CENTRO UNIVERSITÁRIO FEEVALE

RAFAEL HENS RIBAS

DESENVOLVIMENTO DE UMA FERRAMENTA LATEX PARA WEB

Novo Hamburgo

2008

RAFAEL HENS RIBAS

DESENVOLVIMENTO DE UMA FERRAMENTA LATEX PARA WEB

Centro Universitário Feevale
Instituto de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação
Trabalho de Conclusão de Curso

Professor Orientador: Paulo Roberto Ferreira Junior

Novo Hamburgo

2008

AGRADECIMENTOS

Gostaria de agradecer a todos os que, de alguma maneira, contribuíram para a realização desse trabalho de conclusão, em especial:

Aos meus pais, que proporcionaram e apoiaram minha jornada até aqui, atém de todo o amor e carinho.

A minha noiva Aline, pelo amor, companheirismo e total apoio durante a realização deste.

A D. Diva, por apoiar, compreender e ajudar na tarefa de correção ortográfica.

Ao meu professor orientador, pela contribuição, dedicação e atenção, que foram de grande importância.

RESUMO

Já faz algum tempo que os aplicativos têm migrado para a Web com o objetivo de dar onipresença a seus usuários. Os webmails, por exemplo, vêm substituindo os clientes de e-mail tradicionais por permitirem que usuários manipulem suas correspondências a partir de qualquer computador conectado a Internet. Atualmente, seguindo esta mesma tendência, os editores de texto têm sido implementados para a utilização via Web, onde se destaca o Google Docs. No cenário das ferramentas para edição de documentos atuais, podemos encontrar duas classes distintas de processadores de textos: os convencionais, como o Word, OpenOffice e Google Docs, que são totalmente voltados à metodologia de proporcionar o maior conforto, abrangência e facilidade, utilizando menus intuitivos e demonstrando passo-a-passo a formatação do texto; e os científicos, como o Latex, que utilizam uma linguagem de formatação focada na abstração do texto em relação a sua aparência final oferecendo recursos sofisticados para a utilização de símbolos matemáticos, notação para formalização científica e para a descrição de pseudo-algoritmos. Há uma carência de editores científicos disponíveis para a Web. Hoje não se conhece nenhum totalmente funcional, simples e disponível gratuitamente. Sendo assim, este projeto de conclusão de curso tem como objetivo estudar e desenvolver uma ferramenta de processador de texto, baseada em Latex, para utilização via Web (através de um navegador) seguindo a atual convergência tecnológica dos aplicativos.

Palavras-chave: Latex. Processador de Texto. Internet. Mobilidade e Onipresença.

ABSTRACT

Aplicatives (APP) have already migrated to the Web many years ago in order to give to the users an omnipresence feature. Webmails for example are replacing the traditional customer's e-mails to avoid any user to manipulate their mails from any other computer connected to the internet. Presently, according to the same trend, the edition programs have been implemented to be used through the Web like the Google Docs to which we must call special attention. Nowadays, in terms of tools for editing documents we can find two different types of editors: the conventional ones such as Word, Open Office and Google Docs which are totally focused on the methodology of providing the greatest confort coverage and ease, using intuitive menus and showing step-by-step the text formatting process and the scientific ones, such as Latex which use the formatting language focusing on the abstract aspect of the text regarding its final appearance offering sophisticated resources for the use of mathematical symbols, notes for scientific formalizations and the descriptions of pseudo-algorithms. There is a lack of available scientific publishers to the Web today. Presently, we do not know any fully functional, simple and free to download and use it. Therefore, the realization of this project as a pre-requisite of college graduation aims to explore and develop a tool for edition based on Latex, to be used by the Web (using a browser) following the current technological convergence of applicatives.

Key-words: Latex. Edition Programms. Internet. Mobility. Omnipresence.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

WYSIWYG	What You See Is What You Get
AJAX	Asyncronous Javacript and XML
RSS	Really Simple Syndication
HTML, HTM	Hyper Text Markup Language
PDF	Portable Document Format
RTF	Rich Text Format
CSS	Cascading Style Sheets
KB	Kilobyte
JVM	Java Virtual Machine
XML	eXtensible Markup Language
URL	Uniform Resource Locator
TXT	Text File
DPI	Dots Per Inch
JPEG	Joint Photographic Experts Group
ABNT	Associação Brasileira de Normas Técnicas
DOC	Microsoft Word Document
PPT	Microsoft Power Point Document
XLS	Microsoft Excel Document
PPS	Microsoft Power Point Slide
ODT	Open Office Writer Document
ODS	Open Office Calc Document
SWX	StarOffice Document
CVS	Comma Separated Value
MB	Megabyte
DOCX	Microsoft Word Document

GWT	Google Web Toolkit
API	Application Programming Interface
JRE	Java Runtime Environment
SWT	Standard Widget Toolkit
PNG	Portable Network Graphics
GIF	Graphics Interchange Format
RPC	Remote Procedure Calls
JSNI	JavaScript Native Interface
IDE	Integrated Development Enviroment

SUMÁRIO

INTRODUÇÃO

O processador de texto é um aplicativo que serve para escrever e editar qualquer tipo de texto, desde um comunicado, uma carta para um ente querido ou até um currículo para procurar emprego (OLIVEIRA, 2003). Através da utilização dos recursos computacionais é possível realizar tarefas que, provavelmente, seriam difíceis ou mesmo impossíveis pelo com um aplicativo simples de edição de textos.

A principal vantagem de sua utilização é, através de recursos disponíveis de produtividade, como por exemplo: correção ortográfica e sinônima, poder facilmente modificar, apagar, corrigir, acrescentar gráficos ou tabelas, em uma linha ou parágrafo sem a necessidade de reescrever todo ou parte do documento utilizando menus intuitivos e demonstrando passo-a-passo a formatação do texto.

Já faz algum tempo, e seguindo a atual convergência tecnológica, que os aplicativos têm migrado para Web tornando-se uma plataforma de serviços, com o objetivo de dar onipresença e mobilidade a seus usuários, onde os aplicativos funcionam via Web e não somente instalados em computadores, sendo atualizados quando necessário, de forma constante, linear e independente da intervenção do usuário utilizador, em oposição ao que acontece com aplicativos tradicionais (FERNÁNDEZ, 2007).

Os Webmails, por exemplo, vêm substituindo os clientes de e-mail tradicionais por permitirem a seus usuários manipular suas correspondências a partir de qualquer computador conectado a Internet. Além de proverem interfaces intuitivas e semelhantes, possibilitam que as mensagens fiquem armazenadas e centralizadas no servidor, proporcionando acesso ao conteúdo independente da localidade e horário através da mobilidade e onipresença oferecidas pela Web.

No cenário das ferramentas para edição de documentos na atualidade, podemos encontrar duas classes distintas de processadores de texto: os convencionais como o Word, que é o processador de texto mais utilizado na atualidade, OpenOffice, Google Docs e os científicos, como o Latex que foi desenvolvido com base no Tex (SANTOS, 2000).

Os processadores de texto convencionais são mais conhecidos e utilizados pelos usuários em geral. Eles têm como padrão seguir a metodologia de *What You See Is What You Get* (WYSIWYG), traduzindo para o português "O que você vê é o que você tem". Em outras palavras, estes editores mostram imediatamente na tela do computador como o texto, que está sendo digitado pelo usuário, ficará posteriormente, impresso ou salvo em outro formato.

Seguindo a atual tendência tecnológica, os processadores de texto convencionais estão migrando para o ambiente Web, proporcionando onipresença, mobilidade e dinamismo. Com o objetivo de continuar seguindo a metodologia WYSIWYG e acrescentando novas funcionalidades, como o compartilhamento instantâneo, para editar ou visualizar, documentos, recurso encontrado no Google Docs.

Os processadores de texto convencionais têm migrado através da utilização das tecnologias envolvidas pelo conceito de WEB 2.0, proporcionando aos desenvolvedores criarem aplicações semelhantes aos tradicionais do mercado. Entre as principais tecnologias empregadas podemos destacar o AJAX, RSS e Web Services (BERLITZ, 2007).

Os processadores de texto científicos são mais conhecidos e utilizados na academia. O principal representante desta classe de processadores de texto é o Latex (HEFFERON, 2002). O Latex é um pacote para a preparação de textos de alta qualidade (SANTOS, 2000). Ele foi desenvolvido por Leslie Lamport em 1985 a partir do programa Tex, criado por Donald Knuth entre os anos de 1977 e 1989. Ambos continuam sendo atualizados constantemente pelos grupos TeX Users Group e LaTeX Project.

O objetivo do Latex é fornecer uma linguagem de formatação para o desenvolvimento de textos, científicos ou não, utilizando comandos específicos para informar como deve ser sua formatação, mas não a realizando no mesmo momento do desenvolvimento. O texto e estes comandos de formatação são posteriormente compilados para que a formatação final seja realizada.

Nos processadores de texto convencionais o usuário configura, em tempo real, como deve ser a formatação do documento, mas em Latex esse processo depende de etapas separadas, descritas abaixo (HEFFERON, 2002):

1 Tem-se o arquivo contendo o desenvolvimento do texto juntamente com comandos específicos que informam a formatação a ser realizada (*input file*);

2 Utiliza-se a macro Latex, podendo ser especificada outra, para aumentar os recursos de formatação além dos oferecidos pelo Tex (macro *package*);

3 Obtem-se o arquivo preparado (salvo) para ser executado contendo o desenvolvimento do texto e comandos de formatação (tex);

4 Emprega-se o arquivo contendo as definições de formatação especifica (*output driver*) que procederá a interpretação dos comandos Latex para a formatação final. O arquivo de formatação específico pode ser alterado a qualquer momento de acordo com o padrão de formatação esperado, por exemplo: ABNT, PRODANOV, sem alterar o arquivo contendo o desenvolvimento do texto.

5 Finalmente obtemos o arquivo final (*output file*).

Representação gráfica das etapas envolvidas e descritas anteriormente do processo de formatação de documentos Latex:

Figura 1 – Representação gráfica do processo de formatação de um arquivo Latex.
Fonte: http://www.ctan.org/documents/whyTeX/whyTeX.pdf

Pode-se verificar, através das etapas envolvidas no processo para obtenção do resultado final que é complexo, se comparado com os processadores de texto convencionais. Em contrapartida o processo para formatação do estilo, conforme etapa quatro, é simples e separada do desenvolvimento intelectual do texto, proporcionando abstração de ambas as partes.

Utilizando-se o Latex como processador de texto têm-se como vantagens (OETIKER et al., 2007):

- Uma formatação de melhor qualidade do texto para impressão;
- Fórmulas matemáticas são facilmente criadas e formatadas por comandos específicos, tendo uma grande abrangência de simbologias e métodos matemáticos que não são encontrados nos processadores de texto convencionais;
- Os processadores de texto desenvolvidos para a utilização do Latex, na sua maioria, são mais leves e rápidos que os processadores de texto convencionais, consumindo assim menos recursos do computador;
- Para desenvolver textos utilizando a metodologia Latex não é necessário ter um processador de texto. É possível utilizar o bloco de notas para tal finalidade, pois sua formatação é processada à parte, a partir da sua metalinguagem;
- Textos desenvolvidos em Latex são universais, seu arquivo gerado é independente de sistema operacional ou de processador de texto especifico. Trata-se de um arquivo contendo, além do texto em si, metalinguagem de como é sua formatação, que é processada à parte.

O Latex também possui desvantagens:

- Necessidade de aprendizagem de sua linguagem de formatação;
- Não segue a metodologia WYSIWYG, dificultando sua utilização;
- Necessidade de trabalhar com o arquivo de definição da formatação e do arquivo com o texto propriamente dito.

No cenário atual das ferramentas de processadores de texto científicos encontramos diversas implementações para o Latex, tanto pagas como gratuitas, que fornecem uma experiência mais próxima possível dos processadores de texto convencionais.

A maioria das ferramentas são desenvolvidas com o objetivo de serem instaladas e utilizadas em computadores pessoais, sendo que pouquíssimas estão disponíveis e funcionais para a Web, causando uma carência em contrapartida com as ferramentas de processadores de texto convencionais para o mesmo ambiente.

Entre as ferramentas encontradas, na sua maioria, são equivalentes aos editores de texto comuns, disponibilizando poucos ou nenhum recursos como os encontrados nos seus equivalentes, para computador pessoal, como por exemplo: menus intuitivos, assistentes para composição de fórmulas matemáticas, quadros ou tabelas. Sendo assim, são simples e poucos

funcionais, não proporcionando ao usuário uma experiência e ambiente ideal para desenvolvimento produtivo de textos.

Este trabalho está organizado em dois capítulos: no primeiro capítulo há uma análise sobre os processadores de texto para a Web, tanto os convencionais quanto os voltados para o Latex e o segundo, uma abordagem sobre as tecnologias que serão empregadas para o desenvolvimento da ferramenta Latex para a Web. Ao final, seguem as considerações finais analisando o que foi realizado até o presente momento.

1 ANÁLISE DOS PROCESSADORES DE TEXTO PARA WEB

Este capítulo está organizado da seguinte maneira, nas seções subseqüentes são apresentadas algumas análises dos processadores de texto para Web, tanto os convencionais quanto Latex, demonstrando suas características em especial e comuns em relação a sua proposta de prover uma ferramenta de edição de textos.

1.1 Google Docs

O Google Docs é um editor de texto convencional com foco em proporcionar a seus usuários onipresença e acessibilidade mundialmente pela Web através do endereço http://docs.google.com e focado em oferecer os principais recursos para edição de documentos.

Quando é realizado *login* no Google Docs, tem-se uma tela (interface) inicial demonstrando um gerenciador do ambiente, simples e objetivo, onde contêm as principais funções necessárias para utilização do sistema.

Na parte superior do gerenciador do ambiente há um campo para pesquisar documentos, tanto de textos como planilhas de cálculo ou apresentações de slides, e ao lado existem dois *links*, um contendo opções avançadas de pesquisa e outro que salva as que já foram realizadas conforme destacado na figura 2.

