

Universidade Estadual de Maringá  
Centro de Tecnologia – Departamento de Informática  
Especialização em Desenvolvimento de Sistemas para a *Web*

**Estudo comparativo entre tecnologias para desenvolvimento  
Web**

Glenn William Rodrigues Barbosa

Prof. Marco Aurélio Lopes Barbosa  
Orientador

Prof. Alysson Neves Bessani  
Co-Orientador

Maringá, 2007

Universidade Estadual de Maringá  
Centro de Tecnologia – Departamento de Informática  
Especialização em Desenvolvimento de Sistemas para a *Web*

Glenn William Rodrigues Barbosa

**Estudo comparativo entre tecnologias para desenvolvimento  
Web**

Trabalho submetido à Universidade Estadual de Maringá  
como requisito para obtenção do título de Especialista  
de Desenvolvimento de Sistemas para a *Web*.

**Salmos 23:4** Ainda que eu ande pelo vale da sombra da morte, não temerei mal nenhum, porque tu estás comigo; o teu bordão e o teu cajado me consolam.

## Agradecimentos

Agradeço a ajuda dos meus companheiros de jornada que sem eles eu não teria conseguido chegar ao fim.

Aos meus professores pelo tempo e conhecimento.

Aos meus entes queridos, por vocês.

## **Resumo**

Este trabalho apresenta uma visão geral sobre tipos de interfaces e suas aplicações, abordando as interfaces *desktop*, que oferecem inúmeros recursos para a interação entre usuário e sistema, e as interfaces baseadas na Web, que buscam distribuir aplicações Web para os usuários com a mesma ou melhor qualidade que uma interface *desktop*.

É abordado o que vem a ser Web 2.0 e o conceito de RIA, juntamente das tecnologias que fazem essa idéia possível.

O foco principal dessa pesquisa foi as tecnologias AJAX e JWS visando a possibilidade de fazer uma comparação entre ambas.

Palavras-chave: RIA, AJAX, JWS, Web 2.0

## **Abstract**

This work presents a general vision on types of interfaces and its applications, approaching the interfaces desktop, that they offer to innumerable resources for the iteration between user and system, and the interfaces based on the Web, that they search to distribute Web applications for the users with the same or better quality that an interface desktop.

He is boarded what it comes to be Web 2,0 and the concept of RIA, together of the technologies that make this possible idea.

The main focus of this research was the technologies AJAX and JWS aiming at the possibility to make a comparison between both.

Word-key: RIA, AJAX, JWS, Web 2.0

## Lista de Figuras

Figura 1 - Arquitetura Web em 3 camadas.....	7
Figura 2- Arquitetura de uma aplicação RIA .....	20
Figura 3- Página para aplicação exemplo Notepad .....	25
Figura 4 - Aplicação Notepad.....	25
Figura 5 - Modelo tradicional de aplicações web comparado com o modelo AJAX.....	28
Figura 6 - Diagrama de Casos de Uso .....	36
Figura 7 - Diagrama de Classes.....	37
Figura 8 - Arquitetura da Implementação em AJAX .....	39

## **Lista de Tabelas**

Tabela 1 – Cronograma.....	5
Tabela 2 - Comparativo entre aplicações Web 1.0 e Web 2.0.....	16
Tabela 3 - Lista de utilidade de aplicações RIA. (DUHL, 2003).....	22

