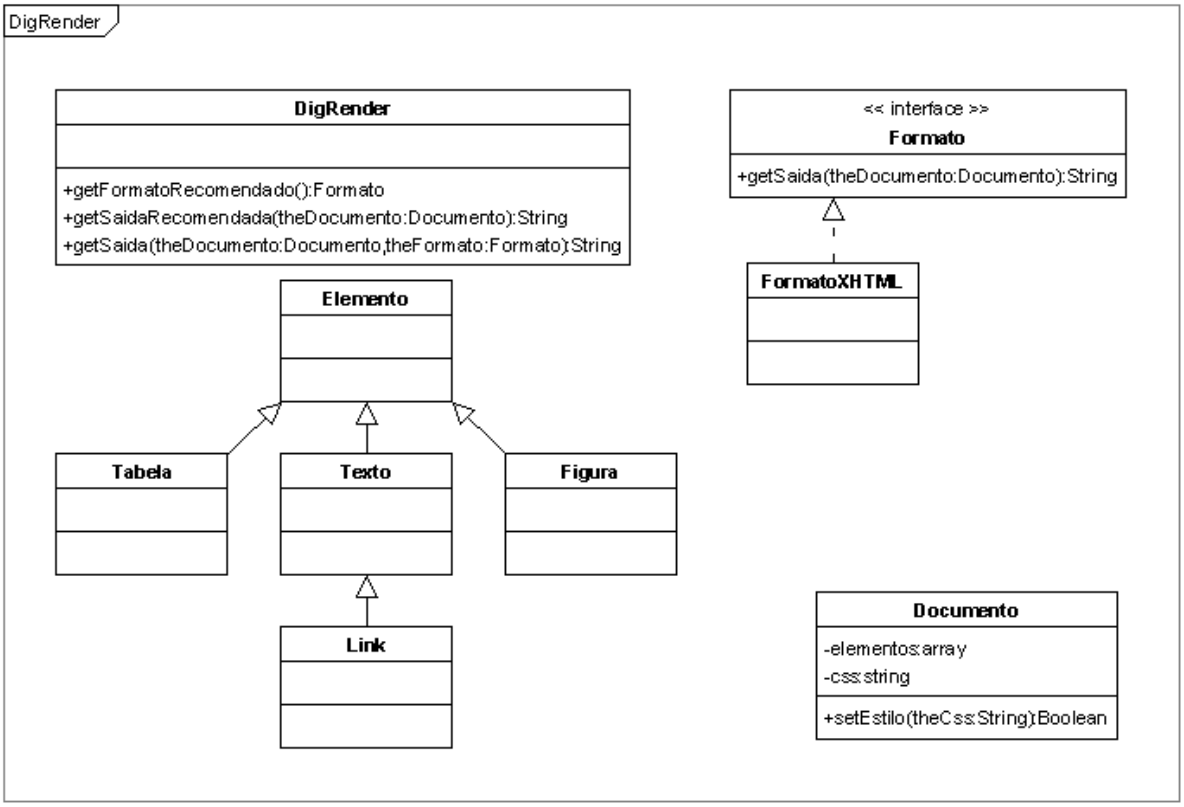

Figura 4: Diagrama de Classes

O diagrama de classes apresentado na figura 4 mostra, de forma geral, as principais classes e a interação entre as mesmas. A interface Elemento é implementada por todos os componentes que devem poder ser usados para a composição do objeto do tipo Documento, dentre os quais podemos destacar tabelas, figuras, textos e links. Também é mostrada, na figura, a classe Formato, que é uma interface que define o que deve constar na implementação de uma Interface de Renderização, componente que será explicado no item 4.2. A implementação da Formato que será inicialmente disponibilizada juntamente com o DigRender será a FormatoXHTML, que oferecerá suporte à renderização do documento no formato XHTML. Também é mostrada no diagrama a classe DigRender, a classe principal que faz o controle da renderização.

4.1 Documento DigRender

O primeiro item de importância no DigRender é o documento definido e disponibilizado pelo mesmo. O documento possui estrutura e funcionamento simples, permitindo que o programador insira, acesse e modifique atributos e elementos do mesmo.

Os atributos são informações relevantes ao documento como um todo, como título, autor e data. Esses atributos podem ou não ser utilizados por cada interface de renderização (as interfaces de renderização são explicadas na seção 4.2) no momento da geração do resultado final. Não é obrigatório definir valores para esses atributos na criação de um documento pelo programador; valores ou comportamentos padrão são definidos quando um determinado atributo não estiver definido. Por exemplo, ao gerar uma página de um documento que não teve o título definido, o renderizador poderá simplesmente colocar “Sem Título” no espaço reservado ao mesmo no resultado final.