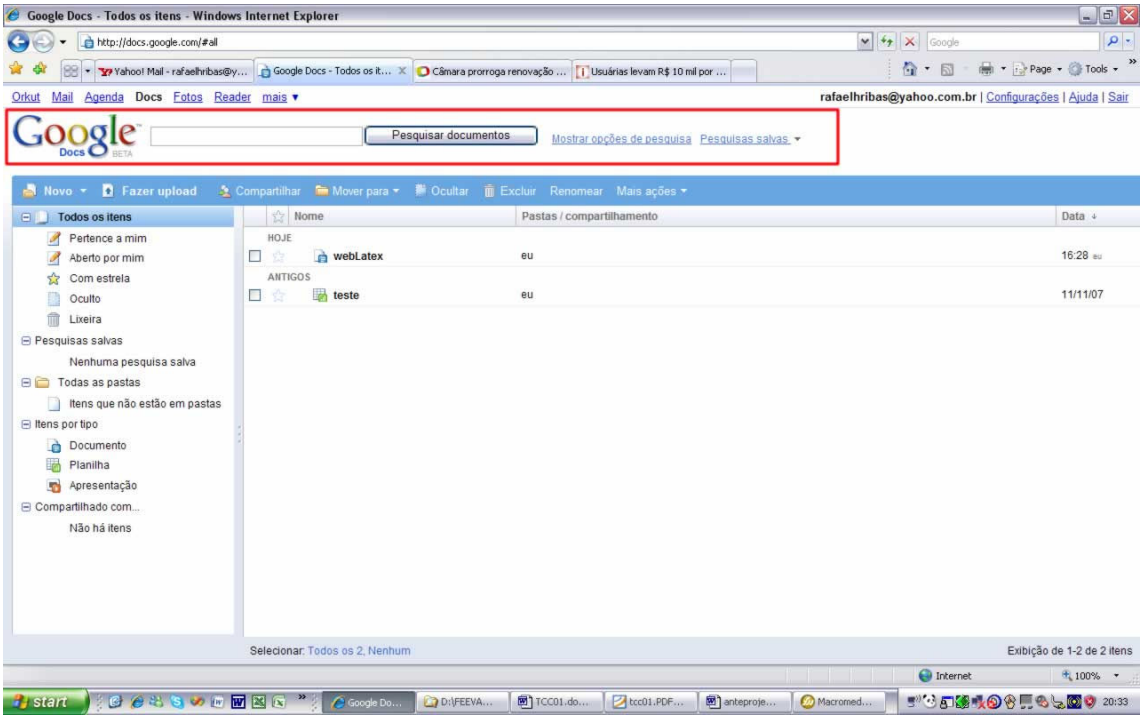

Figura 2 – Tela do gerenciador do ambiente com destaque para pesquisa de documentos.
Fonte: http://docs.google.com

A pesquisa de documentos, tanto simples quanto avançada, utiliza a *engine* do buscador do Google, oferecendo aos usuários todo o poder de indexação e processamento para procurar algum documento em especifico como se estivesse realizando uma pesquisa normal na Web.

Seguindo a tendência normal da maioria dos processadores de texto, este menu tem como finalidade principal, prover as principais ações para o gerenciamento dos documentos de maneira intuitiva e simplificada pela interface proporcionando ao usuário um maior conforto de usabilidade e adaptação. Abaixo se encontra uma descrição dos itens existentes no menu principal bem como suas funções:

- Criar novos documentos, planilhas, apresentações e pastas através da opção "Novo";
- Fazer o upload de arquivos, pelo recurso "Fazer upload", nos formatos: Microsoft Word (.doc), PowerPoint (.ppt, .pps), Excel (.xls), OpenDocument (.odt, .ods), StarOffice (.sxw), Rich Text (.rft), HTML (.html, .htm), texto simples (.txt) e até valores separados por vírgula (.csv). Existem limites de tamanhos máximos

permitidos de acordo com o tipo de documento: caso texto 500KB, apresentação 10MB quando enviado pelo usuário, 2MB quando proveniente da Web e 500KB quando de e-mail, planilha é 1MB;

- Recurso para compartilhamento de arquivos para adicionar colaboradores ou leitores a algum documento específico pelo recurso “Compartilhar”;
- Mover o arquivo para outra pasta especifica através da opção “Mover para”;
- Ocultar arquivos da listagem de arquivos pela opção “Ocultar”;
- Excluir arquivos da listagem de arquivos pela opção “Excluir”;
- Renomear somente um arquivo por fez através do recurso “Renomear”;
- Através da opção “Mais ações” tem-se como opção gerenciar compartilhamento, publicar, realizar revisões no arquivo ou salvar como HTML, OpenDocument, PDF, Microsoft Word e texto simples, visualizar ou alterar o proprietário.

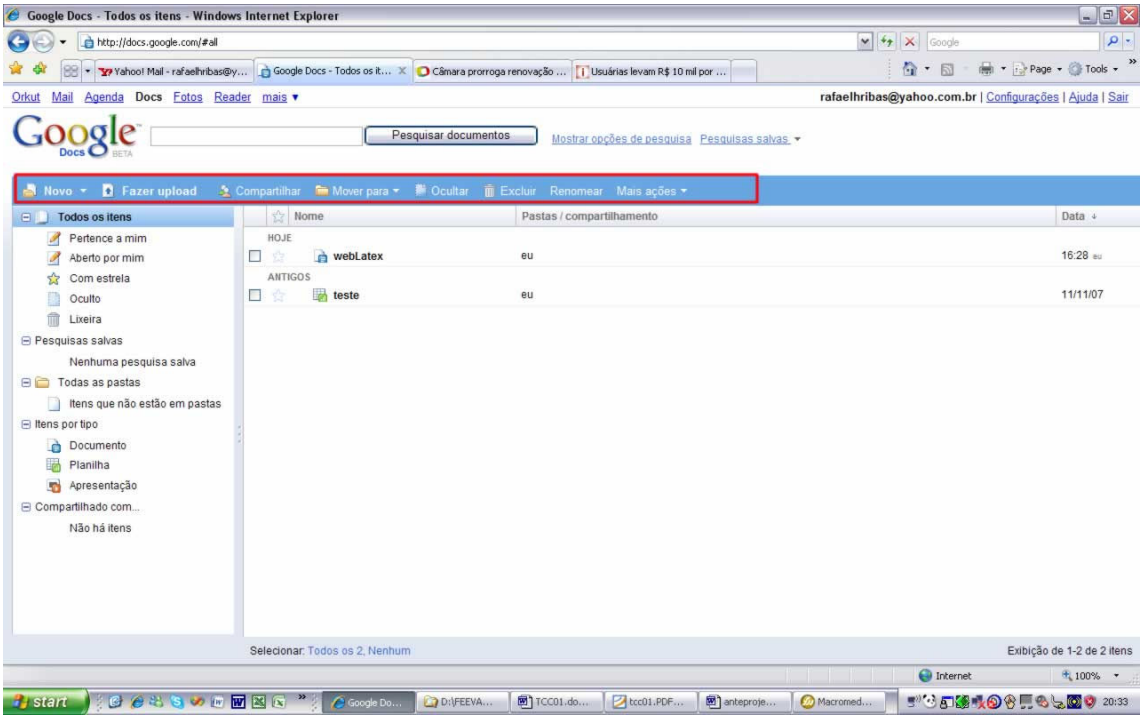

Figura 3 – Tela do gerenciador do ambiente com destaque para o menu principal.
Fonte: http://docs.google.com

Na parte esquerda do gerenciador do ambiente e abaixo do menu principal, temos um recurso útil de visualização em modo de árvore (*tree view)* possibilitando a organização dos

documentos por categorias ou pastas diferentes, na qual se tem a possibilidade de criar novas pastas. É possível arrastar com o mouse qualquer documento para uma pasta existente, ou seja, utilizando-se dos conceitos e tecnologias proporcionados pela Web 2.0 criou-se um ambiente de fácil manipulação e interação com o usuário.

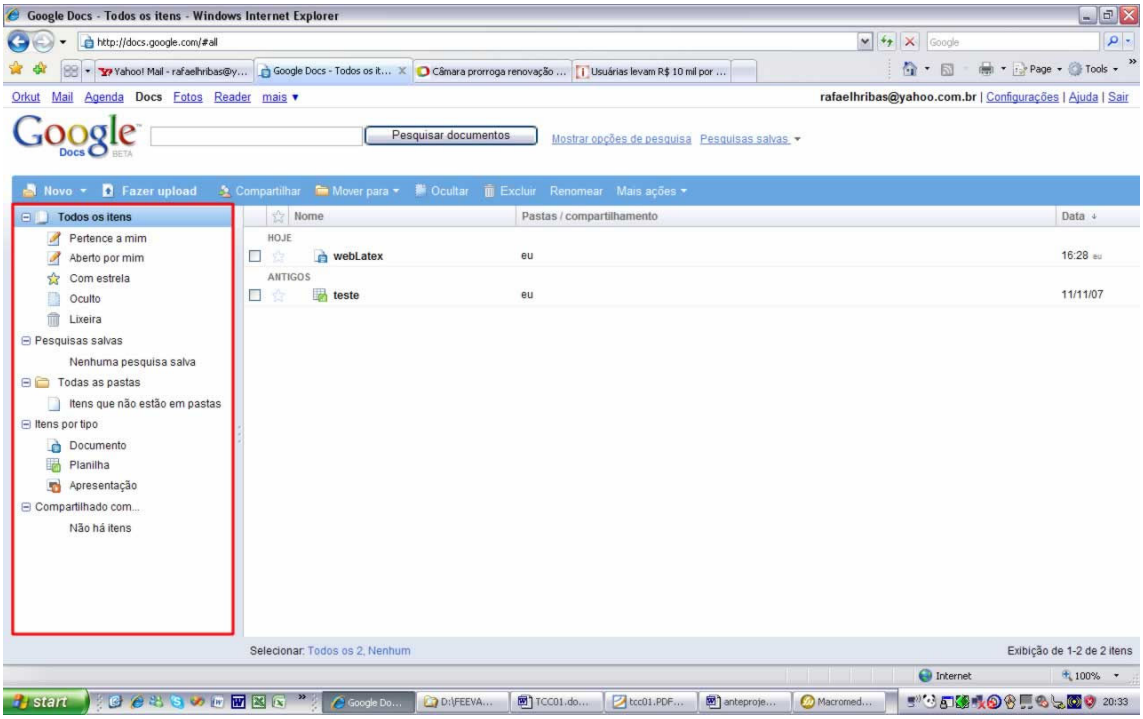

Figura 4 – Tela do gerenciador do ambiente com destaque para o *tree view*.
Fonte: http://docs.google.com

Na parte central do gerenciador do ambiente temos a listagem dos documentos contendo os dados de favoritos, nome do documento, pastas/compartilhamento e a data. Analisando esta parte da interface podemos ver que o Google reutiliza conceitos já assimilados e difundidos em outros serviços, como por exemplo: favoritos que é encontrado no Gmail, aproveitando o conhecimento adquirido pelo usuário, a fim de tornar sua experiência o mais agradável possível e intuitiva diminuindo a curva de aprendizagem do sistema utilizado.

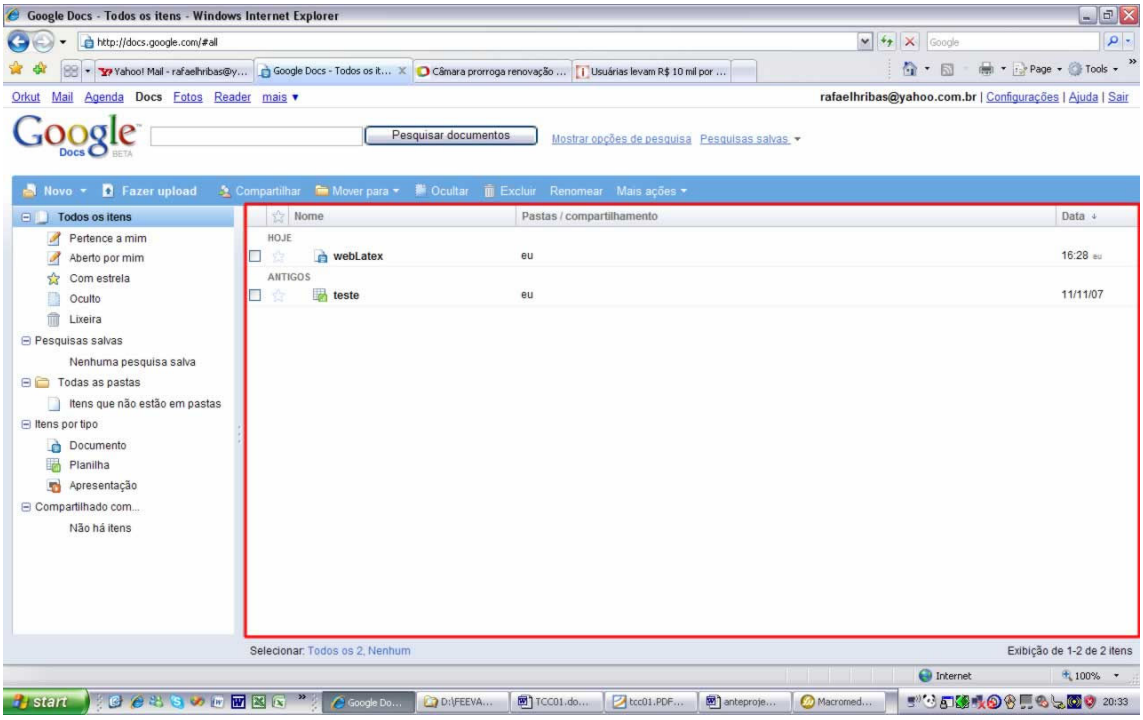

Figura 5 – Tela do gerenciador do ambiente com destaque para a listagem de documentos.
Fonte: http://docs.google.com

O editor de documentos do Google Docs segue o padrão da maioria de suas ferramentas, contendo um visual limpo e objetivo. São apresentados sete menus de opções (*menu bar*) contendo os mesmos recursos ou semelhantes ao encontrados nos processadores de texto tradicionais. E ainda oferece o tradicional menu de atalhos visuais utilizando-se de ícones intuitivos, iniciais de letras, por exemplo: B para negrito ou apenas uma palavra breve para realizar uma funcionalidade ou formatação do texto.