## Sumário

1 - Introdução .....	1
1.1 Objetivo .....	1
1.2 Definição do problema .....	2
1.3 Justificativa.....	2
1.4 Motivação .....	3
1.5 Importância do Tema.....	3
1.6 Limitações da pesquisa.....	3
1.7 Organização do Trabalho.....	3
1.8 Metodologia de desenvolvimento da pesquisa .....	4
2. Interfaces e Aplicações .....	6
2.1 Aplicações Web .....	6
2.2 Aplicações Desktop .....	12
2.3 Comparação entre interfaces Web e Desktop.....	14
3 WEB 2.0 .....	16
4. RIA – Rich Internet Application .....	17
4.1 Arquitetura.....	20
4.2 Utilidade .....	21
4.3 Tecnologias.....	22
5. Java Web Start.....	24
5.1 Como funciona o JWS – Perspectiva do Usuário.....	24
5.2 JWS – Perspectiva do Desenvolvedor .....	26
6 AJAX.....	27
6.1 O que é AJAX?.....	27
6.2 Comparação entre o modelo clássico de aplicações web e o modelo AJAX .....	28
6.3 Como funciona uma aplicação Web com AJAX.....	<b>Erro! Indicador não definido.</b>
6.4 Casos de sucesso com AJAX .....	31
7. MVC .....	33
7.1 Funcionamento MVC .....	33
8. Estudo de Caso .....	35
8.1 Estrutura e Descrição da aplicação.....	35
8.2 Requisitos do sistema .....	35
8.3 Atores .....	35
8.4 Casos de Uso .....	36
8.5 Diagrama de Classe da Implementação em Java.....	37
8.6 Implementação em AJAX .....	39
8.7 Infra-Estrutura utilizada.....	40
9 – Resultados e Observações .....	41
9.1 Curva de aprendizado .....	41
9.2 Limitações .....	41
9.3 Carregamento .....	42
9.4 Comparação Geral .....	42
10. Considerações Finais .....	43
Referências bibliográficas .....	44

# 1 - Introdução

Interação, esta é uma das palavras mais mencionadas quando se fala em desenvolvimento de IHC (Interface Humano-Computador). Como o usuário irá se relacionar com o software.

Atualmente quando se fala em desenvolvimento Web logo se pensa no padrão MVC (Model-View-Controller, Modelo-Visão-Controlle), onde as partes do sistema ficam bem definidas facilitando sua manutenção. O maior problema está na fase de desenvolvimento da interface, onde o tempo gasto é maior que nas outras fases do projeto. Existem diversos métodos para facilitar a criação de tais interfaces, que mesmo assim tornam a manutenção e o reuso do código tarefas difíceis e complicadas de se fazer.

## 1.1 *Objetivo*

### 1.1.1 Geral

Realizar o estudo das tecnologias AJAX e JWS (Java Web Start), estabelecendo os prós e contras para o desenvolvimento, manutenção e reuso da camada de apresentação.

### 1.1.2 *Objetivos Específicos*

Dentro deste objetivo principal, podemos destacar vários objetivos específicos:

- Realizar um estudo sobre interfaces de usuário, focando nas Interfaces de Usuário baseados na Web – WUI (*Web User Interface*);
- Descrever Web 2.0, metodologias e padrão de desenvolvimento;

- Conceituar RIA (*Rich Internet Application*), conjunto de metodologias para o desenvolvimento de interfaces de usuário para aplicações web;
- Realizar estudo sobre AJAX, seus benefícios e dificuldades.
- Estudar a API *Swing* de Java e a forma de distribuição de uma aplicação utilização JWS.

## **1.2 Definição do problema**

Quando se trata de desenvolvimento web, logo é pensado em uma aplicação que necessita de um *web browser* para ser executada. Mesmo com tantos avanços os browsers apresentam algumas limitações inerentes as suas características.

É possível desenvolver aplicações para o contexto web sem que tais limitações tornem um transtorno aos analistas e programadores. Caso seja, tal solução é viável ao projeto?

## **1.3 Justificativa**

Existe um grande debate a respeito de qual é a melhor tecnologia para o desenvolvimento de aplicações para a Internet. Muitos sistemas carecem de formas de interação mais poderosas que as providas pelo modelo cliente/servidor suportado pelo *framework* geral de desenvolvimento para a Web (baseado em HTTP/HTML).

Perante isso, surge a pergunta: por que não desenvolver uma mesma interface gráfica semelhante a um sistema desktop que possa ser disponibilizada via Web (através de browsers)? Este tipo de interface, além de oferecer uma interação muito mais amigável com o usuário, permitiria o acesso a aplicação de qualquer lugar onde houvesse acesso a Internet.