Os elementos, por sua vez, são os itens que compõem o documento em si. Texto, tabelas, figuras, links e outros componentes comumente encontrados em páginas Web. Alguns dos elementos podem ser mais complexos, podendo ter elementos aninhados dentro de si, como, por exemplo, uma figura em uma célula de uma tabela. Os elementos, por sua vez, também possuem atributos, que funcionam da mesma forma que os atributos do documento. Esses atributos podem influenciar na aparência ou funcionalidade do elemento em questão e deverão funcionar sem a necessidade de uma definição explícita. Os elementos que não fizerem sentido sem a definição de determinados atributos (por exemplo, um link que não tem definido para onde ele aponta) irão forçar a obrigatoriedade dos mesmos exigindo-os em seu construtor.

Na prática, a geração do documento é realizada com os seguintes passos: a) cria-se o documento, b) definem-se as propriedades relevantes do mesmo (como, por exemplo, título, autor e folha de estilos a ser utilizada), e c) para cada um dos elementos desejados, ele é criado, definido e inserido no documento. Isso pode, obviamente, ser feito durante o curso de um programa normal, permitindo, de forma simples, que o documento seja gerado de forma dinâmica. O programador pode, por exemplo, criar e inserir linhas em uma tabela de acordo com os resultados de uma consulta ao banco de dados. Também será uma funcionalidade do documento a navegação no mesmo, permitindo ao programador editar o documento após a

criação e modificação do mesmo, removendo, alterando ou inserindo novos elementos conforme sua vontade.

O documento gerado pode, então, ser repassado, através do DigRender, para uma interface de renderização.

4.2 Interface de Renderização

Uma interface de renderização é uma classe que instancia um objeto capaz de gerar a saída necessária a partir de um documento DigRender. O DigRender define o que uma interface de renderização deve ser capaz de fazer, e fica a cargo de qualquer programador PHP capacitado desenvolver quaisquer interfaces específicas que suportem a saída no formato esperado.

A interface de renderização é capaz de conseguir as informações necessárias para gerar a saída do documento através da utilização de métodos do documento e do DigRender. Com esses métodos, ela conseguirá descobrir os elementos a serem renderizados e seus atributos, bem como os atributos do documento em si. Ela pode, então, gerar a saída da forma que achar melhor.

O DigRender possui, inicialmente, uma interface de renderização para XHTML. Essa interface será utilizada para o desenvolvimento e avaliação da ferramenta.

4.3 Funcionalidades Adicionais

O DigRender tem uma série de funcionalidades adicionais que facilitam o uso da ferramenta, realizando tarefas comuns ao funcionamento da mesma. A intenção de prover esse tipo de suporte é retirar ainda mais do usuário da ferramenta obrigações e preocupações relacionadas diretamente com a geração e entrega da página ao usuário. Das funcionalidades adicionais, podemos citar:

- Detecção do formato de saída. A ferramenta é capaz de descobrir qual das

interfaces de renderização disponíveis é a ideal para ser utilizada, baseando-se, para tal, em parâmetros passados pelo navegador do usuário e repassados à ferramenta pelo PHP como tipo de navegador, resposta esperada, etc.

- Entrega do resultado ao usuário. O DigRender fornece métodos que renderizam a página entregando a mesma diretamente ao usuário, decidindo inclusive quais métodos adicionais podem ser utilizados na entrega do mesmo, como compactação da página através de *gzip* e *bufferização* do resultado.

4.4 Vantagens

Há uma série de vantagens em se utilizar um sistema como o DigRender para resolver o problema de geração de páginas Web, algumas delas já citadas neste documento. Abaixo seguem alguns dos principais ganhos na utilização dessa ferramenta.

- Separação da lógica do negócio da lógica do sistema.
- Suporte a vários formatos de saída para um mesmo código.
- Redução da complexidade e custo do desenvolvimento para Web.
- Redução da equipe necessária para se desenvolver páginas dinâmicas.

O DigRender também pode oferecer algumas utilidades além do seu escopo original. Muito embora a utilidade principal do DigRender seja criar páginas para a Web, um programador pode implementar interfaces de renderização customizadas que gerem a saída nos mais variados formatos. O programador poderia, por exemplo, implementar uma interface que gere saída em Latex, ou uma saída em arquivo texto puro.

