FACULDADE DE ADMINISTRAÇÃO E NEGÓCIOS DE SERGIPE - FANESE

NÚCLEO DE PÓS-GRADUAÇÃO E EXTENSÃO – NPGE

CURSO DE PÓS-GRADUAÇÃO “LATO SENSU”

ESPECIALIZAÇÃO EM SISTEMA DE INFORMAÇÃO PARA WEB

FRAMEWORK GWT-GOOGLE WEBTOOLKIT

Aracaju - SE

2014
MARCOS AUGUSTO DOS SANTOS

FRAMEWORK GWT-GOOGLE WEBTOOLKIT

Trabalho de Conclusão de Curso apresentado ao Núcleo de Pós-Gra-duação e Extensão da FANESE, como requisito para obtenção do título de Especialista em Sistema de Informação para Web.

Orientador:

Aracaju - SE
2014

MARCOS AUGUSTO DOS SANTOS

FRAMEWORK GWT-GOOGLE WEBTOOLKIT

Trabalho de Conclusão de Curso apresentado ao Núcleo de Pós-Graduação e Extensão – NPGE, da Faculdade de Administração de Negócios de Sergipe – FANESE, como requisito para a obtenção do título de Especialista em

_____

Nome completo do Avaliador

_____

Nome completo do coordenador do curso

_____

Nome completo do Aluno

Aprovado (a) com média: _____

Aracaju (SE), ____ de _____ de 2014

RESUMO

GWT é uma ferramenta para construção de interfaces Web mais amigáveis e rápidas. O GWT foi feito para facilitar e agilizar o processo de desenvolvimento das interfaces de aplicações Web utilizando AJAX.

O desenvolvimento de aplicações que utilizam o conjunto de tecnologias que hoje são chamadas de AJAX já é lugar comum, mas todo esse movimento trouxe novos problemas, como o aumento considerável da quantidade de código *JavaScript* em aplicações web. Essa volta do *JavaScript* a tona no mercado trouxe problemas da não tão distante guerra dos navegadores web, onde cada fornecedor (na época a Microsoft e a Netscape) implementavam a linguagem *JavaScript* de uma forma diferente.

Isso terminou por gerar problemas, principalmente para os desenvolvedores, que tinham reescrever partes intera das aplicações para garantir a compatibilidade entre os diversos navegadores. A implementação diferenciada do AJAX pelos navegadores atuais trouxe mais uma vez o fantasma da incompatibilidade do *JavaScript* para a vida dos desenvolvedores. Hoje, com o grande apelo dessa nova tecnologia, surgiram diversos frameworks que prometem simplificar o desenvolvimento de aplicações utilizando AJAX e um destes frameworks é o Google Web Toolkit (**GWT**), desenvolvido dentro do Google e liberado pelos seus desenvolvedores como software livre. Diferentemente de outros frameworks, que normalmente fazem com que você ainda tenha que escrever “ganchos” em *JavaScript* para a sua aplicação utilizar as facilidades do AJAX, o GWT deixa com que você escreva o código da aplicação completamente em Java, para que você possa ter todas as facilidades que o desenvolvimento Java provê, como ambientes de desenvolvimento integrados de alta qualidade, tipagem estática e debug nativo. Após escrita a aplicação, um compilador especial transforma o código Java em código *JavaScript*, para que ele possa executar no navegador cliente. E como você já deve imaginar, os desenvolvedores do projeto se preocuparam com a compatibilidade e fizeram com que o código gerado fosse garantidadamente compatível com os navegadores mais utilizados da atualidade.

A abordagem to GWT, entretanto, tem uma desvantagem, pois como o código Java é transformado em código.