Este trabalho pretende comparar o uso de tecnologia Java (baseada em *Swing* e *Java Web Start*) e AJAX para a construção do tipo de interface rica descrita (Interface com poder de desktop, porém disponibilizada na web). Estas duas tecnologias se mostram bastante atraentes pela facilidade de uso e grande suporte por parte dos browsers, além da grande penetração no mercado.

## **1.4 Motivação**

A motivação vem do enorme leque de tecnologias disponíveis para a resolução do problema em questão e da falta de trabalhos acadêmicos que as comparam.

## **1.5 Importância do Tema**

A Internet é atualmente a forma de comunicação mais utilizada, todos os dias milhares de pessoas tem acesso a *Websites* em busca de algum tipo de informação, tornar esses *Websites* mais amigáveis e atrativos é tarefa árdua que requer muitos estudos e conhecimento sobre diversas tecnologias diferentes.

## **1.6 Limitações da pesquisa**

Falta de conhecimento de algumas tecnologias estudadas para se desenvolver os protótipos propostos. Tornando o estudo comparativo difícil e incompleto.

## **1.7 Organização do Trabalho**

Ele está dividido em 9 capítulos onde, Capítulo 1, Introdução, é abordado as informações gerais sobre o tema, qual o objetivo e a metodologia utilizada neste trabalho. Capítulo 2, Interfaces a Aplicações, é feito um panorama sobre os tipos de interface para aplicações Web e aplicações tradicionais de Desktop. Capítulo 3, Web 2.0, o tema principal do trabalho aborda tecnologias para o desenvolvimento de aplicações que seguem o novo conceito da Web, portanto este capítulo se fez necessário. Capítulo 4, RIA – *Rich Internet Application*, é explanado o que vem a ser RIA, informações gerais e conceitos, quais as tecnologias mais conhecidas sua utilidade e funcionamento. Capítulo 5, JWS – *Java Web Start*, é apresentado do que se trata o conceito de JWS, seu funcionamento e características. Capítulo 6, AJAX, apresentada seus conceitos, quais as tecnologias que a compreendem e como se dá o desenvolvimento utilizando este conceito. Capítulo 7, Caso

de Uso, são descritas as tecnologias empregadas para o desenvolvimento das aplicações de exemplo, a estruturas das aplicações juntamente com a descrição e alguns diagramas usadas para explicar seu funcionamento. Capítulo 8, MVC, visão geral sobre o padrão de projeto MVC. Capítulo 9, Resultados, é feito um comparativo sobre as duas tecnologias utilizadas para o desenvolvimento dos protótipos, dando ênfase na implementação das interfaces com o usuário. Capítulo 10, Considerações Finais, conclusões obtidas no termino do trabalho e passos futuros a serem seguidos.

## ***1.8 Metodologia de desenvolvimento da pesquisa***

Este trabalho utilizou pesquisas em bibliografias existentes e também em matérias disponíveis na Internet, como por exemplo, livros digitais (*E-Books*), publicações em *Websites*, sendo essas bibliografias e fontes de informação na Internet, relacionadas a temas específicos abordados no decorrer do trabalho.

### **1.8.1 Etapas da Pesquisa:**

Etapa 1 - Levantamento Bibliográfico: nesta primeira fase foi realizado o levantamento de trabalhos, artigos, livros, e outros materiais referentes ao tema. Feito isso foi realizado um estudo mais detalhado, com o objetivo de extrair o máximo do conhecimento.

Etapa 2 - Estudos das tecnologias empregadas para o desenvolvimento do projeto;

Etapa 3 - Projeto do sistema exemplo a ser desenvolvido.

Etapa 4 - Desenvolvimento das duas versões do sistema exemplo.

Etapa 5 - Testes comparativos.

Etapa 6 - Escrita da monografia.

ETAPAS	J U N	J U L	A G O	S E T	O U T	N O V	D E Z	J A N
Etapa 1								
Etapa 2								
Etapa 3								
Etapa 4								
Etapa 5								
Etapa 6								

**Tabela 1 - Cronograma**

## 2. Interfaces e Aplicações

Segundo (MARCHAL, 1998), a evolução do padrão de interface podem ser divididos em 3 gerações:

1ª Geração: Interfaces baseadas em caracteres, utilizada em terminais e estações de trabalho, ex.: MS-DOS e Unix.