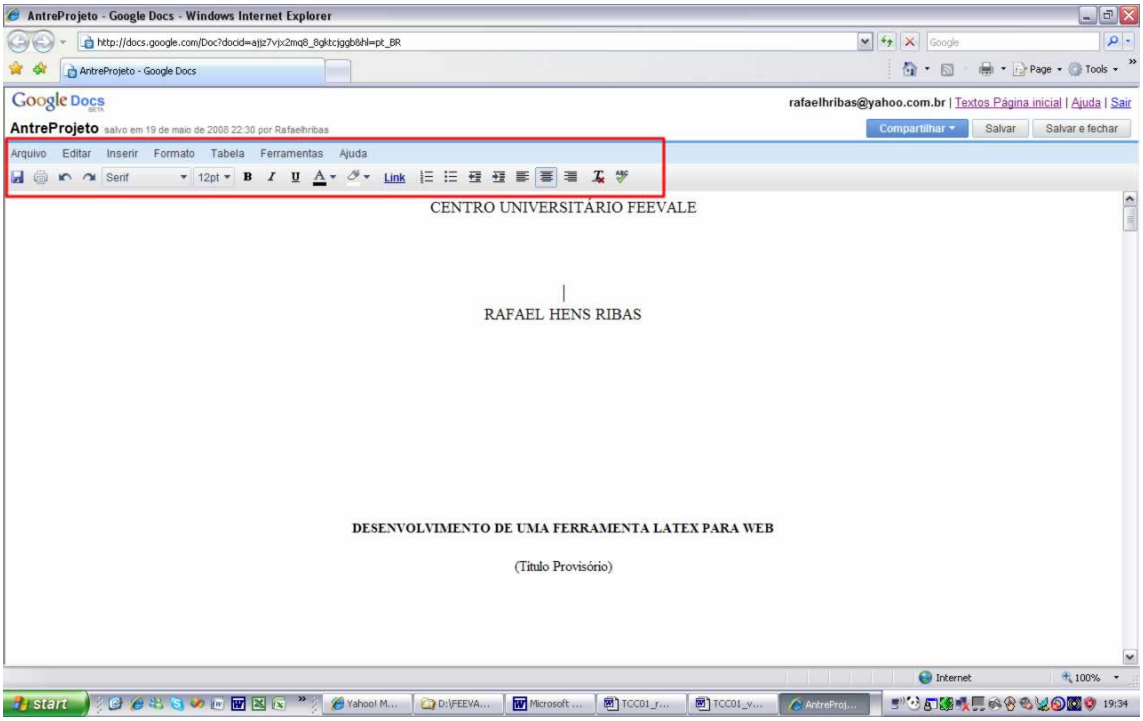

Figura 6 – Tela do editor de documentos com destaque para os menus de ferramentas.
Fonte: http://docs.google.com

Entre as ferramentas encontradas e mencionadas anteriormente, no Google Docs, a maioria corresponde à ação que seria realizada por um processador de textos convencionais. Porém existem outras específicas, e interessantes, dele próprio.

No menu "Arquivo" encontramos os recursos próprios de "Fazer download como...", que se pode ser escolhido entre os formatos de PDF, Word, HTML (zip), RTF, Texto e OpenOffice. Este recurso é bastante interessante e possibilita um dinamismo à ferramenta proporcionando ao usuário exportar de modo fácil e rápido para um dos formatos sem erros ou demais problemas. As opções de "Imprimir como página da web..." e "Visualizar como página da web..." realiza a exportação do documento para o padrão HTML através de um *pop-up*, sendo que a diferença entre os dois recursos é que o primeiro realiza a chamada ao sistema de impressão do *desktop*.

No menu "Editar" encontramos os recursos específicos de "Editar HTML" onde o texto do documento é mostrado contendo a metalinguagem de marcação do padrão HTML para editar alguma *tag* diretamente no seu fonte. A opção de "Editar CSS" abre uma caixa de interação onde se pode editar algum padrão de CSS a fim de aplicar no seu documento. Essas

opções são úteis caso seja necessário editar alguma propriedade do documento, que não tenha uma ferramenta do próprio Google Docs, porém requer que o usuário tenha esses conhecimentos e habilidade prévia.

Ao utilizar-se a opção de "Fazer Upload" do gerenciador do ambiente para testar o procedimento de enviar um documento a fim de editá-lo, surgem algumas limitações quanto ao tamanho do arquivo relativo e ao tipo de documento. Este procedimento faz com que não seja possível editar documentos, por exemplo: do Microsoft Word, que tenham um tamanho superior de 500 KB, gerando um empecilho ao usuário que utiliza ou desenvolve grandes textos ou recursos que contribuíam para o tamanho final.

A importação ocorreu normalmente, quando não ultrapassou o limite, porém nem toda a formatação do documento foi importada corretamente sendo necessário revisar e arrumar os itens que se encontraram nessa situação. Este fato, realmente, não se pode definir como um grande problema, mas um empecilho e incredibilidade quanto a sua idéia de proporcionar uma experiência o mais parecida ao de um processador de texto convencional para *desktop*, gerando um retrabalho ao usuário; ter que revisar e formatar o documento novamente.

A ferramenta desenvolvida pelo Google cumpre com o seu papel fundamental de prover uma solução, disponível e acessível pela Web, contendo os recursos mais utilizados pelos processadores de texto convencionais buscando sempre a simplicidade que é a sua característica.

1.2 ThinkFree

O *ThinkFree* é um editor de texto convencional, focado em proporcionar a seus usuários uma experiência o mais rica possível ao um processador de texto *desktop*. Com foco em ser onipresente e acessível, principalmente, pela Web pelo endereço http://www.thinkfreedocs.com/.

Ao realizar-se *login* no *ThinkFree*, e acessando a opção de *"My Office*", obtem-se uma tela demonstrando um gerenciador de ambiente, simples e objetivo, contendo as

principais funcionalidades para utilização do sistema. Este recurso é encontrado nos seus equivalentes, por exemplo: Google Docs, e demonstra ser uma tendência eficiente e eficaz em proporcionar ao usuário um maior domínio, dinâmica e facilidade de operação e gerenciamento.

O gerenciador de ambiente, diferente de seus equivalentes, possui um acesso simples com maior intuitividade e facilidade às opções principais, como: "Novo documento", "Nova Planilha" e "Nova apresentação", eliminando que o usuário necessite acessar um menu, que contenha uma listagem especifica de opções, para somente então escolher uma das opções de documentos.

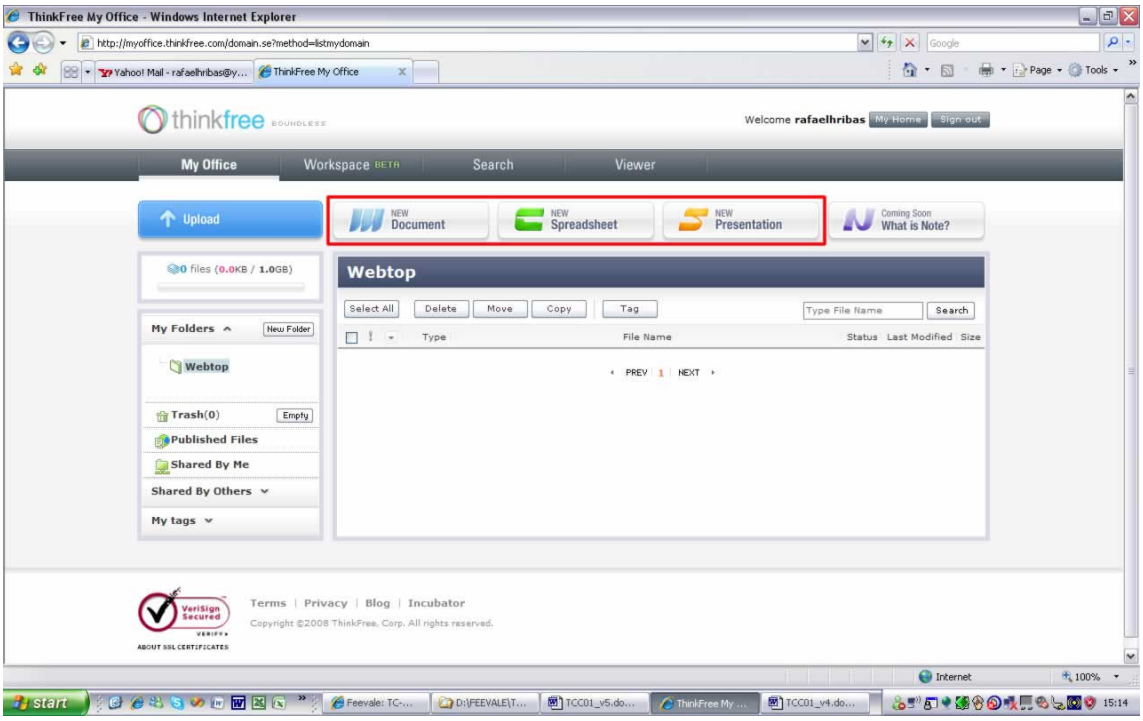

Figura 7 – Tela do gerenciador do ambiente com destaque para as opções principais.
Fonte: http://thinkfreedocs.com

Na parte esquerda do gerenciador do ambiente temos um recurso útil, e bastante utilizado atualmente, de visualização por *stackpanel*, contendo dentro o *treeview* de documentos ou pastas. Estes recursos oferecem, ao usuário, uma prévia organização, porém permitindo que cada individuo faça a sua, particularmente, dentro dessa configuração, por exemplo: No *stackpanel* denominado "*My Folders*" podem serem criadas pastas de acordo com cada tipo de documento estabelecido pelo usuário.

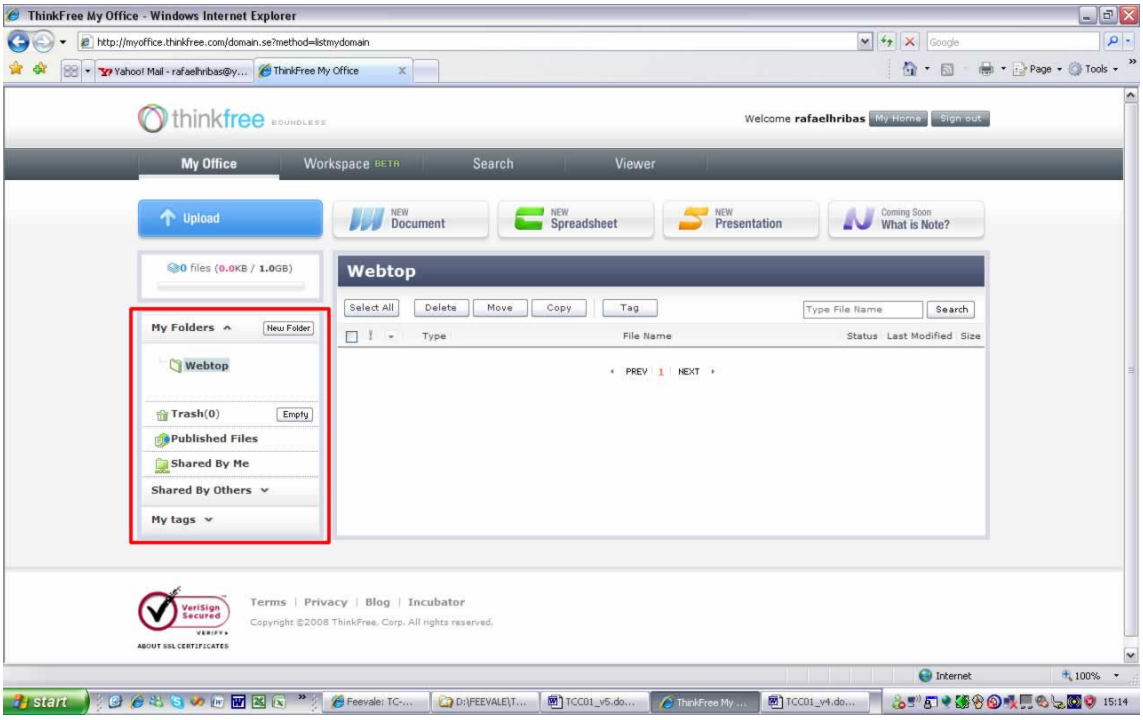

Figura 8 – Tela do gerenciador do ambiente com destaque para o *stackpanel*.
Fonte: http://thinkfreedocs.com

Na parte central do gerenciador do ambiente temos uma listagem dos documentos contendo os dados de "*Flag*", "tipo", "nome do documento", "*Status*", "data da última modificação" e "tamanho". Obtem-se também um menu, que é acionado por um botão contendo mais opções, por exemplo: deletar, alterar, publicar, entre outros. O gerenciador não segue a tendência de utilizar recursos de Web 2.0 pois segue o modelo tradicional de precisar selecionar um ou mais documentos, através do *checkbox* para, posteriormente, realizar uma ação, como por exemplo: mover de pasta, algum documento.

Juntamente com a listagem de documentos há um campo para pesquisar documentos de qualquer tipo de padrão suportado pelo *ThinkFree*. A pesquisa é bastante simples, oferecendo apenas a opção de informar o nome ou parte dele para realizar a procura, retornando o resultado na mesma listagem. Ao ser utilizado esse recurso comportou-se de maneira normal, como qualquer outro tipo de pesquisa, mas poderia ter algumas opções avançadas como encontrados no Google Docs, como: pesquisar por tipo de documento, a fim de aumentar a performance e filtrando as possibilidades de arquivos a serem retornados.

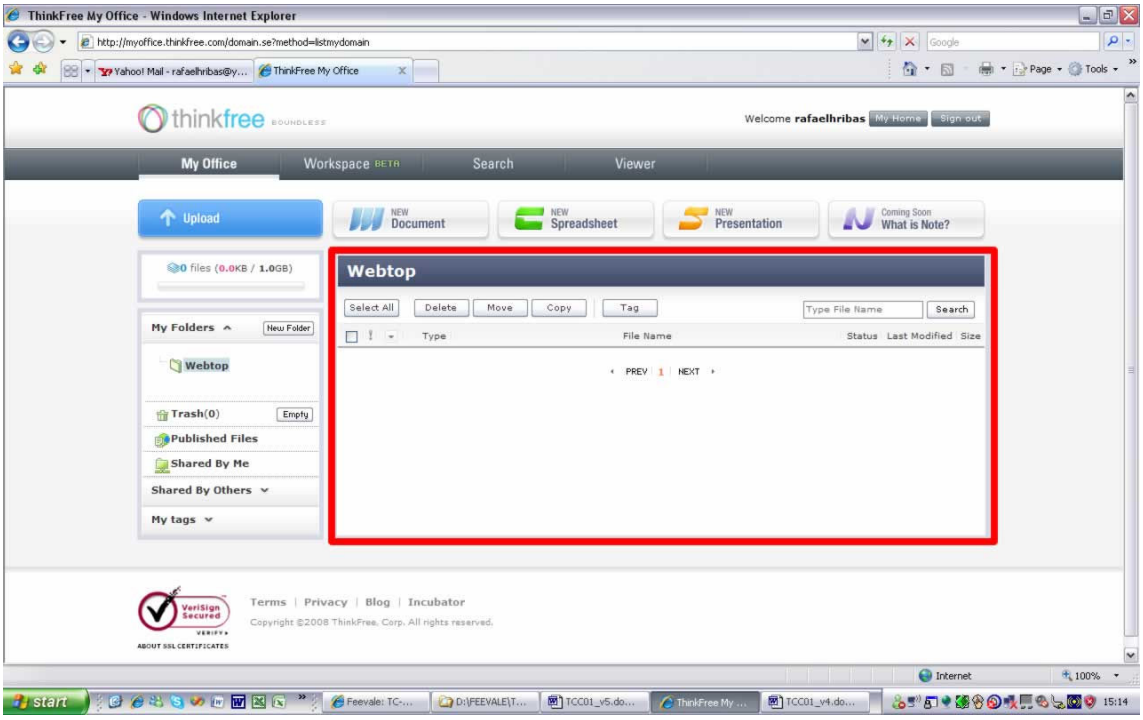

Figura 9 – Tela do gerenciador do ambiente com destaque para a listagem de documento.
Fonte: http://thinkfreedocs.com

Ao realizar o acesso ao editor de documentos é feito uma verificação pelo sistema, onde recebe-se a mensagem de que a instalação não foi propriamente finalizada, pois não tem o *Office 3* instalado no computador. Este recurso permite que se possa trabalhar com o editor *offline* sem precisar estar todo tempo na Web, apenas para fazer a sincronização com o sistema, por exemplo: salvar algum documento.