JavaScript para executar no navegador, apenas algumas classes do Java estão disponíveis para uso (em sua maioria classes dos pacotes “java.lang” e “java.util”) e funcionalidades avançadas da linguagem (como acesso a bancos de dados) tem que ser feitos utilizando o suporte a invocação remota de métodos do GWT, que faz a ponte entre a aplicação no navegador e o código Java no servidor.
Palavras-chave: *Java script*

ABSTRACT

GWT is a tool for building more friendly and fast Web interfaces . GWT has been done to facilitate and expedite the process of development of Web applications using AJAX interfaces .

The development of applications using the set of technologies that are now called AJAX is now commonplace , but this whole movement has brought new problems , such as the substantial increase in the amount of JavaScript code in web applications . This turn of JavaScript in the market brought to light problems of not so distant war of web browsers , where each supplier (then Microsoft and Netscape) were implementing the JavaScript language in a different way .

This eventually lead to problems , especially for developers who had to rewrite parts of the interaction applications to ensure compatibility between different browsers . The differentiated AJAX implementation by current browsers brought again the ghost of the incompatibility of JavaScript to the lives of developers . Today , with the great appeal of this new technology , many frameworks have emerged that promise to simplify the development of applications using AJAX and frameworks is one of the Google Web Toolkit (GWT) , developed within Google and released by its developers as free software . Unlike other frameworks , which normally cause you still have to write " hooks " in JavaScript for your application to use the facilities of AJAX , GWT lets you to write the application code completely in Java , so you can have all facilities that provide Java development , integrated development environments such as high quality , static typing and debug native . Upon written application , a special compiler turns Java code into JavaScript code , so it can run on the client browser . And as you might guess , the project developers were concerned with the compatibility and made guaranteed were generated code compatible with more browsers used today.

The approach to GWT , however, has a disadvantage because as the Java code is converted into code.

JavaScript to execute in the browser , only certain classes of Java are available for use (mostly classes package " java.lang " and " util ") and advanced language features (such as access to databases) must be made using the support remote method invocation GWT , which bridges the gap between the application in the browser and Java code on the server .

SUMÁRIO

1. **O FRAMEWORK GWT**

É um framework *opensource* desenvolvido pelo Google para a camada de apresentação com desenvolvimento java que possui um conjunto de ferramentas e um conjuntos de componentes visuais projetados para desenvolvimento em aplicações web, onde o seu código que é executado no *browse* é feito em java, eliminando toda a necessidade que o desenvolvedor tenha de se escrever código *javascript*, jsp, *taglibs*, html, etc. O objetivo é tornar o desenvolvimento de aplicações web semelhante ao de um aplicativo desktop (orientado a eventos). Rivaliza com o *JavaServerFaces* (JSF) da Sun. Onde todo o sistema é criado em java, com classes e *breackpoint*. Com isso muitos erros são encontrados em tempo de compilação. Além disso, o GWT gera seu código compatível com os *browses* mais utilizados. Nesses aspectos desenvolvedores que não utilizam esta ferramenta sabe que não é uma tarefa simples quanto utilizando o GWT.

2. **COMO FUNCIONA FRAMEWORK GWT**

Com o Google Web Toolkit (GWT), você cria o *front end AJAX* na linguagem de programação Java e o GWT, então, faz a compilação cruzada para o *javascript* otimizado que funciona automaticamente com todos os principais navegadores. Durante o desenvolvimento, é possível criar rapidamente como no *javascript*, no mesmo ciclo "editar - atualizar - exibir" com o qual você está acostumado, com a vantagem adicional de poder depurar e percorrer o código Java linha por linha. Quando você estiver pronto para implementar, o GWT compila o código fonte Java nos arquivos independentes otimizados *javascript*. Crie com facilidade um *widget* para uma página da web ou todo um aplicativo usando o *Google Web Toolkit*.

Crie aplicativos *AJAX* na linguagem Java e, depois, compile em *javascript* otimizado.