2ª Geração: Interfaces orientadas a janelas, sistemas que levam em consideração o fator humano, ex.: Windows.

3ª Geração: Aplicações via *web browsers*, *E-commerce*, *home banking*, sistemas corporativos via Intranet, são alguns exemplos mais comuns atualmente, de aplicações complexas para a web

### 2.1 Aplicações Web

Na engenharia de software, uma aplicação web, algumas vezes chamada de *webapp*, são aplicações que podem ser acessadas através de um *web browser*, sobre uma rede tal como a Internet ou uma intranet.

Trata-se de um conjunto de programas que são executados em um servidor HTTP (*Web Host*). O desenvolvimento da tecnologia web está relacionado, entre outros fatores, à necessidade de simplificar a atualização e manutenção mantendo o código fonte em um mesmo local, de onde ele é acessado pelos diferentes usuários.

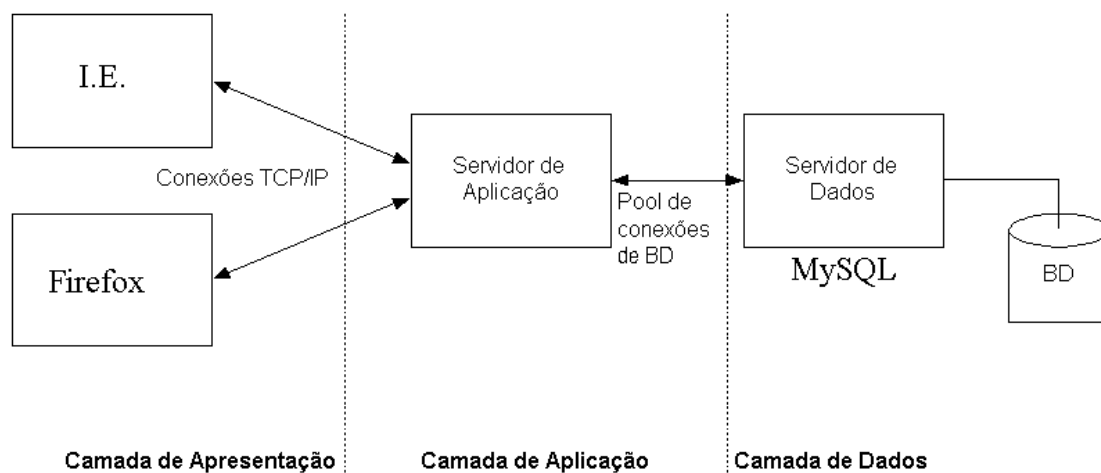
A habilidade de atualizar e manter aplicações web sem distribuir e instalar o software potencial em milhares de computadores do cliente é uma razão chave para sua popularidade.

Uma vantagem significativa de construir aplicações web de acordo com os padrões do navegador é que devem executar como especificado não importando o tipo nem a versão do sistema operacional instalado do lado do cliente. Melhor que criar clientes para Mac OS X, Windows, GNU/Linux, e outros sistemas operacionais, a aplicação pode ser escrita uma vez e executada quase em qualquer lugar.

Entretanto, implementações inconsistentes do HTML, CSS, DOM e das outras tecnologias usadas no navegador podem causar problemas no desenvolvimento e no suporte de aplicações web. Adicionalmente, a habilidade dos usuários de personalizarem muitos dos ajustes da exposição de seu browser (tais como selecionar tamanhos, cores e tipos diferentes de fontes, ou desabilitar suporte a *scripts*) pode interferir com a execução de uma aplicação web.

### 2.1.1 Estrutura de uma aplicação web

Embora, seja possível muita variação, uma aplicação web é normalmente estruturada como uma aplicação de três camadas como ilustrado na Figura 1. Em seu formato mais comum, o navegador é a primeira camada, uma tecnologia web dinâmica (por exemplo, JSP, PHP, ASP) é a camada intermediária, e a base de dados é a terceira camada. O navegador emite pedidos a camada intermediária, que lhes presta serviços de manutenção fazendo perguntas e atualizações de encontro à base de dados e gerando uma interface de usuário.