O editor, com ou sem documento propriamente, demorou para ser carregado devido à utilização de *applet* Java, para proporcionar uma experiência mais rica e semelhante aos convencionais, porém somente no primeiro acesso ao editor isso ocorre, pois nos acessos posteriores a aplicação já estará carregada na JVM do computador local.

A interface gráfica segue o padrão da maioria das ferramentas, porém contendo um visual bastante rico e semelhante aos processadores de texto convencionais como Microsoft Word. São apresentados oito menus principais de opções (*menu bar*), além de oferecer o tradicional menu de atalhos visuais de ícones intuitivos. Essa interface proporciona ao usuário uma melhor experiência e aproveitamento do sistema, pois utiliza o conhecimento prévio adquirido a fim de tornar sua experiência o mais agradável e intuitiva possível diminuindo a sua curva de aprendizagem.

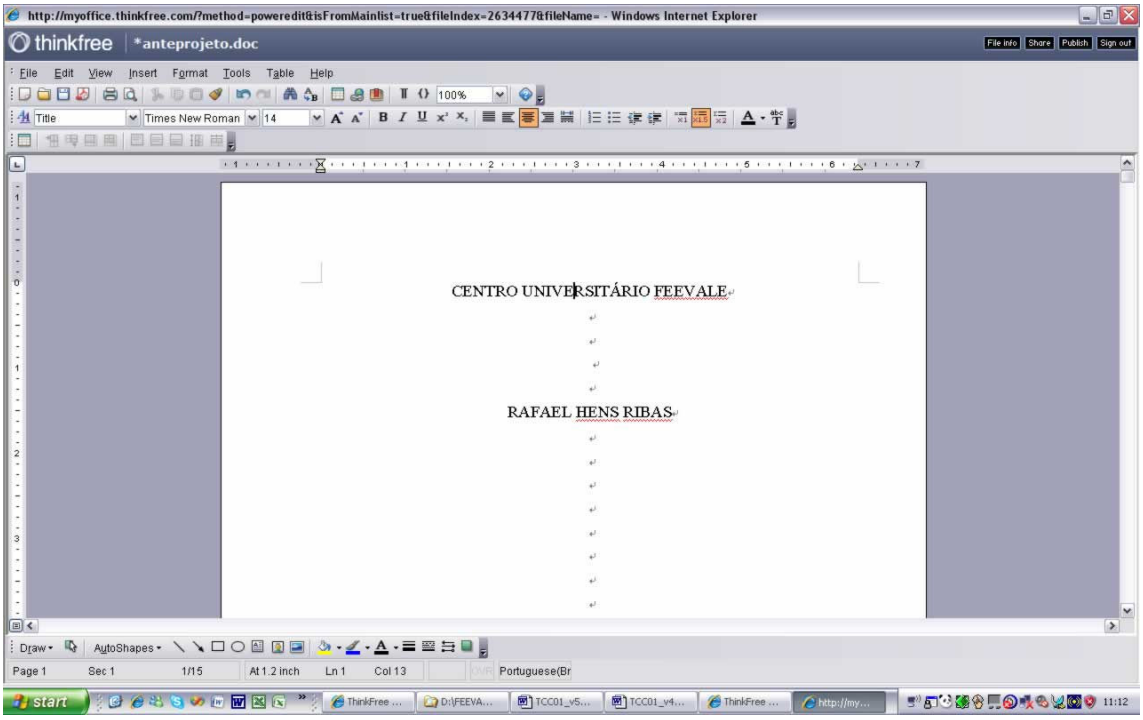

Figura 10 – Tela do gerenciador do ambiente com destaque para o editor.
Fonte: http://thinkfreedocs.com

As ferramentas encontrada no editor do *ThinkFree* suportam e fazem menção a praticamente todas encontradas nos seus equivalentes para *desktop*, contendo apenas algumas limitações da sua abrangência, porém cumprindo com suas funcionalidades.

Há outro recurso bastante interessante para salvar o documento encontrado no menu *"File"*, além dos formatos de DOC, RTF, TXT, HTML, tem-se como escolha salvar como um arquivo PDF, XML e DOCX.

Contamos com os recursos de "*Share*" e "*Publish*", respectivamente, encontrados no canto superior direito do editor. Onde se pode compartilhar o documento com outro usuário do sistema. Este recurso é encontrado nos seus equivalentes para Web e permite uma escrita colaborativa ou podendo publicar na Web o documento, disponibilizando para que qualquer pessoa o acesse através da URL fornecida pelo próprio sistema. Ao realizar sua publicação na Web, o documento é convertido automaticamente para uma página no formato HTML, ou seja, ao fazer esse teste algumas formatações não se mantiveram como no documento original.

Ao utilizar-se da opção de "*Upload*" do gerenciador do ambiente a fim de testar o procedimento de enviar um documento para editá-lo no sistema, podemos enviar até cinco

documentos ao mesmo tempo para a mesma pasta. Este recurso é um facilitador ao usuário, pois em apenas um procedimento possibilita o envio de múltiplos documentos, não sendo necessário enviar separadamente cada um. Não existe uma limitação quanto ao tamanho do arquivo a ser enviado pelo "*Upload*", porém existe a limitação quanto ao espaço disponível para utilização no sistema que atualmente é de 1.0 GB.

A importação do arquivo foi realizada normalmente e quanto à formatação contida no documento foi mantida igualmente sem que o usuário tenha que revisar e arrumar algum item. Este é o ideal de toda ferramenta, pois cumpre com o seu propósito de proporcionar ao máximo uma experiência o quanto mais semelhante ao de um processador de texto convencional para *desktop* e proporcionando sua portabilidade entre ambos.

A ferramenta desenvolvida pela *ThinkFree Corp* cumpre com o seu papel fundamental de prover uma solução, tanto disponível, quanto acessível pela Web como também possibilita ao usuário utilizar o editor *offline*, através de um *plugin* próprio, contendo os recursos encontrados e utilizados quanto por seguir a mesma aparência de interface gráfica dos processadores de texto convencionais para *desktop*, proporcionando uma experiência o quanto mais semelhante.

1.3 Zoho Writer

O *Zoho Writer* é um editor de texto convencional com foco de ser utilizado, principalmente, pela Web provendo uma solução onipresente e acessível mundialmente por seus usuários de qualquer lugar do mundo através do endereço http://writer.zoho.com/.

Ao realizar *login* no *Zoho Writer* acessa-se diretamente o editor de documentos, onde contém, juntamente nessa mesma tela as principais funcionalidades para a utilização do sistema. Este não segue o padrão dos seus equivalentes em ter um gerenciador de ambiente para, somente então, acessar o editor e suas funcionalidades realizando um acesso rápido e mantendo ainda mais o foco no item principal que é a edição de documentos.

Na parte esquerda do ambiente há as funcionalidades de "novo", "importar" e "deletar" algum documento. Logo abaixo, existe um menu em forma de *stackpanel*. Este

recurso demonstra ser uma tendência em uso pois seus benefícios e justificativas já foram comentados na seção anterior, contendo outras funcionalidades do ambiente. Porém, o usuário somente pode utilizar a organização e recursos já encontrados dentro do *stackpanel*, não permitindo que cada individuo, particularmente, faça a sua própria, como exemplo: no *stackpanel* denominado "*My Docs*" não tem a possibilidade de criar pastas de acordo com o tipo de assunto do documento.

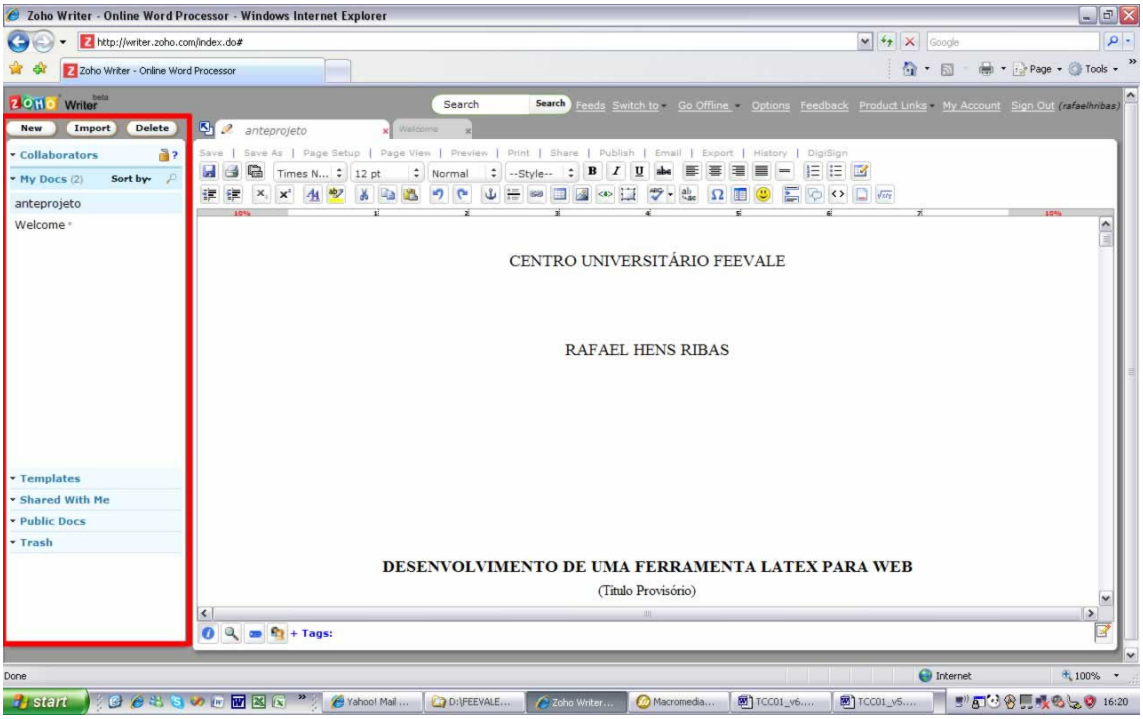

Figura 11 – Tela do ambiente com destaque para o *stackpanel*.
Fonte: http://writer.zoho.com

A listagem de documentos é feita somente dentro do *stackpanel* denominado "*My Docs*" contendo os recursos de ordenar por nome, data de criação ou alteração e pesquisa de documentos, que também pode ser acessado na parte superior central do ambiente.

A pesquisa de documentos possui apenas um campo simples para informar o nome ou parte dele, para realizar a procura que suporta qualquer tipo de padrão permitido pelo *Zoho Writer*. Porém, esse recurso, quando testado, não obteve o resultado satisfatório e esperado, sendo que apenas encontrou o documento, denominado "anteprojeto", quando informado toda a palavra e ainda mostrou como se tivesse encontrado mais de um arquivo com o mesmo nome, quando na verdade, apenas foi enviado um com esse nome. Poderia ter o recurso de

opções avançadas, igualmente ao Google Docs, de poder pesquisar, pelo menos, por tipo de documento com o intuito de aumentar sua performance e filtrar as possibilidades de arquivos a serem encontrados, posteriormente ao realizar as correções descritas anteriormente para que, pelo menos, cumpre satisfatoriamente com o seu papel.

O editor é carregado rapidamente e sempre com um novo documento em branco, já que é possível acessá-lo assim que realiza-se o *login* no sistema. A interface gráfica segue o padrão da maioria das ferramentas contendo um visual bastante rico, não buscando uma semelhança aos processadores de texto convencionais como idealizado pelo *ThinkFree*, mas buscando a sua própria. São apresentados doze menus principais de opções (*menu bar*) além de oferecer o tradicional menu de atalhos visuais de ícones intuitivos. Mesmo que o editor utilize a sua própria identidade visual e diferenciada dos demais mencionados anteriormente, a sua interface também, proporciona ao usuário uma melhor experiência e aproveitamento, utilizando o conhecimento prévio adquirido, mantendo alguns padrões de mercado.

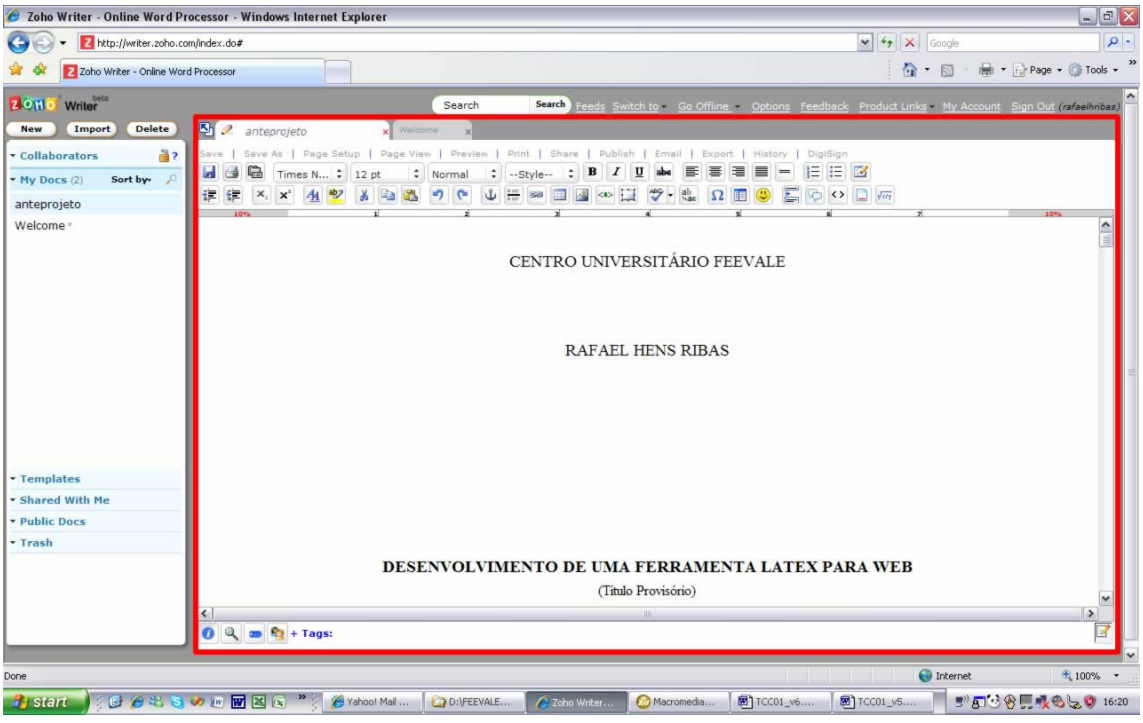

Figura 12 – Tela do ambiente com destaque para o editor.
Fonte: http://writer.zoho.com

As ferramentas encontradas no editor suportam e realizam praticamente todas encontradas nos seus equivalentes, tanto para *desktop* quanto para Web, cumprindo com o seu papel, mas disponibilizando juntamente outras bastantes interessantes.