Ao contrário dos *minifiers javascript* que funcionam somente em nível textual, o compilador GWT executa análises estáticas abrangentes e otimizações em toda a base de códigos do GWT, produzindo freqüentemente *javascript* que carregam e executam mais rapidamente do que um *javascript* equivalente criado por você. Por exemplo, o compilador GWT elimina código sem função com segurança, cortando agressivamente classes, métodos, campos e mesmo parâmetros de métodos não utilizados, para garantir que o script compilado seja o menor possível. Outro exemplo: o compilador GWT incorpora métodos de forma seletiva, eliminando os excessos das

chamadas do método.

A compilação cruzada permite que você mantenha as abstrações e a modularidade necessárias para o desenvolvimento, sem prejudicar o desempenho do tempo de execução.

3. CARACTERISTICAS

3.1. Compilador JavaScript

A característica mais interessante do GWT é o seu compilador de Java para *javascript*. GWT converte um subconjunto (bastante grande) da linguagem de programação Java para *javascript*. Basicamente toda a sintaxe da linguagem Java é suportada (inclusive a sintaxe Java 5, suportada a partir do GWT 1.5) com algumas poucas exceções:

- nem todas as classes da biblioteca padrão de Java são emuladas, ex: *LinkedList*, *StringTokenizer*, *Logging* e *BigNumber* são algumas das classes que estão faltando.
- existem algumas outras restrições como por exemplo a falta de reflexão, *multithreading*, sincronização, *Throwable.getStackTrace*() e asserções.

3.2. Aumento de desempenho

Além disto uma preocupação constante da equipe do google enquanto desenvolvem o GWT é em torná-lo eficiente, produzindo código *javascript* rápido e o menor possível. Para isto, uma das medidas tomada para otimizar o espaço ocupado pelo *javascript* gerado é produzir apenas o código que é utilizado, para isto, todo o código produzido é analizado para determinar se existe alguma parte que nunca é executada, qualquer trecho que não for executado é automaticamente removido do *javascript* gerado. Além disto, só são incluídas as funções da biblioteca que são utilizadas.

Outra característica que poucos programadores conhecem a respeito do comportamento dos browsers é que estes abrem normalmente uma ou duas conexões para fazer o download dos arquivos *javascript* incluídos em uma página. Este download é feito de forma sequencial. Ou seja, há um *tradeoff* para os

programadores *javascript* tradicionais entre modularidade e velocidade. No caso do GWT, todas as funções *javascript* são agrupadas em um único arquivo de modo a tornar mais ágil este processo de download. Por fim, todo o código gerado é ofuscado e compactado (utilizando um filtro gzip no servidor web), deixando o tamanho final dos artefatos muito menor do quê provavelmente um programador conseguiria fazer manualmente.

3.3. **Múltiplos browsers**

Quem já precisou desenvolver algum código em *javascript* para múltiplos browsers ou mesmo teve a oportunidade der ler algum código feito para este fim, provavelmente deparou-se com uma infinidade de “ifs” que tentam determinar qual o browser que está acessando a página para realizar a mesma tarefa com pequenas modificações. Os desenvolvedores do GWT além de isolar os programadores deste tipo de problema ainda tiveram uma ideia bastante interessante, o compilador de *javascript* gera uma versão da aplicação para cada tipo de browser diferente e o servidor web fornece para cada cliente apenas a versão adequada ao seu browser.

3.4. **Múltiplos Idiomas:**

Da mesma forma que o GWT gera uma versão diferente da aplicação para cada browser, a mesma estratégia é utilizada caso você utilize internacionalização em sua aplicação. Desta forma, o compilador *javascript* gera a combinação de todos os idiomas com todos os browsers possíveis, assim, cada cliente recebe apenas a versão da aplicação para o seu browser e para o seu idioma.