**Figura 1 - Arquitetura Web em 3 camadas**

As interfaces web são usadas cada vez mais em aplicações originalmente planejadas como aplicações tradicionais, *single-user*. Por exemplo, a ajuda em HTML da

Microsoft substituiu a ajuda do Windows como o sistema de ajuda preliminar do Microsoft Windows.

### **2.1.2 WUI - Interface de Usuário baseadas na Web**

As interfaces de usuário baseadas na web aceitam a entrada e fornecem a saída gerando as páginas Web que são transportados através da Internet e vistas pelo usuário através de web browser. Algumas implementações utilizam Java, AJAX, Microsoft .NET, ou tecnologias similares para fornecer um controle mais dinâmico, eliminando a necessidade de atualização que uma aplicação tradicional baseada em HTML necessita. (Wikipedia, 2006)

Os desenvolvedores web usam freqüentemente scripts do lado do cliente para adicionar funcionalidades de desktop, visando especialmente criar uma experiência interativa sem que seja necessária a recarga da página. Recentemente, algumas tecnologias foram desenvolvidas para coordenar os scripts do lado cliente com tecnologias do lado do servidor tais como JSP. Um Exemplo de tecnologia desse tipo é o Ajax.

### **2.1.3 Tecnologias para a construção de WUI**

#### **2.1.3.1 HTML**

HTML (HyperText Markup Language - Linguagem de Formatação de Hipertexto) é fruto do "casamento" dos padrões HyTime e SGML.

- HyTime - Hypermedia/Time-based Document Structuring Language(ISO 10744:1992) - padrão para representação estruturada de hipermídia e informação baseada em tempo. Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (áudio, vídeo, etc.), conectados por webs ou hiperlinks. O padrão HyTime é independente dos padrões de processamento de texto em geral. Ele fornece a base para a construção de sistemas hipertexto

padronizados, consistindo de documentos que alicam os padrões de maneira particular.

- SGML - Standard Generalized Markup Language. Padrão ISO 8879 de formatação de textos: não foi desenvolvido para hipertexto, mas torna-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações. SGML não é padrão aplicado de maneira padronizada: todos os produtos SGML têm seu próprio sistema para traduzir as etiquetas para um particular formatador de texto.
- DTD - Document Type Definition - define as regras de formatação para uma dada classe de documentos. Um DTD ou uma referência para um DTD deve estar contido em qualquer documento conforme o padrão SGML. Portanto, HTML é definido segundo um DTD de SGML.

Todo documento HTML apresenta elementos entre parênteses angulares (< e >); esses elementos são as etiquetas (tags) de HTML, que são os comandos de formatação da linguagem. A maioria das etiquetas tem sua correspondente de fechamento:

```
<etiqueta>...</etiqueta>
```

Isso é necessário porque as etiquetas servem para definir a formatação de uma porção de texto, e assim marcamos onde começa e termina o texto com a formatação especificada por ela.

### **2.1.3.2 JavaScript**

O JavaScript é uma linguagem pequena, leve, orientada por objetos e multi-plataforma. Apesar de não ser útil como uma linguagem autônoma, foi desenhada para ser fácil o seu embutimento em outros produtos e aplicações, como navegadores web. Dentro de um ambiente-hospedeiro, a JavaScript pode ser ligada aos objetos desse ambiente para providenciar controle programático sobre eles. (Mozzila, 2006)

### 2.1.3.3 DHTML

DHTML não é um padrão da *World Wide Web Consortium* (W3C). DHTML é um "termo de marketing" – usado pela Netscape e pela Microsoft para descrever as tecnologias de quarta geração suporta pelos *browser*.

DHTML é a combinação de tecnologias usadas para criar Web sites dinâmicos.

Para muitos o DHTML significa uma combinação de HTML 4.0, *Style Sheets* e JavaScript.