Podemos salvar o documento, além do padrão do próprio sistema, como um *template* caso futuramente se deseje partir de para um documento pré-estabelecido.

Contamos também com as opções de "*Share*" e "*Publish*", respectivamente encontrados juntamente com o menu de opções (*menu bar*) onde podemos compartilhar o documento com outros usuários ou com um grupo especifico de usuários do sistema. Este recurso é encontrado em todos, até o momento, pois permite uma escrita colaborativa ou podendo publicar o documento na Web, através das opções de "postar em um *blog*", onde temos uma lista de sistemas de *blog* previamente configurados ou adicionar outro apenas informando os campos necessários, disponibilizando para que qualquer pessoa o acesse através da URL fornecida pelo próprio sistema ou utilizando o recurso de "*Doc Roll*" onde é gerado um *scriptlet* para ser inserido em um *blog* ou site da Web.

A opção "*Email*" contém os recursos de "*Email Out*" e "*Email In*", onde, respectivamente, pode-se enviar o documento para outra pessoa apenas informando o endereço de e-mail, que o próprio sistema realiza o envio ou remeter de nosso *desktop* algum documento para, através de um endereço gerado automaticamente para cada usuário. Este recurso mostra-se bastante útil e dinâmico por não precisar o usuário acessar o sistema para realizar o *upload* de algum documento bastando, apenas enviar um e-mail para executar tal tarefa.

O editor, através do recurso "*Export*", pode exportar o documento para os formatos DOC, DOCX, ODF (ODT), PDF, SXW, RTF, Text File, HTLM ou Latex. Como todos abrangem a maioria dos tipos de formatos mais utilizados, porém com um destaque para o formato Latex provendo suporte também aos editores científicos.

A opção de "*History*", diferente dos seus equivalentes, mostra as diferenças ocorridas diretamente no documento, disponibilizando a opção de poder comparar entre as versões. Esta sistemática mostrou-se melhor, pois realiza a marcação diretamente no documento, facilitando a compreensão do que foi alterado, em vez de abrir de uma vez cada versão do mesmo documento.

O recurso de "*DigiSign*" permite ao usuário inserir sua assinatura digital no documento a ser enviado por e-mail. Este recurso foi apenas encontrado nesse sistema e permite que documentos sejam trocados com segurança e sua veracidade testada contra fraudes.

Igualmente encontrado em seus equivalentes, também pode-se trabalhar *offline* nos documentos que estão no sistema através do *plugin* do Google, denominado *Google Gears*, que é responsável em fazer a sincronização dos documentos e do sistema para serem utilizados no *desktop*. Este recurso é bastante útil em permitir que se possa trabalhar sem necessariamente estar *online* na Web apenas para realizar a sincronização dos dados. Porém, o sistema do *Zoho Writer*, de certa forma, está presa a um *plugin* desenvolvido por um terceiro diferentemente ao caso do *ThinkFree* que desenvolveu o seu próprio.

Ao utilizar-se a opção de "*Import*" para testar o procedimento de enviar algum documento a fim de editá-lo, pode-se realizá-lo informando onde buscar o arquivo no *desktop* ou enviando um e-mail para o endereço que cada usuário tem para tal finalidade. Este recurso já foi apresentado anteriormente e demonstra um facilitador ao usuário. Não existe uma limitação quanto ao tamanho do arquivo e nem de quanto espaço destinado a cada usuário.

A importação do arquivo foi realizada normalmente, porém nem toda a formatação foi importada corretamente, sendo necessário revisar e arrumar os itens que se encontraram nessa situação. Este fato, igualmente encontrado no Google Docs, gera ao usuário final um retrabalho desnecessário, não proporcionando totalmente uma experiência o quanto mais semelhante ao de um processador de texto *desktop*.

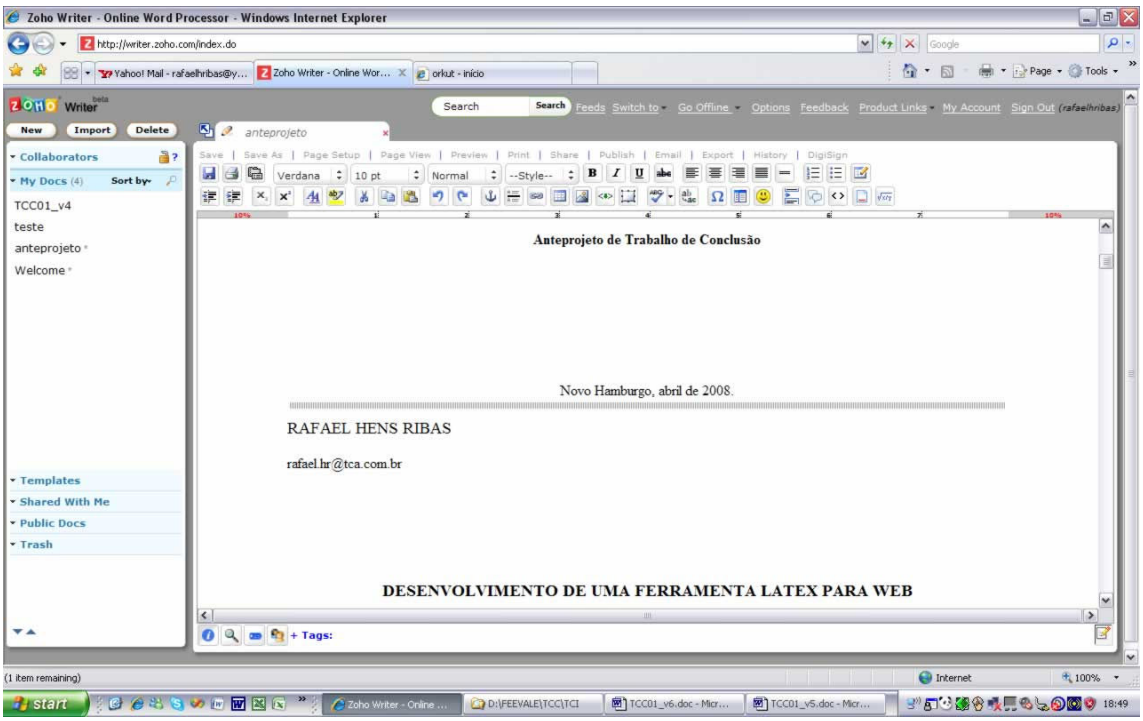

Figura 13 – Tela do ambiente com destaque para o editor.

Fonte: http://writer.zoho.com

A ferramenta desenvolvida pela *AdventNet Inc* cumpre com o papel de prover uma solução, tanto disponível pela Web quanto *offline*, através de um *plugin* do Google, contendo os recursos encontrados e utilizados nos seus equivalentes, tanto para Web quanto para *desktop*, porém com destaque a novas funcionalidades encontradas somente nesse sistema, como exemplo: exportar o documento para o formato Latex, provendo aos seus usuários facilitadores e uma experiência mais dinâmica.

1.4 Online LATEX (FLASH)

Os editores de Latex encontrados disponíveis para utilização na Web são, em maioria, rudimentares, sendo praticamente apenas um campo de texto com a opção de compilar o texto para gerar o arquivo final. Devido a esse fato será apresentado somente a

análise daquele que obteve a maior semelhança com os editores convencionais para a Web e Latex.

Diferentemente dos editores convencionais para Web demonstrados e estudados anteriormente, esse editor não necessita que o usuário realize *login* para utilizá-lo. Assim que se acessa o *site*, a interface gráfica é carregada utilizando a tecnologia Flash, na qual consume algum tempo para completar a ação.

Sua interface é bastante simples, contendo somente o necessário para poder desenvolver o seu texto na ferramenta. Separadas por janelas modais, onde consistem em ter janelas independentes uma das outras, mas encapsuladas na principal, que é o próprio *browse*r.

Na sub janela, denominada "*LaTeX – Symbols*", onde contém os principais símbolos utilizados, como exemplo: letras gregas, hebraicas, e ao seu lado o respectivo comando Latex. Ao clicar sobre algum desses símbolos, o código é copiado para a área de transferência (*clipboard*) do *desktop*, onde o usuário, posteriormente, deve realizar a ação de colar na janela destinada ao desenvolvimento do texto, denominada "*Latex Editor*". Este recurso seria útil se partisse do principio de, ao clicar sobre um símbolo, automaticamente fosse acrescentado o seu respectivo código Latex na posição corrente do cursor no editor, ao contrário do que é feito atualmente, tornando um recurso ineficaz para o usuário.

A sub janela, denominada "*Latex Editor*", é onde se desenvolve o texto, juntamente com os comandos Latex. Este editor contém pouquíssimos recursos, se comparado aos editores Latex para *desktop* ou com os convencionais para Web, sendo praticamente apenas um campo de texto. Entre os recursos encontrados podemos configurar o DPI, formato de saída final do documento, por exemplo: png16m ou jpeg. O melhor recurso encontrado é a opção de poder carregar, previamente, um *template* simples fornecido pelo próprio sistema.

A geração do arquivo final ocorre através do recurso "*start LATEX*" responsável pelo processo de compilação e formatação do texto. O processo ocorreu satisfatoriamente e obteve-se o resultado esperado.

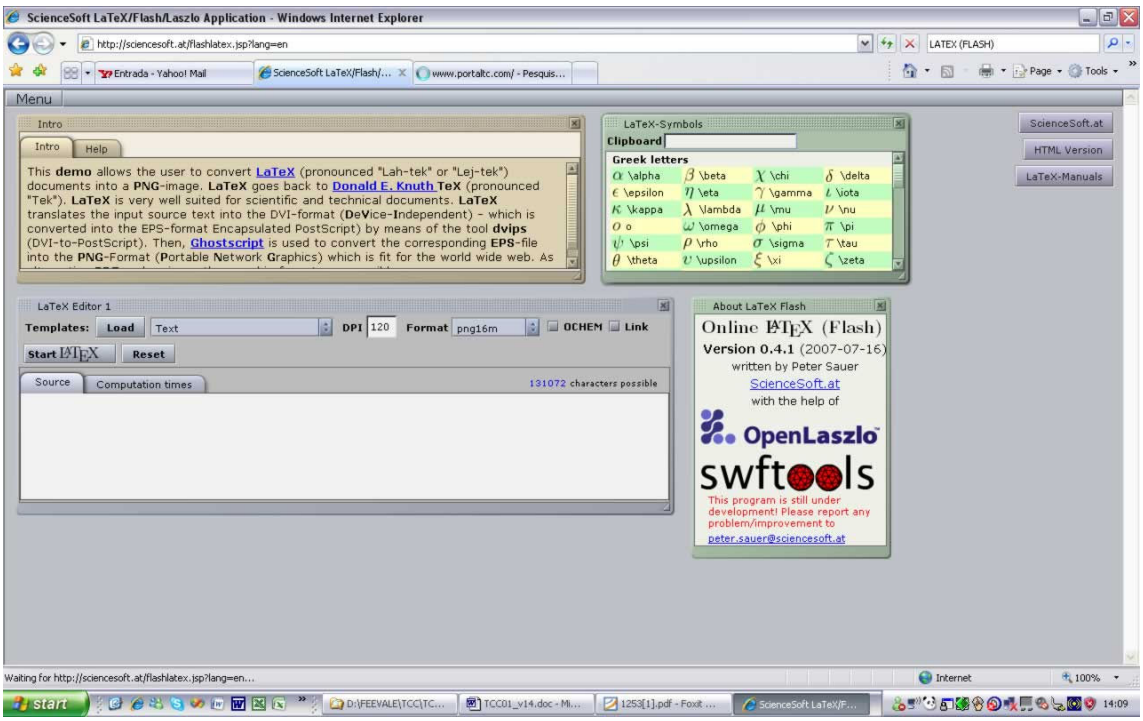

Figura 14 – Tela do ambiente do Latex (Flash).

Fonte: http://sciencesoft.at/flashlatex.jsp?lang=en.

A ferramenta desenvolvida pela *ScienceSoft* não cumpre com o papel de prover uma solução ao usuário. Por não conter os recursos de produtividades mínimas, como encontrados nos editores Latex para *desktop* ou convencionais para a Web, tornando o trabalho árduo e semelhante à utilização de um bloco de notas.

2 ANÁLISE DAS TECNOLOGIAS A SEREM UTILIZADAS

Este capítulo está organizado da seguinte maneira: nas seções subseqüentes são apresentados o conceito e a finalidade do *Google Web Toolkit* para o desenvolvimento de soluções para a Web juntamente com exemplos práticos de utilização.

2.1 Google Web Toolkit

O *Google Web Toolkit* (GWT) é um *framework open source* (API) para o desenvolvimento de aplicativos Web com o conceito de Web 2.0. Permite compilar uma aplicação desenvolvida em Java para AJAX integrado ao HTML, CSS e Javascript (BERLITZ, 2007).

A arquitetura do GWT é separada em quatro componentes principais: um compilador Java-to-JavaScript, um *browser* em modo hospedado, que executa localmente o aplicativo Web para *debug*, biblioteca de classes Web de IU (*widgets*) e duas bibliotecas Java de emulação JRE (GOOGLE, 2008).

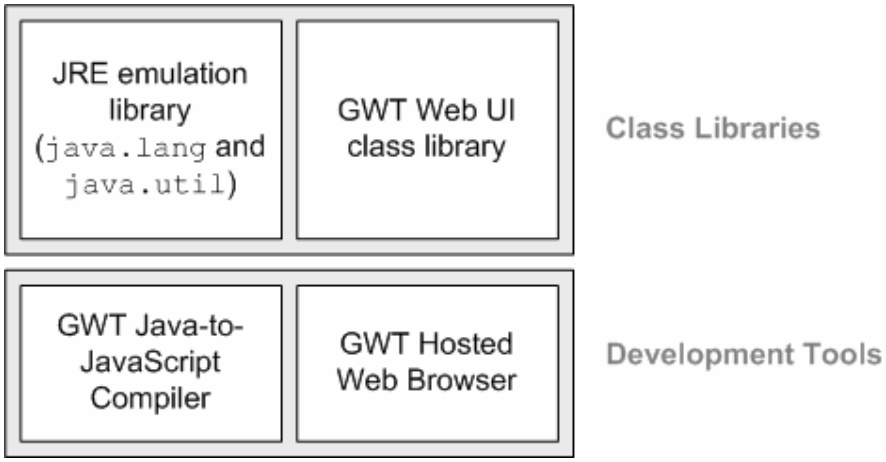

Figura 15 – Representação gráfica da arquitetura do GWT.
Fonte: http://code.google.com/intl/pt-BR/webtoolkit/overview.html

A tarefa de desenvolver um aplicativo Web, usufruindo da tecnologia AJAX, é um processo trabalhoso no ciclo de desenvolvimento. Pois trata-se de ser mais uma camada de desenvolvimento no ciclo do projeto. É um código Javascript complexo que, integrado com outras tecnologias, se comunica com o servidor e atualiza o *browser* do usuário apenas do conteúdo necessário, sem atualizar toda a página.