4. **VANTAGENS**

- Evita incompatibilidades entre *browsers* e plataformas;
- Erros encontrados em tempo de compilação;
- Suporte o teste Unitário com Junit;

- Elimina trechos de código não utilizados;
- Disponível para Windows, Linux e Mac;
- Projeto *Open-Source* da Google;
- Não é necessário saber *Java-Script*;
- Economia de Espaço;
- Processo de download diferenciado;
- O código gerado é compatível com a maioria dos navegadores;
- Suporte especial ao botão “voltar” do navegador; Integração completa com o JUnit;
- Suporte a internacionalização padrão do Java;
- Controle total sobre a aplicação e possibilidade de extensão utilizando a JSNI (*JavaScriptNativeInterface*);
- (Quase) todo o código escrito é Java, nada de absurdos de *JavaScript*;
- Muitos componentes;

5. **DESVANTAGENS**

- Nem todas as classes da biblioteca padrão de Java são emuladas;
- Há menor controle sobre o código do cliente no aplicativo. Não é possível utilizar todas as classes do Java, apenas um conjunto delas;
- Não é possível utilizar todas as classes do Java no lado cliente, apenas um conjunto delas.
- Há menor controle sobre o código do cliente no aplicativo Difícil de *debugar* CSS
- Difícil de simular integração com outras tecnologias (ex. *Flash*).

6. **COMUNICAÇÃO CLIENTE SERVIDOR**

Assíncrona

- Envia requisição ao servidor e espera resposta, mantendo interface respondendo a entradas do usuário, podendo inclusive enviar/enfileirar outras requisições.

- Não garante que a ordem de chegada das respostas das requisições seja a mesma ordem das requisições.

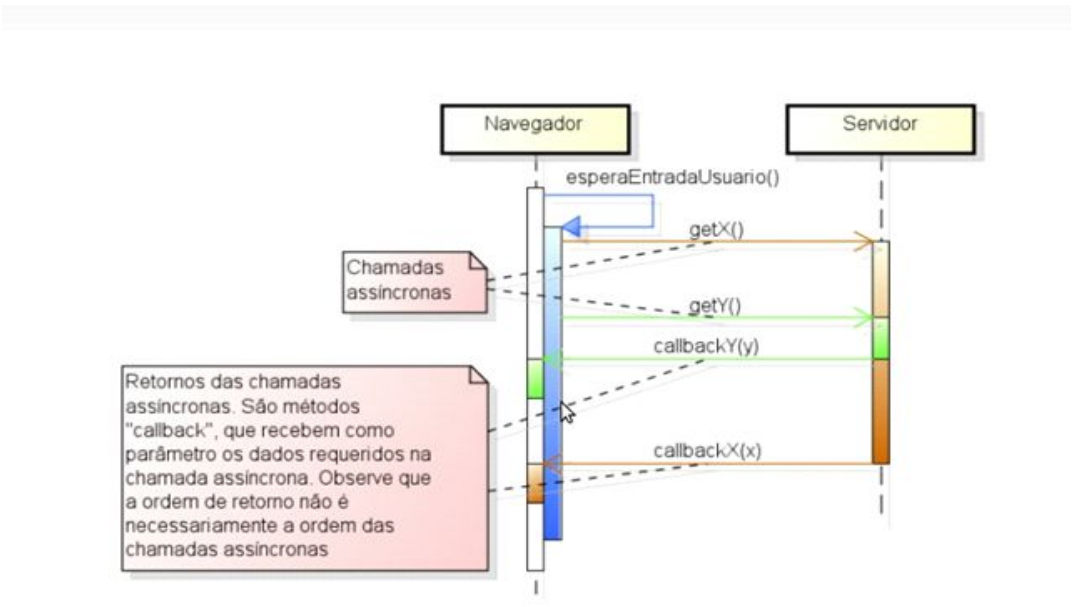
Funcionamento geral para chamada assíncronas.

- Prepara-se o método que será chamado, passando os parâmetros que serão utilizados

- Um objetivo *callback* é passado para receber a resposta da requisição quando ela é disparada. Esta resposta é recebida a partir do método *callback*, que são:

- *onSucess*: nenhum erro ocorreu na requisição que ela retornou, tem como parâmetro tipo de retorno da chamada executada.