Segundo a W3C: “*Dynamic HTML* é um termo usado por alguns fabricantes para descrever a combinação de HTML, *style sheets* e scripts que permita criar documentos animados”. (W3SCHOOLS)

Dynamic HTML, ou DHTML é a união das tecnologias HTML, Javascript e uma linguagem de apresentação, como folhas de estilo CSS aliada a um Modelo de Objeto de Documentos, para permitir que uma página Web seja modificada dinamicamente na própria máquina cliente, sem necessidade de novos acessos ao servidor web.

Atualmente há pelo menos três grandes grupos trabalhando no DHTML: o W3C, que é o responsável pelas versões oficiais da HTML, a Netscape, e a Microsoft.

Em relação à especificação oficial, a Netscape e a Microsoft incluíram vários recursos extras que ajudaram ainda mais o DHTML. Sendo assim, esses DHTMLs vão algo além da capacidade de alterar as propriedades das marcações tags HTML dinamicamente. O DHTML da Microsoft, por exemplo, permite que se adicionem efeitos como sombra e néon a imagens dentro de um documento HTML. Outro recurso conhecido como Fontes Dinâmicas (Dynamic Fonts), da Netscape, permite que fontes sejam transmitidas pelo servidor junto com o documento HTML, possibilitando, a qualquer browser que implemente esse recurso, mostrar os caracteres exatamente como planejou o autor do documento. Resumindo, DHTML é um conjunto de ingredientes que proporcionam um controle sem precedentes sobre a apresentação do conteúdo de páginas da Web, além de possibilitar a inclusão de componentes multimídia, como animações, diretamente no código HTML, sem a necessidade de plug-ins. Toda a curiosidade em torno do HTML Dinâmico se justifica então, já que o mesmo estabelece novos patamares de

interação e movimento na internet a um baixo custo - medido em velocidade, abrangência e flexibilidade, por exemplo, melhor do que em moeda corrente. (Wikipedia, 2006)

#### **2.1.3.4 XML**

XML (*eXtensible Markup Language*) é uma recomendação da W3C para gerar linguagens de marcação para necessidades especiais.

É um subtipo de SGML (acrônimo de *Standard Generalized Markup Language*, ou Linguagem Padronizada de Marcação Genérica) capaz de descrever diversos tipos de dados. Seu propósito principal é a facilidade de compartilhamento de informações através da Internet. Entre linguagens baseadas em XML incluem-se XHTML (formato para páginas Web), RDF, SMIL, MathML (formato para expressões matemáticas), NCL, XBRL, XSIL e SVG (formato gráfico vetorial). (Wikipedia, 2006)

XML é um formato de texto simples e muito flexível derivado da SGML (ISO 8879). Projetada originalmente para encontrar mudanças em publicações eletrônicas de larga escala.

#### **2.1.3.5 SWF**

SWF é um formato de arquivos de gráfico vetorial produzido pelo software Flash da Adobe (anteriormente da Macromedia). Pretendeu ser pequeno o bastante para publicações na Web, os arquivos SWF podem conter animações ou applets apresentando vários graus de interação e funções. SWF é usado em alguns casos para criar animações gráficas e menus em DVD de filme , e comerciais de televisão.

O programa Flash produz arquivos SWF comprimidos e não editáveis, visto que usa o formato .fla para editar os trabalhos. (Wikipedia, 2006)

## **2.2 Aplicações Desktop**

As aplicações desktop se caracterizam por trazer toda a lógica e definição de interface para a máquina onde seu usuário interagirá com ela. Isso significa que, havendo vários usuários para uma aplicação, cada uma em máquina própria, deve haver também uma instalação própria da aplicação desktop para cada máquina em uso. E o que for feito em uma dessas instalações não poderá ser compartilhado com as outras. Temos um sistema com lógica e layout em um só lugar, isolado.

Aplicações desktop apresentam uma grande variedade de componentes para se compor um layout mais rico e interativo. Por se tratar de uma aplicação que está instalada diretamente no computador, o usuário tem a sensação de maior segurança e agilidade.