4.5 Limitações

A forma pela qual o DigRender resolve o problema de geração de páginas dinâmicas, ao mesmo passo que retira do utilizador da ferramenta a necessidade de se lidar diretamente com o código de saída, também o impede de fazê-lo. A ferramente é quem gera o código de

saída. Logo, se o programador que está utilizando a ferramenta precisar gerar algum código específico que não é gerado pelo DigRender, ou se o DigRender não suporta determinado elemento da forma como o programador espera (até mesmo não suportando por completo), ele não poderá fazê-lo através da ferramenta.

A solução que foi adotada para amenizar essa limitação é disponibilizar um elemento de documento especial que permitirá ao programador inserir código de saída customizado no documento. Isso, contudo, tornaria o programa desenvolvido menos portável para outras linguagens de saída, além de colocar de volta nas mãos do desenvolvedor a responsabilidade de escrever diretamente o código de saída, mesmo que somente uma parte. Esse novo código, por não ser parte da ferramenta, também não tem garantida a sua conformidade com as normas.

O que foi feito para evitar a necessidade de se utilizar esse tipo de solução é planejar a estrutura de documento e escrever as interfaces de renderização da forma mais completa e correta possível.

4.6 Documentação

Para uma boa utilização por parte do programador de uma ferramenta como o DigRender, é imprescindível a disponibilidade de uma documentação completa e acessível. Tendo isso em mente, o DigRender foi desenvolvido contendo em seu código diversos comentários que podem propiciar ao usuário da ferramenta um melhor entendimento acerca da mesma.

Além disso, foram utilizadas várias marcações específicas nesses comentários, identificando arquivos, classes, métodos e variáveis, entre outros itens. Além de seguirem um padrão de documentação que pode facilitar para o programador o acesso a certas informações, essas marcações são entendidas por diversos ambientes de programação, como o *Zend Studio* (http://www.zend.com/studio), que são capazes de, por exemplo, completar automaticamente código relacionado ao DigRender que o programador esteja digitando e apresentar documentação de forma dinâmica para classes e métodos que estejam sendo utilizado em um determinado momento.

Uma outra utilidade dessas marcações é a possibilidade de se utilizar uma ferramenta como o *phpDocumentor* (http://www.phpdoc.org), que é capaz de gerar uma documentação baseada nessas marcas. Essa ferramenta é capaz de gerar documentação em vários formatos diferentes (HTML, PDF, CHM e XML DocBook), oferecendo, dessa forma, um manual completo das classes e métodos do DigRender no formato de preferência do programador.

4.7 Avaliação

As características desejáveis do DigRender, durante o seu funcionamento, são estabilidade, conformidade com as normas, previsibilidade e desempenho, nessa ordem de importância.

A estabilidade se refere ao funcionamento sem problemas ou erros perceptíveis, tanto ao usuário quanto ao programador. A ferramenta deve funcionar sem apresentar “bugs” que impeçam a boa utilização do mesmo. Isso foi avaliado durante o desenvolvimento da ferramente e durante a avaliação dos outros itens através de testes simples. Como procedimento normal nesse tipo de projeto, os erros de programação que não puderam ser evitados durante a fase de planejamento foram corrigidos baseados nas possíveis ocorrências durante o desenvolvimento da ferramenta.

A conformidade com as normas representa a necessidade de as páginas geradas pelo DigRender estarem de acordo com as normas estabelecidas para tal. Como o DigRender suportará, inicialmente, saída no formato XHTML, essa avaliação foi feita a partir da validação das páginas geradas por ferramentas da W3C. Cada novo elemento desenvolvido para o documento DigRender foi utilizado em páginas de exemplo que, além de permitir detecção de erros a serem corrigidos conforme citado anteriormente, também permite a utilização do código gerado nas verificações disponibilizadas no próprio site da W3C. No caso de alguma página gerada não ter verificada sua conformidade com as normas, o motivo para tal foi descoberto e a interface de renderização foi corrigida para atender às normas.

A questão de previsibilidade atende à necessidade de o resultado gerado pela ferramenta ser o esperado a partir do documento descrito pelo programador. Como já foi citado, cada elemento desenvolvido foi testado individualmente e em conjunto em páginas de

exemplo, permitindo, assim, avaliar se o resultado obtido foi o esperado. A avaliação neste quesito foi positiva.

Por fim, o desempenho, referente ao tempo computacional que o DigRender ocupa, foi avaliado, após os outros itens terem sido verificados, através de uma comparação direta do DigRender com o *Smarty*. O *Smarty* é um sistema de *templates* que realiza a tarefa de auxiliar o programador na tarefa de gerenciar a entrega da interface gráfica para o usuário. O *Smarty* foi escolhido para ter o seu desempenho comparado ao do DigRender por possuir suporte a pré-compilação de *templates* e carregamento sob demanda das diversas funcionalidade que provê, caracterizando-se como uma solução com boa *performance* para o mesmo problema que o DigRender resolve. Além disso, dentre os sistemas de *templates* para PHP mais amplamente utilizados, o *Smarty* é o que possui funcionamento geral mais próximo do ideal esperado para um sistema desse tipo (LOZIER, 2003).