O compilador do GWT tem o propósito de realizar a compilação do código Java para o Javascript. Compilando o código, funcional e testado, da aplicação local para a Web, pois o mesmo irá funcionar após a compilação. Ele prevê o suporte amplo à linguagem Java, métodos ou classes, e caso alguma não seja, ele emite um erro ao desenvolvedor.

Após a compilação é criado um diretório de saída contendo todos os arquivos convertidos. É necessário realizar o *upload* ao servidor Web e configurar a aplicação a apontar para uma página HTML *default*, normalmente com o mesmo nome do projeto.

Ao contrário do que se possa imaginar, o GWT não aumenta o código da aplicação com lixo de programação, ao contrário, otimiza ao máximo. Além de remover a preocupação, do desenvolvedor, em manter o código compatível entre *browsers*, pois toda compilação feita tem compatibilidade com *Internet Explorer, Firefox, Safari* e *Opera*. Cada elemento da interface gráfica é traduzido para o mesmo do *browser* nativo do usuário, como por exemplo: um botão feito por um *widget* é convertido para a *tag* HTML verdadeira em vez de fazer uma emulação com outro recurso.

O GWT oferece classes para desenvolvimento de UI semelhantes a outros *frameworks*, como por exemplo *Swing* e SWT. Realiza a transformação para o código HTML, dinamicamente, em vez de usar outros tipos de recursos gráficos. A interface gráfica da aplicação é construída através de *widgets*, que contêm *panels* que funcionam, praticamente, da mesma maneira em todos os *browsers*. Com este tipo de solução é possível que se desenvolvam, rapidamente, interfaces gráficas ricas. Sem precisar dedicar um tempo excessivo para tratamento de compatibilidades entre *browsers*.

Para modificar visualmente, algum desses *widgets*, utiliza-se a técnica de CSS, onde cada um tem o seu próprio configurado diretamente pelo GWT. Porém, se necessário, pode-se incluir um estilo de CSS produzido, à parte ou de terceiro, sendo referenciado no HTML. Através desse recurso pode-se facilmente configurar como será o visual da aplicação, mas caso necessário, existe a possibilidade de aproveitar algum CSS já implementado, ou seja, o GWT oferece essa liberdade ao desenvolvedor.

Quadro 1 – Exemplo de utilização CSS através do GWT para formatar um botão.

```
.gwt-Button { font-size: 130% ; font-family:Arial, Helvetica, sans-serif ; font-style:normal }
```

Fonte: http://code.google.com/webtoolkit/overview.html.

Pode-se desenvolver novos *widgets*, e de modo simples, através da classe *composite*, onde normalmente combina-se mais de um componente pronto. Caso seja necessário, é possível criá-los a partir do zero, mas não é uma tarefa simples. Pois é preciso escrever códigos, diretamente em Javascript, sendo considerados de baixo nível pelo GWT. Dificilmente é necessário escrever um novo *widgets* do início porque existem muitos já implementados, além dos que são fornecidos.

Quadro 2 – Exemplo de desenvolvimento de novo *widgets*.

```
public static class OptionalTextBox extends Composite implements
        ClickListener {

    private TextBox textBox = new TextBox();
    private CheckBox checkBox = new CheckBox();

    public OptionalTextBox(String caption) {
       VerticalPanel panel = new VerticalPanel();
       panel.add(checkBox);
       panel.add(textBox);

       checkBox.setText(caption);
       checkBox.setChecked(true);
       checkBox.addClickListener(this);

       initWidget(panel);

       setStyleName("example-OptionalCheckBox");
    }

    public void onClick(Widget sender) {
       if (sender == checkBox) {
          textBox.setEnabled(checkBox.isChecked());
       }
    }

    public void setCaption(String caption) {
       checkBox.setText(caption);
    }

    public String getCaption() {
       return checkBox.getText();
    }
 }

  public void onModuleLoad() {
    OptionalTextBox otb = new OptionalTextBox("Check this to enable me");
    RootPanel.get().add(otb);
 }
```

Fonte: http://code.google.com/webtoolkit/overview.html.

Um conceito introduzido pelo GWT é o recurso de *Image Bundles*, que consiste em gerar uma imagem única, porém criada através de outras pequenas imagens. Esta funcionalidade é para a resolução de que, tipicamente, uma aplicação Web sempre se utiliza de pequenos ícones ou imagens para refinar a parte visual. Porém, sempre que é realizada

alguma requisição HTTP, são enviadas novamente, gerando uma transação de dados desnecessária, causando uma perda de performance à aplicação. Com este recurso, o carregamento é realizado apenas no primeiro acesso, colocando-a no *cache* do *browser*.

Este processo é feito automaticamente, estendendo a interface *ImageBundle*, para que seja criada a imagem única. É necessário que todas as imagens individuais fiquem localizadas no mesmo *package* definido pelo desenvolvedor. São permitidas somente as extensões de arquivos PNG, GIF e JPG. Essa limitação não representa um empecilho ao desenvolvimento, pois são estas extensões comumente utilizadas.

Quadro 3 – Exemplo de desenvolvimento de *Image Bundles* para ícones.

```
public interface WordProcessorImageBundle extends ImageBundle {

  /**
   * Would match the file 'new_file_icon.png', 'new_file_icon.gif', or
   * 'new_file_icon.png' located in the same package as this type.
   */
  public AbstractImagePrototype new_file_icon();

  /**
   * Would match the file 'open_file_icon.gif' located in the same
   * package as this type.
   *
   * @gwt.resource open_file_icon.gif
   */
  public AbstractImagePrototype openFileIcon();

  /**
   * Would match the file 'savefile.gif' located in the package
   * 'com.mycompany.mygwtapp.icons', provided that this package is part
   * of the module's classpath.
   *
   * @gwt.resource com/mycompany/mygwtapp/icons/savefile.gif
   */
  public AbstractImagePrototype saveFileIcon();
}
```

Fonte: http://code.google.com/webtoolkit/overview.html.

Quadro 4 – Exemplo de utilização de *Image Bundles* defindo pelo quadro acima.

```
public void useImageBundle() {
  WordProcessorImageBundle wpImageBundle = (WordProcessorImageBundle)
GWT.create(WordProcessorImageBundle.class);
  HorizontalPanel tbPanel = new HorizontalPanel();
  tbPanel.add(wpImageBundle.new_file_icon().createImage());
  tbPanel.add(wpImageBundle.openFileIcon().createImage());
  tbPanel.add(wpImageBundle.saveFileIcon().createImage());
}
```

Fonte: http://code.google.com/webtoolkit/overview.html.

A grande diferença entre o método de troca de dados do GWT, em relação ao HTTP, é que não é necessário enviar todos os dados da página, como o *layout* novamente, mas apenas atualizar o que é necessário. Para que isso ocorra, é necessário usar o mecanismo de RPC, igualmente como todas as aplicações cliente/servidor. Utilizando esta tecnologia consegue-se separar a interface gráfica dos dados, ou seja, o *browser* do usuário armazena o *layout* somente na primeira solicitação, enquanto os dados são atualizados, reduzindo a transação de dados e aumentando a performance.

Para realizar o desenvolvimento de novas interfaces, que invoquem e consumam algum serviço, é utilizado a interface Java de *client-side*, que estende a interface *RemoteService*, podendo ser desenvolvida de modo síncrono ou assíncrona. Toda interface desenvolvida usando o modo síncrono deve estender e implementar a interface de serviço *RemoteServiceServlet* no lado servidor. Para o modo assíncrono deve-se, além de implementar a mesma interface do modo síncrono, acrescentar o método de *callback*. Como a chamada é desta natureza, necessita-se que seja feito esse controle, para notificar quando o processamento foi finalizado e poder enviar as informações ao método chamador.

Quadro 5 – Exemplo de criação de uma interface no lado cliente de serviço que estende *RemoteService*.

```
public interface MyService extends RemoteService {
  public String myMethod(String s);
}
```

Fonte: http://code.google.com/webtoolkit/overview.html.

Quadro 6 – Exemplo de criação de uma interface no lado servidor de serviço síncrono que estende *RemoteServiceServlet* e implementa o lado cliente do quadro acima.

```
public class MyServiceImpl extends RemoteServiceServlet implements
    MyService {

  public String myMethod(String s) {
    return s;
  }
}
```

Fonte: http://code.google.com/webtoolkit/overview.html.

Quadro 7 – Exemplo de criação de uma interface no lado servidor de serviço assíncrono que deve complementar um método síncrono referente ao quadro acima.

```
interface MyServiceAsync {
  public void myMethod(String s, AsyncCallback callback);
}
```

Fonte: http://code.google.com/webtoolkit/overview.html.

A criação e utilização de RPCs não é uma tarefa simples, pois dependendo do modo que é desenvolvido, pode-se causar algum problema no servidor Web, assim como no cliente que fez a requisição. Para contornar eventuais problemas, pode-se utilizar os mesmos métodos de *exception* do Java, permitindo a declaração de *throws*. A validação e checagem do andamento são feitas pelo desenvolvedor do lado cliente ou através do método *InvocationException* para o lado servidor.

JavaScript Native Interface (JSNI) permite que, o desenvolvedor, possa acrescentar diretamente algum código Javascript dentro do Java, independente da compilação do GWT. Este recurso demonstra uma flexibilidade do *framework* em oferecer esta opção porém, desse modo, há a perda de portabilidade entre os *browsers*, aumento de consumo da memória e perda de otimização quando o código for compilado.

Quadro 8 – Exemplo de código Javascript dentro do Java usando o GWT.

```
public static native void alert(String msg) /*-{
  $wnd.alert(msg);
}-*/;
```

Fonte: http://code.google.com/webtoolkit/overview.html.

Também é possível interagir com métodos das classes Java através do Javascript, usando o conceito similar do Java em usar código C. Porém, parâmetros e tipos do retorno de

algum método, são tratados como tipos do Java pelo JSNI. Isso se deve, pelo fato de que, os tipos de dados suportados, entre as duas linguagens são, na maioria, diferentes, como por exemplo: um tipo de dado numérico no Javascript não utiliza a definição de inteiro, *double* ou *float* como no Java.

Quadro 9 – Exemplo de código Javascript dentro do GWT interagindo com métodos Java.

```
public native void bar(JSNIExample x, String s) /*-{
   this.@com.google.gwt.examples.JSNIExample::instanceFoo(Ljava/lang/String;)(s);
   @com.google.gwt.examples.JSNIExample::staticFoo(Ljava/lang/String;)(s);
}-*/;
```

Fonte: http://code.google.com/webtoolkit/overview.html.

O componente de *browser* hospedado tem a funcionalidade de permitir que execute o aplicativo Web localmente, ou seja, o código é executado diretamente, na JVM. Este recurso ignora a compilação, para o Javascript, pois o código desenvolvido em Java é posto em funcionamento para realizar o processo de depuração de erro (*debug*).

A biblioteca de classes Web de IU é um *framework* de classes e interfaces personalizadas. Esta é, a principal, biblioteca utilizada para o desenvolvimento de interfaces dos aplicativos pois, praticamente, possui toda a implementação dos recursos gráficos simples e avançados, como exemplo: botões, caixas de texto, imagem e *stackpanel* .

As bibliotecas Java de emulação JRE disponibilizam implementações das classes mais utilizadas do Java em Javascript. Incluindo grande parte do *package* java.lang e um subconjunto de java.util. Estes dois *packages* são os necessários para qualquer desenvolvimento em Java pois, respectivamente, implementam classes e métodos básicos da linguagem, como por exemplo: tipos de dados, exceções, internacionalização.

2.2 Exemplos de aplicativos com GWT

Este subtítulo apresenta alguns exemplos de aplicativos Web desenvolvidos com o Google Web Toolkit. Pode-se citar, também como exemplo bastante difundido e utilizado atualmente, o *Gmail*, que foi totalmente construído com essa tecnologia.

2.2.1 Hello World

Neste exemplo, fornecido pelo próprio GWT, há a familiarização com o desenvolvimento, através do passo a passo básico fornecido pelo Sang Shin[1]. Utilizando-se a IDE de desenvolvimento do NetBeans juntamente com o *framework* GWT instalado.

Após iniciado o NetBeans e selecionado a opção de "*New Project*" do menu "*File*", seleciona-se o tipo de projeto "*Web Application*" entre as opções encontradas dentro da guia "*Categories*".

[1] Maiores informações em http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

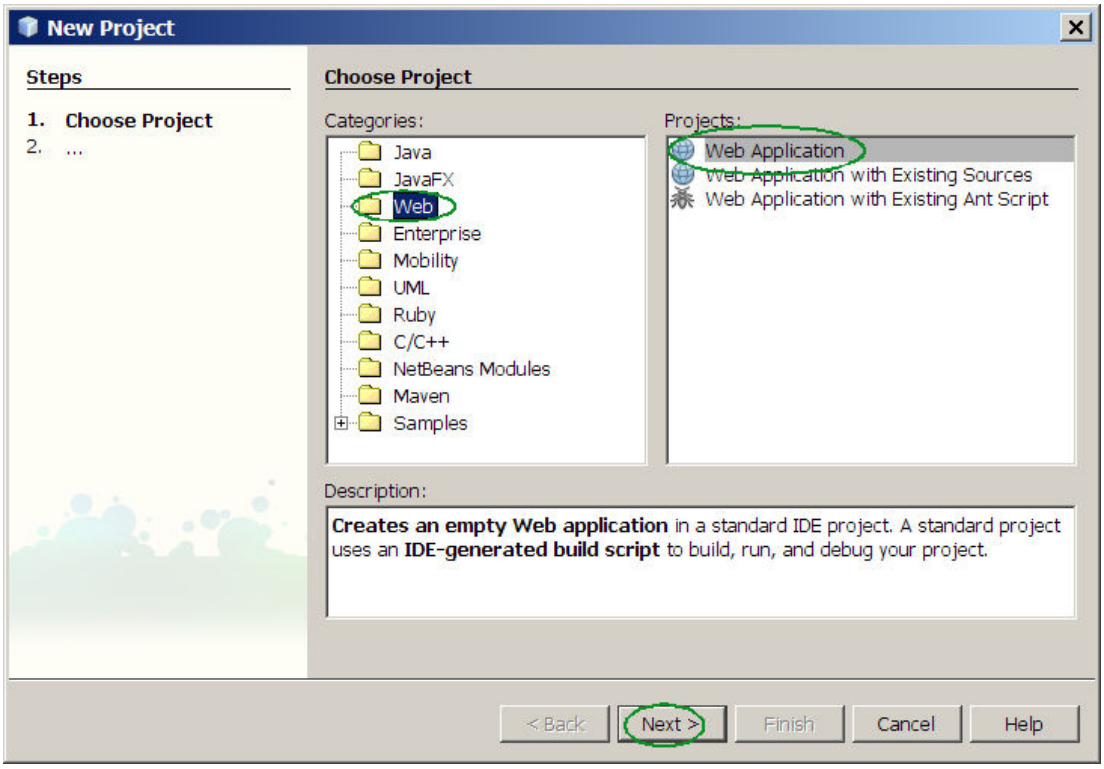

Figura 16 – Tela do *Choose Project* do NetBeans.
Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

Na tela da etapa, denominada "*Name and Location*", informe-se o nome do projeto e localização que será salvo no *desktop*. Para este exemplo utiliza-se, respectivamente, "*GWTApplication*" e "C:\mygwtprograms".