### **2.2.1 Estrutura de uma aplicação desktop**

Aplicações em rede para usuários finais podem ter dois tipos de estrutura: cliente-servidor<sup>1</sup> e ponto-a-ponto<sup>2</sup>. Nos dois casos, temos um acréscimo de complexidade para o tratamento da comunicação por rede e a troca de dados. Também há a necessidade de se implementar um servidor. Para aplicações cliente-servidor, o servidor será um novo programa, enquanto que para aplicações ponto-a-ponto, o servidor será parte do programa com que o usuário interagirá.

Tanto para aplicações em rede como para aplicações locais, temos uma instalação para cada máquina onde o sistema deverá ser utilizado, o que dificulta a atualização em grandes parques de máquinas.

---

<sup>1</sup> Arquitetura cliente-servidor

<sup>2</sup> Arquitetura ponto-a-ponto

### 2.2.2 GUI - Interfaces gráficas com o Usuário

Uma GUI (*Graphical user interfaces* – Interface gráfica com o usuário) aceita a entrada através dos dispositivos tais como o teclado e o mouse de computador e fornecem a saída através de um monitor de vídeo. Há pelo menos dois princípios diferentes usados extensamente no projeto do GUI: interfaces orientadas a objeto e as orientadas a aplicação (Wikipedia, 2006)

Na área de interfaces com o usuário, um ambiente de desktop é um sistema que oferece soluções com interface gráfica utilizando um conceito de design denominado GUI (*Graphic User Interface*) que trouxe de forma adjacente a preocupação com fatores relevantes ao usuário, o que proporcionou um aumento nas pesquisas de estudos referentes à interface, como uso de cores, sons, padronização de layouts e ergonomia.

Um ambiente de desktop fornece ícones, pastas, barra de ferramentas, *applets*, papéis de parede e habilidades como arrastar e colar. Como um todo, as particularidades de projeto e funcionalidade de um ambiente de desktop lhe dão uma aparência de uso, ou *look and feel*, distintivo.

## 2.3 Comparação entre interfaces Web e Desktop

Aplicações Web e Desktop, ambas apresentam algumas vantagens e desvantagens em comparação uma com a outra.

Algumas vantagens das aplicações Web são:

- Interface HTML reconhecida por uma grande gama de usuários já acostumados com o funcionamento dos navegadores.
- Desenvolvimento, manutenção e atualização centralizada da aplicação. Não é necessário instalar a aplicação em diversos equipamentos diferentes. Basta colocá-lo no servidor para que os usuários a acessem<sup>3</sup>.
- A exportação de dados entre usuários remotos usando o protocolo HTTP é muito mais fácil do que em aplicações desktop.
- Escalabilidade no processamento. Se houver necessidade de aumentar o poder de processamento, basta fazer isto no servidor.

Desvantagens de aplicações Web:

- A interface HTML pode ser um problema, pois não há uma padronização entre os diversos navegadores e sua aplicação poderia ser exibida de uma maneira diferente dependendo do navegador.
- A entrada de uma grande massa de dados é prejudicada na interface HTML, pois não existe uma maneira padrão de criar máscaras de entrada de dados.
- A interface HTML não é rica em controles gráficos e peca no quesito posicionamento. O visual da aplicação pode não ficar tão elegante como você imagina.
- A integração com outros componentes não é tão fácil com HTML.

Algumas vantagens encontradas nas aplicações *Desktop*:

---

<sup>3</sup> Algumas tecnologias, como o VB.NET, requerem a distribuição e instalação do .NET *Framework* para as máquinas clientes.

- Uma rica variedade de controles para interface com o usuário.
- Um total controle sobre o posicionamento dos controles na aplicação.
- O desempenho da interface gráfica é melhor em uma aplicação desktop, já que usa processamento local.
- A interface tem suporte a vários dispositivos de entrada.

Desvantagens das aplicações *Desktop*:

- Uma interface gráfica muito carregada deixa a aplicação mais pesada.
- A integração com usuários remotos é mais difícil.
- A distribuição da aplicação é crítica. A aplicação tem de ser instalada na máquina de todos os usuários, que podem ter diferentes tipos de sistemas operacionais.
- A manutenção e atualização da aplicação requerem um esforço extra.