- *onFilure*: alguma falha ocorre na requisição. Tem como parâmetro o Objeto que especifica o erro.



Toda lógica e estrutura inicial das telas já estão no lado cliente ao carregar o

módulo, porém sem nenhum dado

Se uma tela que fica visível depende de dados do servidor para a sua inicialização ela deveria ser exibida desabilitada até que os dados necessários cheguem

Possibilidades comunicação cliente servidor

- Utilização RPC

  RPC é uma tecnologia popular para a implementação do modelo cliente-servidor de computação distribuída. Uma chamada de procedimento remoto é iniciada pelo cliente enviando uma mensagem para um servidor remoto para executar um procedimento específico. Uma resposta é retornada ao cliente. Uma diferença importante entre chamadas de procedimento remotas e chamadas de procedimento locais é que, no primeiro caso, a chamada pode falhar por problemas da rede. Nesse caso, não há nem mesmo garantia de que o procedimento foi invocado.

  Chamada Remota de Procedimento (ou RPC, acrônimo de Remote *Procedure Call*) é uma tecnologia de comunicação entre processos que permite a um programa de computador chamar um procedimento em outro espaço de endereçamento (geralmente em outro computador, conectado por uma rede). O programador não se preocupa com detalhes de implementação dessa interação remota: do ponto de vista do código, a chamada se assemelha a chamadas de procedimentos locais.

  É comum para alguém que esteja começando a programar com o GWT ter um pouco de dificuldade com a comunicação entre cliente e servidor através do mecanismo padrão do GWT, o RPC. Após receber alguns e-mails sobre como fazer isso resolvi escrever este post mostrando de uma forma simplificada e rápida os passos necessários para a execução dessa tarefa simples e extremamente comum em aplicações *ajax*.

  O *ajax:* é o uso metodológico de tecnologias como *Javascript* e XML, providas por navegadores, para tornar páginas Web mais interativas com o usuário, utilizando-se de solicitações assíncronas de informações. *ajax* não é somente um novo modelo, é também uma iniciativa na construção de aplicações Web mais dinâmicas e criativas. *AJAX* não é uma tecnologia, mas um conjunto de tecnologias conhecidas trabalhando juntas, cada uma fazendo sua parte, tipo carregar e

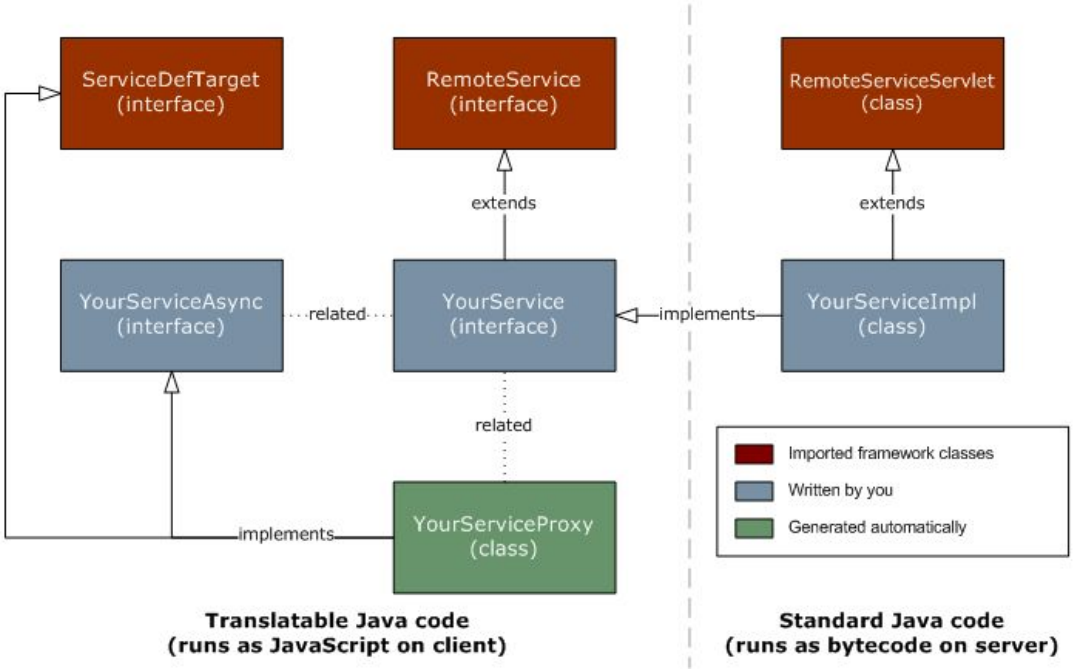
renderizar uma página, utilizando recursos de *scripts* rodando pelo lado cliente, buscando e carregando dados em *background* sem a necessidade de *reload* da página.