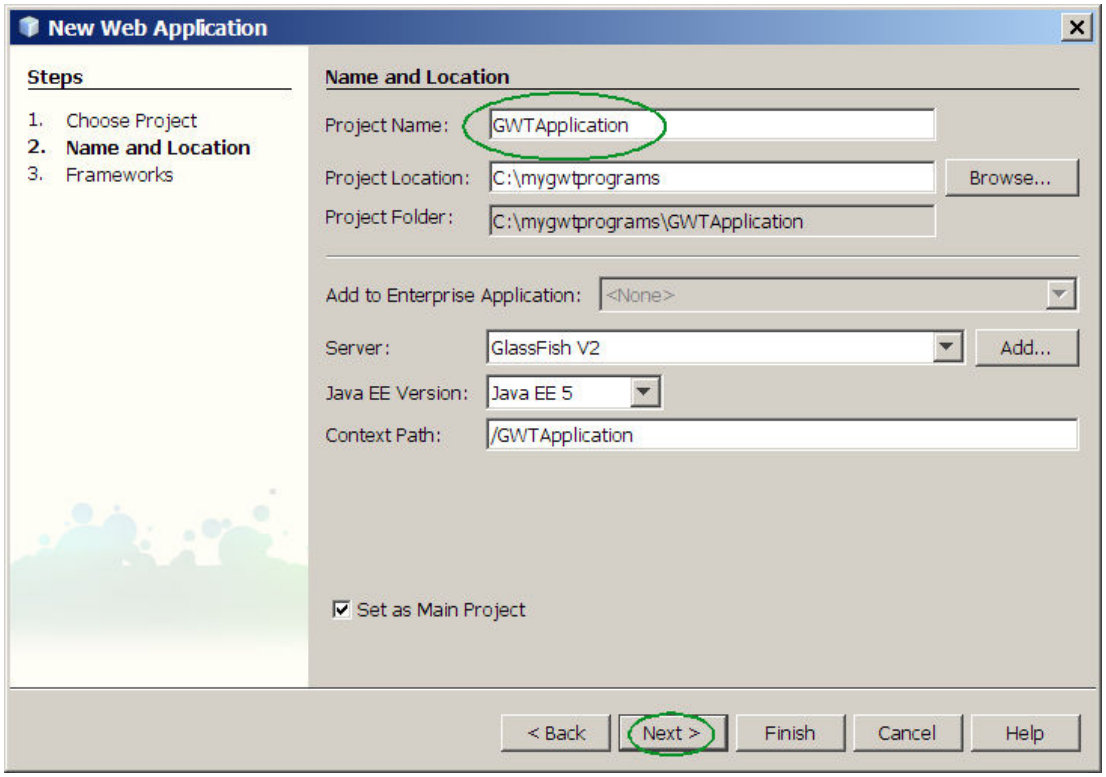

Figura 17 – Tela de configuração da opção *Name and Location* do NetBeans.
Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

Na etapa "*Frameworks*" selecione-se a opção "*Google Web Toolkit*" e indica-se o caminho do diretório, onde está instalado o GWT na opção "*GWT Installation Folder*". Somente na primeira utilização do *framework* é necessário realizar esse procedimento.

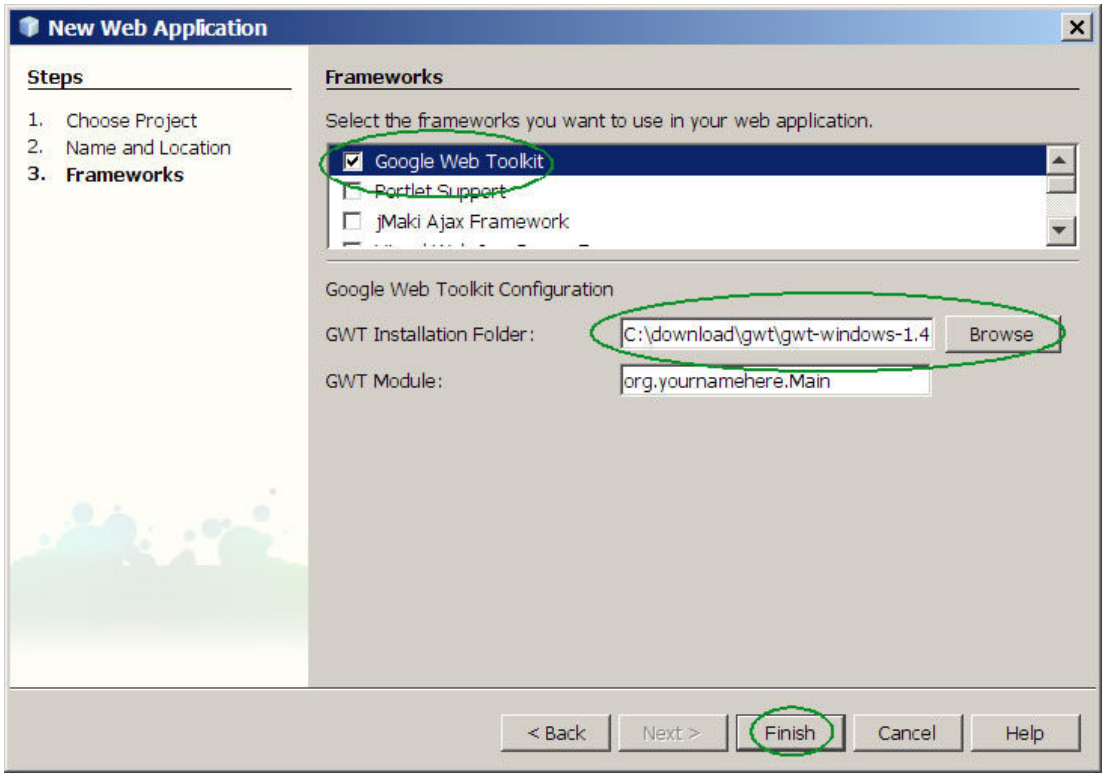

Figura 18 – Tela de configuração da opção Framework do NetBeans.
Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

Após a conclusão da importação e configuração do projeto *HelloWorld* visualiza-se todos os arquivos na guia "*Projects*" que compõe a aplicação. Realmente, utiliza-se apenas Java para desenvolver com o GWT, conforme demonstrado pelo arquivo *MainEntryPoint.java*.

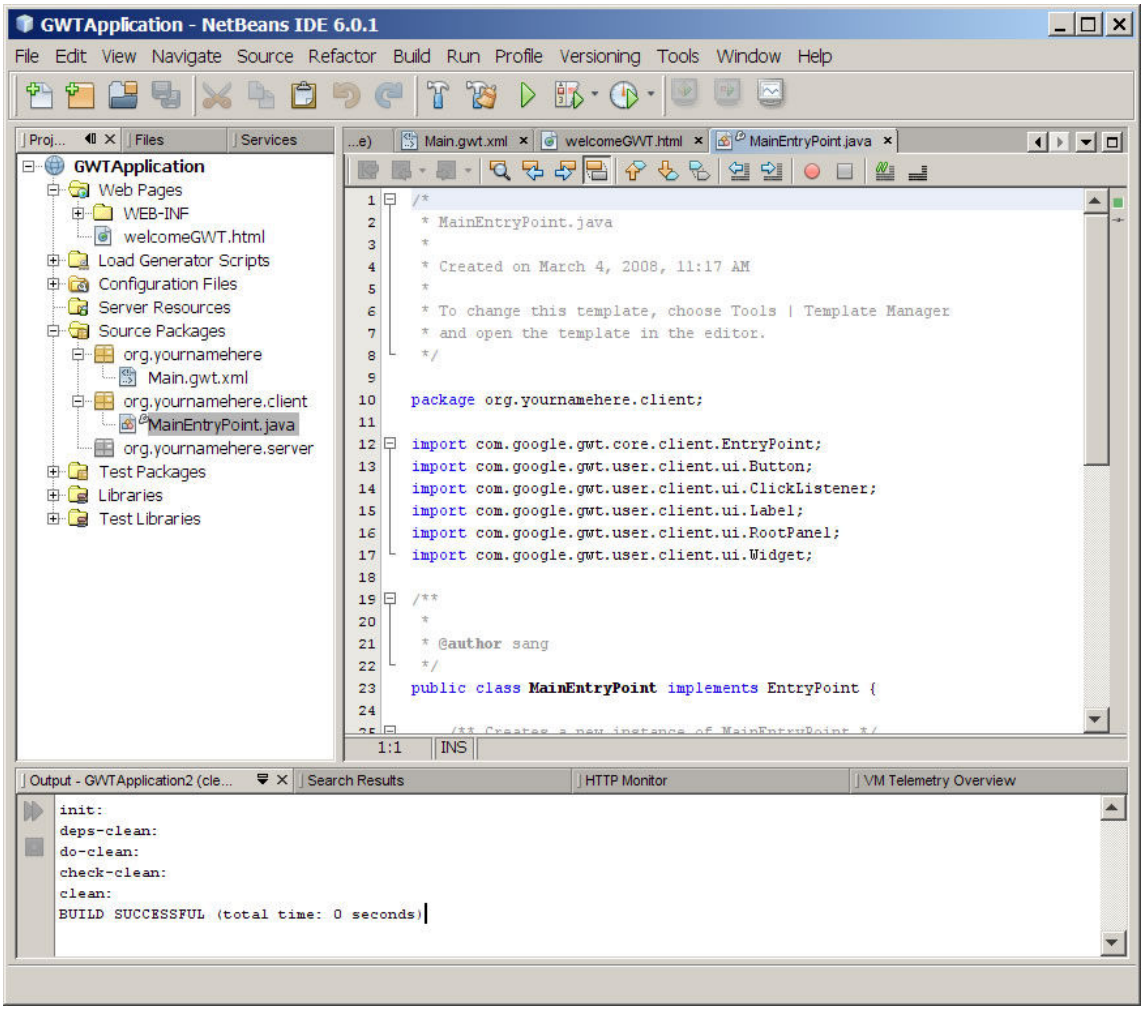

Figura 19 – Tela do ambiente do NetBeans visualizando *MainEntryPoint.java.*

Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

Ao executar o projeto, o *browser* é aberto, juntamente com a aplicação funcionando. Para testar o funcionamento, executa-se a ação do *link*, denominado de "*GWT Page*". Onde é realizada a ação correspondente, que está determinada ao código Java realizar.

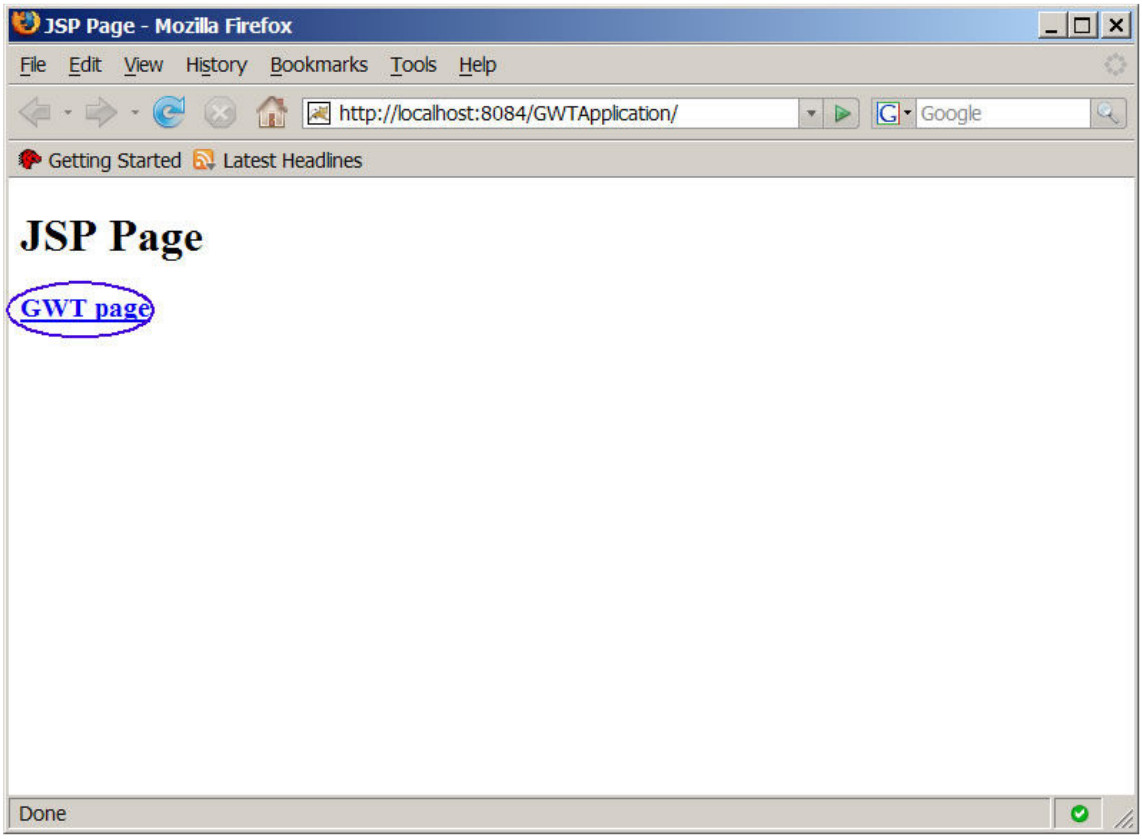

Figura 20 – *Browser* com a aplicação em execução.

Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

Obtém-se como resultado da ação a tela contendo um botão "*click me!*" e, logo abaixo, a frase *"Hello, GWT"*. A ação correspondente a esse controle é de ocultar a frase.