Os resultados da avaliação de desempenho são apresentados pela Figura 5.

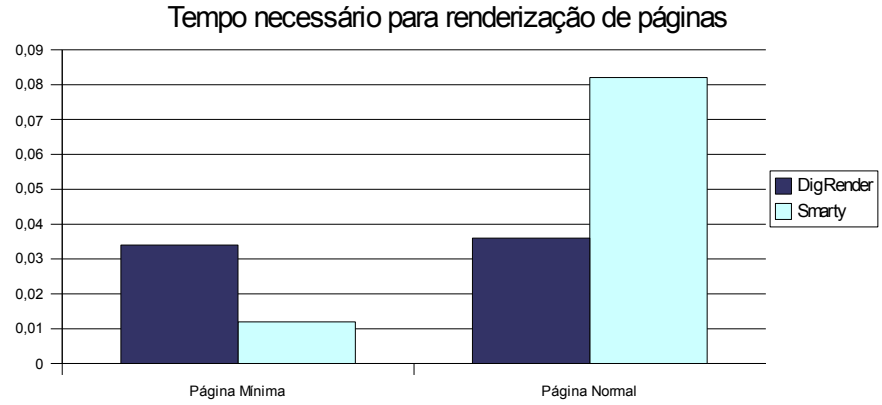

Figura 5: Gráfico de comparação de desempenho entre DigRender e Smarty

O gráfico apresentado na figura 5 mostra que o DigRender possui um desempenho pior do que o *Smarty* na geração de uma página mínima, levando mais tempo que o sistema de *templates* para gerar seu resultado. Para essa medição, a página mínima foi uma página com nada além de uma única linha de texto.

No caso da medição de uma página normal, foi utilizada uma página com mais conteúdo, sendo assim mais semelhante do que seria uma página típica da *web*, contando com alguns parágrafos, linhas de tabelas geradas de forma dinâmica e um formulário. Nesse exemplo, o DigRender leva menos tempo que o *Smarty* para entregar a saída final ao usuário.

Analisando-se esses resultados, pode-se afirmar que uma página que utilize o DigRender consome um tempo maior na inicialização do mesmo do que gastaria na inicialização de um sistema como o *Smarty*. Acredita-se que isso seja devido ao fato de o DigRender possuir uma estrutura interna mais complexa, exigindo o carregamento e inicialização de diferentes estruturas a serem utilizadas na construção do documento, bem como o carregamento da interface de renderização a ser utilizada. Essa modularidade e expansibilidade inerentes ao DigRender tornam a inicialização do mesmo um pouco mais demorada, especialmente considerando-se que PHP é utilizado como uma linguagem interpretada, e não compilada, em servidores *web*. O *Smarty* também tem uma vantagem nesse sentido, pois ele mantém os arquivos de *templates* pré-compilados, acelerando o tempo necessário para sua inicialização.

Em compensação, após vencido o tempo de inicialização, o DigRender apresenta um desempenho melhor do que o *Smarty*, sendo isso mais perceptível em páginas com mais do que somente algumas linhas de texto. Com isso em mente, é possível afirmar que o DigRender possui um desempenho melhor do que o *Smarty* em aplicações reais, visto que a maior parte das páginas *web* possui mais conteúdo (tabelas, parágrafos, links etc) do que apenas algumas linhas. Portanto, a avaliação no quesito desempenho também é positiva.

5 CONCLUSÃO

O desenvolvimento deste trabalho tinha como intenção permitir a criação de uma solução nova para o problema já existente de se gerar a interface de uma página *web*. Com isso, é possível munir-se com uma nova ferramenta que poderá ser utilizada para futuros projetos que apresentem a necessidade do uso da mesma.

A ferramenta desenvolvida acabou apresentando desempenho melhor do que soluções semelhantes, passando a ser, além de tudo, uma opção viável quando isto é uma preocupação. Também foi positivo o resultado com a questão da modularidade e expansibilidade da ferramenta, que foi desenvolvida de tal forma que criar novas interfaces de renderização e elementos do documento, além de possível, é descomplicado.