Para começar é interessante saber como funciona esse mecanismo. De forma resumida o que acontece é o seguinte: Ao receber o seu objeto java, o compilador GWT vai transformá-lo em uma *String* então o serializa e em seguida envia ao servidor por meio de uma conexão HTTP normal. Ao receber esse objeto serializado o servidor(*Servlet* GWT) então deserializa o objeto executando o caminho inverso, transformando a *String* em um objeto java e então este segue o seu ciclo de vida normal. Da mesma maneira, o inverso também é verdade. Assim, a comunicação pode ser efetuada em ambos os sentidos. Para conseguir executar toda essa tarefa é necessário que o objeto a ser enviado obedeça a algumas regras, a seguir:

- Ser de um tipo básico/primitivo (*String, Integer/int, Double/double, Long/long, Boolean/boolean*).
- Implementar a interface *Serializable* ou *IsSerializable*(GWT < 1.4).
- Todos os campos do objeto devem obedecer à(s) regra(s) 1 e/ou 2.

O uso de coleções e mapas é permitido desde que se use *generics* e os tipos que elas encapsulem obedeçam às regras citadas anteriormente. Assim é possível se ter por exemplo uma *List*<*String*> trafegando entre o cliente e o servidor.

Abaixo você encontra o diagrama de implementação de um serviço rpc do GWT.



- **Utilização Request Factory**
  - Una chamada de procedimento remoto mais robusta, que da um tratamento mais completo a entidades, produz DTOs automaticamente, permite que varias chamadas a procedimentos sejam unidos em uma única requisição.
- **Utilização JSON**
  - Chamada que obtém resposta json *with Padding*. Por ser uma resposta java script normalmente utilizada jSNI
  - É a única forma de fazer requisições para outro domínio
  - Muitos desenvolvedores de aplicativos AJAX adotaram JSON como formato de dados de escolha para a comunicação do servidor. É um formato relativamente simples, baseada na notação de objeto literal do *JavaScript*. Se você optar por usar os dados codificados em JSON dentro de sua aplicação, o GWT contém classes que você pode usar para analisar e manipular objetos JSON.
  - codificação JSON

    O formato JSON baseia-se nos dados de sintaxe e tipos de linguagem *JavaScript*. Ele suporta *strings*, números, booleanos, e valores nulos. Você também pode combinar vários valores em *arrays* e objetos. Objetos JSON são simplesmente conjuntos não ordenadas de pares nome / valor, onde o nome é sempre uma *string* e o valor é de qualquer outro tipo JSON válido (até mesmo outro objeto). Aqui está um exemplo de codificação de dados de produtos em JSON:

```
{
 "product": {
          "name": "Widget",
           "company": "ACME, Inc",
           "partNumber": "7402-129",
           "prices": [
                    { "minQty": 1, "price": 12.49 },
                     { "minQty": 10, "price": 9.99 },
                     { "minQty": 50, "price": 7.99 }
           ]
 }
}
```