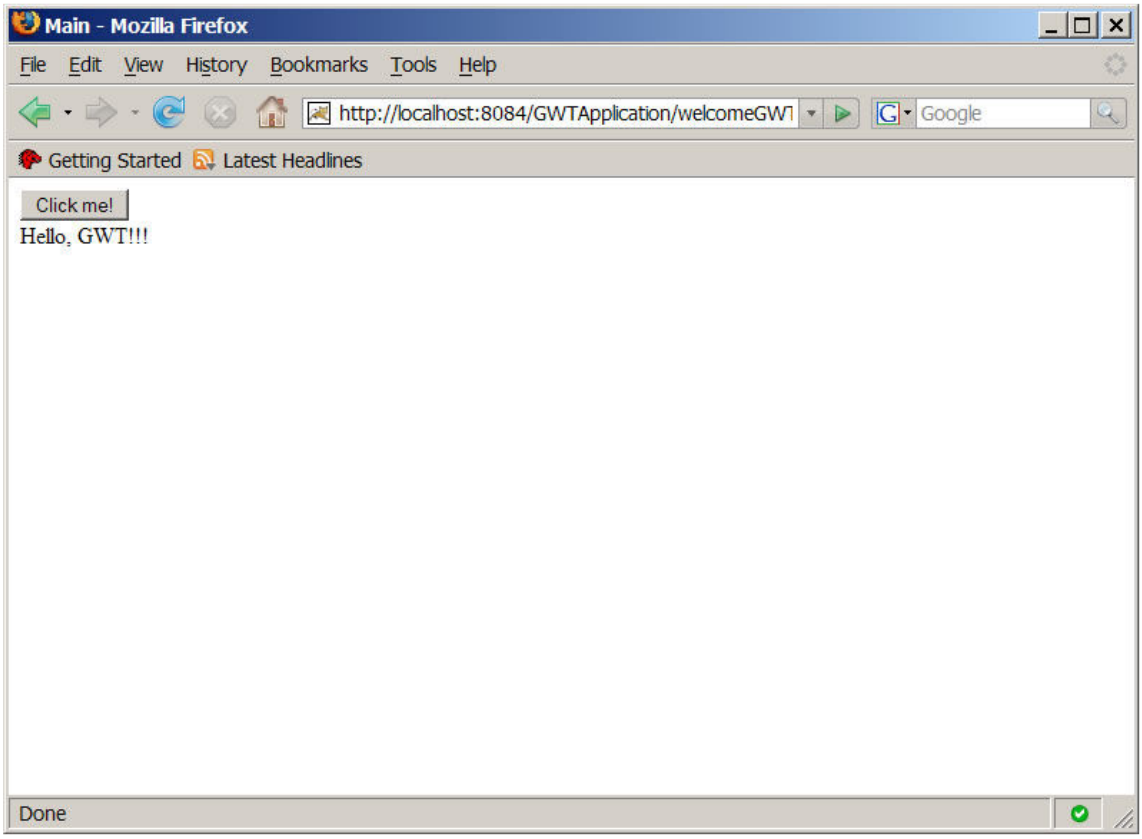

Figura 21 – *Browser* com a aplicação em execução com ênfase para a frase *"Hello, GWT"*.

Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

No exemplo acima pode-se observar que o desenvolvimento de uma aplicação Web torna-se simples e produtivo. Pois, apenas utilizando-se de código Java é possível realizar o desenvolvimento completo de uma solução, para a Web, sem a necessidade de utilizar outros tipos de codificação.

2.2.2 Adicionando código ao Hello World

Para este exemplo será utilizado a mesma aplicação que foi demonstrada na seção anterior. Pretende-se adicionar código a fim de acrescentar uma funcionalidade simples, demonstrando a facilidade e produtividade proposta pelo GWT.

Abrindo-se o arquivo *MainEntryPoint.java* tem-se a classe abaixo, contendo o método responsável em apresentar o botão "*Click me!*" juntamente com a frase *"Hello, GWT"*.

Quadro 10 – Demonstração do código contido no arquivo *MainEntryPoint.java*.

```
package org.yournamehere.client;    //refere-se ao domínio na Internet da Aplicação Web

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.ClickListener;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.Widget;

/**
 * @author sang
 */
public class MainEntryPoint implements EntryPoint {

   /** Creates a new instance of MainEntryPoint */
   public MainEntryPoint() {
   }

   /**
    * The entry point method, called automatically by loading a module
    * that declares an implementing class as an entry-point
    */
   public void onModuleLoad() {
      final Label label = new Label("Hello, GWT!!!");
      final Button button = new Button("Click me!");

      button.addClickListener(new ClickListener(){
         public void onClick(Widget w) {
            label.setVisible(!label.isVisible());
         }
      });

      RootPanel.get().add(button);
      RootPanel.get().add(label);
   }

}
```

Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html.

Para este exemplo é adicionado mais um botão denominado *"I am the 2nd Button. Click me!"*. Acrescenta-se o código Java extra, destacado em negrito, dentro do arquivo *MainEntryPoint.java*. Tem-se como objetivo exibir uma janela de alerta do Javascript quando realizar a ação prevista a esse novo controle.

Quadro 11 – Demonstração do código contido no arquivo *MainEntryPoint.java* com destaque para a modificação do novo botão.

```
package org.yournamehere.client;    //refere-se ao domínio na Internet da Aplicação Web

import com.google.gwt.core.client.EntryPoint;
import com.google.gwt.user.client.Window;
import com.google.gwt.user.client.ui.Button;
import com.google.gwt.user.client.ui.ClickListener;
import com.google.gwt.user.client.ui.Label;
import com.google.gwt.user.client.ui.RootPanel;
import com.google.gwt.user.client.ui.Widget;

/**
 * @author sang
 */
public class MainEntryPoint implements EntryPoint {

   /** Creates a new instance of MainEntryPoint */
   public MainEntryPoint() {
   }

   /**
    * The entry point method, called automatically by loading a module
    * that declares an implementing class as an entry-point
    */
   public void onModuleLoad() {
      final Label label = new Label("Hello, GWT!!!");
      final Button button = new Button("Click me!");

      button.addClickListener(new ClickListener(){
         public void onClick(Widget w) {
            label.setVisible(!label.isVisible());
         }
      });

      Button button2 = new Button("I am the 2nd button. Click me!");

      button2.addClickListener(new ClickListener(){
         public void onClick(Widget w) {
            Window.alert("Life is worth living.. with Passion!");
         }
      });
```

```
        RootPanel.get().add(button);
        RootPanel.get().add(button2);
        RootPanel.get().add(label);
    }
}
```

Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html.

Ao executar a aplicação novamente e repetindo-se a ação do *link "GWT page"*, obtem-se a página exibindo os mesmo elementos anteriores e o novo botão.

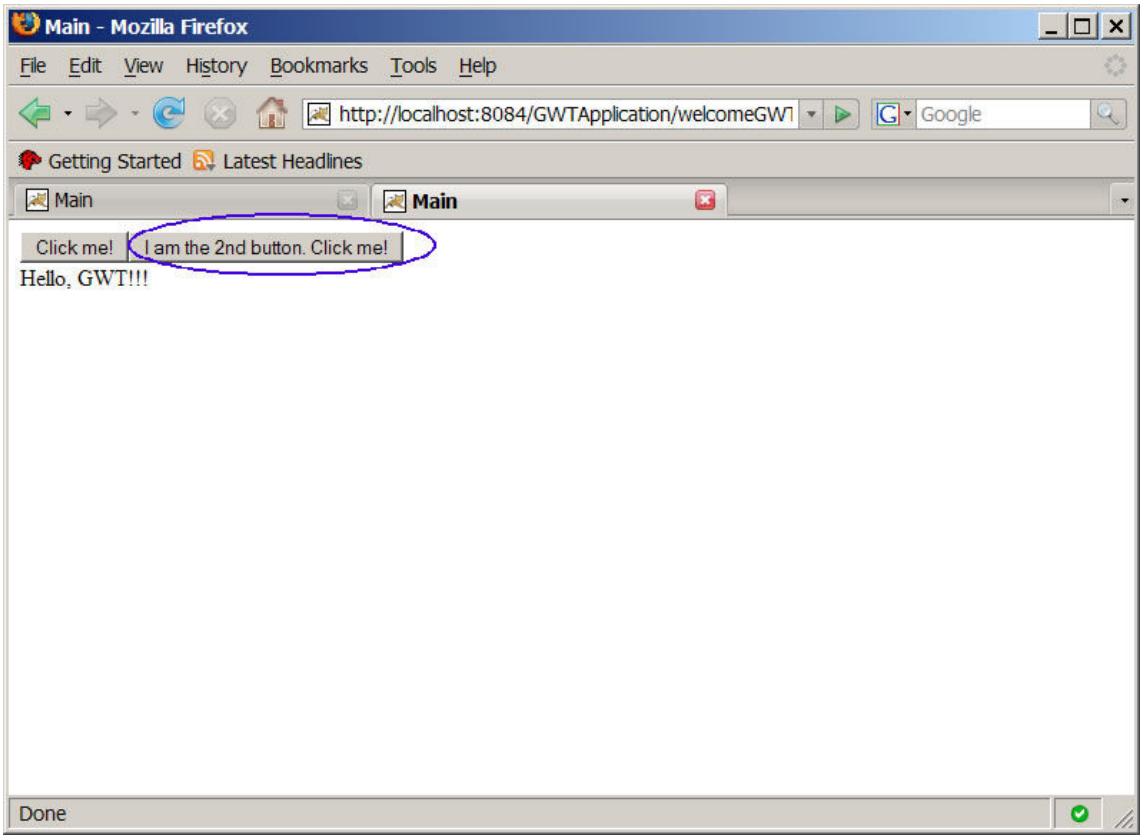

Figura 22 – *Browser* com a aplicação em execução com ênfase para a frase *"Hello, GWT"*.

Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

Quando se utiliza o novo botão, é executada a ação de exibir uma mensagem de alerta do Javascript. Isso ocorre sem a necessidade de implementar nenhum código além do Java, apenas se utilizando o *package com.google.gwt.user.client.Window* contido dentro do GWT.

Figura 23 – Mensagem de alerta do botão *"I am the 2nd Button. Click me!"*.

Fonte: http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html

Os *packages* disponíveis no GWT oferecem, ao desenvolvedor, uma enorme biblioteca de funcionalidades. Tornando o desenvolvimento uma tarefa produtiva e rápida de acordo com o mercado atual.

CONCLUSÃO

Este trabalho apresentou uma análise dos processadores de texto para a Web, demonstrando o funcionamento e a particularidade de cada um, e também sobre o *framework Google Web Toolkit*, do Google, juntamente com alguns exemplos simples de funcionamento.

Na parte que apresentou a análise dos processadores de texto foram apresentados os convencionais e os específicos para Latex. Onde os convencionais procuram ser o mais semelhante possível com os relativos para o *desktop*. Apresentando os mesmos recursos e adicionando outros que utilizam ao máximo o ambiente da Internet. Os específicos para o Latex desapontaram pois, na maioria, são simplesmente um campo de texto, que realiza a conversão para o arquivo final, sem apresentar nenhuma ferramenta a altura dos convencionais.

Na parte que se falou sobre o *framework Google Web Toolkit* foi apresentado o novo conceito do Google para o desenvolvimento de aplicações Web e um simples exemplo. Com a finalidade de proporcionar, ao desenvolvedor, a possibilidade de desenvolver aplicativos ricos com o conceito de Web 2.0, utilizando-se apenas a linguagem Java e os recursos contidos no GWT, sem precisar do conhecimento prévio de outras linguagens, criando um ambienta uniforme e produtivo.

Na segunda parte deste trabalho pretender-se-á aplicar os conhecimentos adquiridos até então para propor o desenvolvimento de um editor de textos Latex para a Web. Serão feitos a definição do escopo, protótipo e implementação da ferramenta proposta.

REFERÊNCIAS

BERLITZ, Igor Henrique. **Gerador gráfico de relatórios utilizando a classe FPDF**. Novo Hamburgo: 2007. Monografia (Trabalho de Conclusão em Ciência da Computação) – Instituto de Ciências Exatas e Tecnológicas, FEEVALE, 2007.

CAMELO, Dioclécio Moreira. **Web service**: Curso de Especialização de Sistemas de informação e aplicações web. 2002. Disponível em: <http://www.cqgp.sp.gov.br/downloads/WebServices.pdf>. Acessado em: 29 mar. 2008.

FERNÁNDEZ, David. **Web 2.0**: Tecnologia e tendências por trás do conceito. 2007. Disponível em: <http://www.convergenciadigital.com.br/cgi/cgilua.exe/sys/start.htm?infoid=7096&sid=15>. Acesso em: 27 mar. 2008.

Google Docs. **Google Docs**, 2008. Disponível em: <http://docs.google.com>. Acesso em: 25 mai. 2008.

GWT. **Google Web Toolkit**, 2007. Disponível em: <http://code.google.com/webtoolkit/overview.html>. Acesso em: 1 jun. 2008.

HEFFERON, Jim. **Why TEX?**. 2002. Disponível em: < http://www.tug.org/TUGboat/Articles/tb22-1-2/tb70heff.pdf >. Acesso em: 23 mar. 2008.

KOPKA, Helmut et. al. **A guide to Latex**: document preparation for beginners and advanced users. 3. ed. Boston: Addison-Wesley, 2003.

LAMPORT, Leslie. **Latex**: a document preparation system: user's guide and reference manual. 2. ed. Boston: Addison-Wesley, 2003.

NUNES, Tíssia et. al. **Cadernos Eletrônicos 2**: WEB 2.0 e aplicativos on-line. São Paulo: 2006. Disponível em: <http://www.acessasp.sp.gov.br/cadernos/caderno_10_02.htm>. Acesso em: 20 mar. 2008.

OETIKER, Tobias et. al. **The not so short introduction to Latex2**. 2007. Disponível em: <http://ctan.tug.org/tex-archive/info/lshort/english/lshort.pdf>. Acesso em: 23 mar. 2008.

OLIVEIRA, Maria da Conceição C et. al. **Cadernos Eletrônicos 2**: Editoração e processamento de textos. São Paulo: 2003. Disponível em: <http://www.acessasp.sp.gov.br/cadernos/caderno_10_02.htm>. Acesso em: 18 mar. 2008.

SANTOS, Reginaldo J. **Introdução ao Latex**. Minas Gerais: Universidade Federal de Minas Gerais, 2002. Disponível em: <http://www.mat.ufmg.br/~regi>. Acesso em: 22 mar. 2008.

SHIN, Sang. **Using Google Web Toolkit (GWT) and NetBeans for Building AJAX Applications**, 2008. Disponível em: < http://www.javapassion.com/handsonlabs/ajaxgwtintro/index.html>. Acesso em: 15 jun. 2008.

Think Free. **Think Free Docs**, 2007. Disponível em: <http://thinkfreedocs.com>. Acesso em: 25 mai. 2008.

Zoho Writer. **Zoho Writer**, 2007. Disponível em: <http://writer.zoho.com>. Acesso em: 25 mai. 2008.