Em síntese, é possível afirmar que os objetivos deste trabalho foram atingidos e que o resultado final teve uma qualidade ligeiramente superior ao esperado. Isto, juntamente com o fato do DigRender ser expansível, torna a ferramenta não só utilizável, como também sendo uma solução que poderá ser aplicada e melhorada a longo prazo.

5.1 Trabalhos Futuros

O DigRender foi desenvolvido utilizando-se princípios de orientação a objetos compatíveis com o suporte para tal disponibilizado pelo PHP. Isso foi feito de tal forma que tenha sido promovida uma boa modularidade do código, possibilitando a adição de novas funcionalidades ao mesmo sem muita dificuldade, em muitos casos sendo somente necessário a adição de algumas classes.

Pode-se citar, como trabalho futuro mais imediato para o DigRender, o desenvolvimento de novas interfaces de renderização que ofereçam suporte a novas linguagens de saída e a criação de novos elementos de documento para cobrir quaisquer componentes do mesmo que não estejam presentes. O DigRender foi desenvolvido de tal forma que a adição de novas interfaces de renderização e de novos elementos de documento

não é capaz de impedir o suporte a páginas já existentes que utilizam a ferramenta, permitindo uma menor preocupação com a questão da compatibilidade com versões anteriores.

Pode-se, ainda, citar a idéia da criação de um complemento para o DigRender que lide com *Cascading Style Sheets* (CSS). CSS é uma linguagem de estilo utilizada para definir a apresentação de documentos escritos em uma linguagem de marcação, como HTML ou XML (WIKIPÉDIA - Cascading Style Sheets). Ela é muito útil para definir estilos e aparência de uma página *web*, sendo recomendada pela W3C e bastante utilizada nas páginas existentes atualmente. O DigRender possui suporte a CSS, permitindo, entre outras coisas, que seja definido um arquivo CSS para uma página e a classe CSS de cada elemento de um documento. A idéia do complemento vislumbra a criação de um formato de estilo, semelhante ao formato de documento, permitindo que o programador descreva o estilo dentro de seu próprio código PHP da mesma forma como pode fazer com o documento. Essa descrição de formato, então, poderia ser utilizada para a geração dinâmica do CSS, retirando a responsabilidade do programador de conhecer mais essa linguagem. Isso também possibilitaria outras funcionalidades, como definir marcações de estilo e *layout* para interfaces de renderização que geram linguagens de apresentação que não suportam CSS.

Por fim, é importante aplicar o uso da ferramenta em novos projetos que venham a ser desenvolvidos em PHP ou adaptar projetos já existentes para a utilização do DigRender, através da aplicação gradual do uso da ferramenta para gerar a saída do mesmos. Com isso, passa a ser possível um teste mais amplo das funcionalidades da ferramenta, pois ela estará sendo utilizada em ambientes reais de trabalho que permitirão uma percepção mais clara de possíveis falhas e erros que porventura ainda existam no código do DigRender.

6 REFERÊNCIAS BIBLIOGRÁFICAS

ACHOUR, Mehdi et al. PHP Manual, 2006

ARMS, William Y.. Digital Libraries, 2000

BERNERS-LEE, T.; CAILLIAU, R.; PELLOW, N; SECRET, A. The World-Wide Web Initiative, 1993

GAMMA, Erich; HELM, Richard; JOHNSON, Ralph; VLISSIDES, John. Design Patterns: Elements of Reusable Object-Oriented Software, 1995

JACOBS, Ian. About W3C. Disponível em: <http://www.w3.org/Consortium/>. Acesso em: 25 nov. 2006.

LERDORF, Rasmus; TATROE, Kevin; MACINTYRE, Peter. Programming PHP, 2006

LOZIER, Brian. Template Engines. Disponível em: <http://www.massassi.com/php/articles/template_engines/>. Acesso em: 25 nov. 2006.

W3C. W3C HTML Home Page. Disponível em: <http://www.w3.org/MarkUp/>. Acesso em: 2 dez. 2006.

WIKIPEDIA. Web template - Wikipedia, the free encyclopedia. Disponível em: <http://en.wikipedia.org/wiki/Web_template>. Acesso em: 10 jan. 2007.

WIKIPÉDIA. Cascading Style Sheets - Wikipédia. Disponível em: <http://pt.wikipedia.org/wiki/Cascading_Style_Sheets>. Acesso em: 20 fev. 2007.

WIKIPÉDIA. W3C - Wikipédia. Disponível em: <http://pt.wikipedia.org/wiki/W3C>. Acesso em: 8 jan. 2007.

ZELDMAN, Jeffrey. Why Don't You Code for Netscape?. Disponível em: <http://alistapart.com/stories/netscape/>. Acesso em: 25 nov. 2006.