Imagem- modelo estrutura JSON

CONCLUSÃO

Neste artigo foi feito uma breve introdução à tecnologia GWT, suas principais características e as vantagens e desvantagens. Há algum tempo houve uma mudança significativa na forma como interagirmos com aplicações web. Uma nova tendência, denominada web 2.0, tem mudado a concepção sobre o paradigma *request/wait/response,* usado na maioria dos sites tradicionais. Atualmente, criar aplicativos para a web é um processo tedioso e com alta incidência de erros. Os desenvolvedores podem passar 90% do tempo trabalhando para contornar peculiaridades do navegador. Além disso, a criação, a reutilização e a manutenção de grandes bases de código *JavaScript* e componentes AJAX pode ser difícil e delicada. Observamos que para quem já desenvolve para Web o GWT é um modo de trabalho que veio para ficar pois além de utilizar Frameworks facilita e muito na criação além de que o resultado final é muito mais agradável. Todavia novos Frameworks ainda estão surgindo para resolver outros problemas, entretanto o código produzido por GWT é compatível com os browsers mais utilizados e código HTML+CSS+*Javascript* que funcione bem em todos os browsers não é uma tarefa das mais triviais. O Google Web Toolkit é uma plataforma muito interessante e sem dúvida será cada vez mais requisitada onde é possível aplicar o framework GWT para escrever o seu *frontend* completo em Java, pois você sabe que é possível deixar que o compilador do GWT converta as classes do Java para *JavaScript* e HTML compatíveis com browsers.

REFERENCIAS

PAULO FERNANDES JR. Disponível em: <http://javafree.uol.com.br/artigo/874221/Iniciando-com-o-GWT.html>. Acessado em 25 de Março de 2014.

PAJÉ ONLINE. Disponível em:<http://pajeonline.blogspot.com.br/2008/09/lanado-o-gwt-15.html> Acessado em 25 de Março de 2014.

JAVAFREE.ORG. Disponível em:<http://javafree.uol.com.br/artigo/860307/GWT-The-ServerSide.htm>. Acessado em 25 de Abril de 2014.

BR-LINUX.ORG. Disponível em: <http://br-linux.org/wparchive/2008/novidade-do-google-lancamento-do-gwt-15.php>. Acessado em 30 de Março de 2014

HUGO CARTELETTI E LORRAN PENGORETTI. Disponível em: <http://pt.slideshare.net/lorran33/seminrio-gwt>. Acessado em 30 de Março de 2014.

UNIVERSIDADE FEDERAL DE CAMPINA GRANDE DEPARTAMENTO DE SISTEMAS E COMPUTAÇÃO. Disponível em: <http://www.dsc.ufcg.edu.br/~pet/ciclo_seminarios/tecnicos/2010/GWT.pdf> Acesso em: 07 de Abril de 2014

GWT DOCUMENTAÇÃO. Disponível em: <http://www.gwtproject.org/doc/latest/tutorial/clientserver.html> Acesso em 07 de Abril de 2014.

DEVE MEDIA. Artigo Java Magazine 39 - AJAX Avançado com GWT. Disponível em<http://www.devmedia.com.br/artigo-java-magazine-39-ajax-avancado-com-gwt/8714>. Acesso: 07 de Abril de 2014.

FERNADES, Paulo. Iniciando com GWT. Disponível em: <http://www.phpaulo.com.br/tag/gwt/> Acesso em: 10 Abril. 2014.

DINIZ, Vinícius. GWT Visão Geral. Disponível em: <http://loogica.wordpress.com/2007/01/04/gwt-parte-1-visao-geral/> Acesso em: 10 Abril 2014.

FREE, Java. GWT. Disponível em: <http://javafree.uol.com.br/artigo/874221/Iniciando-com-o-GWT.html/> Acesso em: 10 Abril. 2014.