### 3 WEB 2.0

O termo surgiu durante uma sessão de *brainstorming* entre representantes da O'reilly Media e da Media Internacional, em 2004. Segundo os autores do termo, a expressão servia para denotar um conjunto interessante e rico de aplicações e sites que estavam surgindo naquela época com características comuns. Em outubro do mesmo ano, a O'reilly organizava a primeira conferência a respeito do tema a Web 2.0 Conference.

Desde a sua primeira menção, o termo tem causado polêmica e divisões entre grupos que acreditam que o mesmo está relacionado a manobras puramente ligadas a marketing da própria O'reilly e um segundo grupo que defende que de fato a web estaria passando por um momento de transformação, justificando assim a presença da expressão "2.0", da mesma forma que em versões de software.

Durante a sessão da conferência entre O'Reilly e MediaLive Internacional, foi formulada uma interpretação de Web 2.0 através de um comparativo:

Web 1.0	Web 2.0
DoubleClick	Google AdSense
Ofoto	Flickr
Akamai	BitTorrent
Mp3.com	Napster
Britannica OnLine	Wikipedia
personal websites	blogging
evite	upcoming.org and EVDB
domain name speculation	search engine optimization
page views	cost per click
screen scraping	web services
publishing	Participation
content management systems	Wikis
directories (taxonomy)	tagging ("folksonomy")
stickiness	syndication

**Tabela 2 - Comparativo entre aplicações Web 1.0 e Web 2.0**

Segundo (CABRERA, 2006) como muitos conceitos importantes, a Web 2.0 não possui uma delimitação.

## 4. RIA – Rich Internet Application

Aplicações RIA (*Rich Internet Application*) são aplicações implementadas no servidor que utilizam a tecnologia *Rich Client* e que tiram vantagem da tecnologia do lado cliente para prover uma nova classe de *Websites* interativos e dinâmicos com a sofisticação de aplicações *desktop*.

Além a oferecer uma ampla variedade de controles (*sliders, date pickers, Windows, abas, spinners gauges*, e assim por diante), RIAs permite geralmente que você construa gráficos perfeitos com qualquer SGQ (*Scalable Vector Graphics*) ou algum outro mecanismo. Algumas tecnologias RIA podem prover animações completas em resposta a mudanças nos dados. (O'ROURKE, 2004).

Segundo (MACROMEDIA, 2002), esses aplicativos utilizam-se de tecnologia *Rich Client*, ampliando a capacidade da Internet de modo a propiciar experiências mais intuitivas, mais interativas e mais produtivas para o usuário. Além de oferecer a mesma qualidade que o usuário observa com aplicativos *desktop*, oferecem flexibilidade de melhor disponibilização.

Segundo (NODA, HELWING, 2005) aplicações web baseadas no browser são muitas vezes apresentadas desta forma:

- *Tags/Scripts* padronizados facilitam o desenvolvimento (Desenvolvimento rápido e baixo custo);
- Não necessita de instalação, atualizações são necessárias<sup>4</sup> (Baixo custo de manutenção);
- Aplicações são acessíveis através de rede de computadores (disponibilidade e flexibilidade);

---

<sup>4</sup> Normalmente é necessário que seja feita a atualização de plug-in's para que as aplicações baseadas no *browser* executem corretamente.

- Aplicações podem executar em diferentes sistemas operacionais (Independência de plataforma);
- Interfaces com o usuário simples e padronizado (Baixa curva de aprendizado do usuário).

Ainda segundo (NODA, HELWING, 2005) aplicações desktop apresentam as seguintes vantagens perante aplicações baseadas na web:

- *Richer user experiences* (Áudio, vídeo, comunicação);
- Não precisa recarregar as páginas;
- Suportam operação tanto *on-line* como *off-line*;
- Aplicações mais complexas;
- Melhores respostas e mais interativo.

A tecnologia *Rich Client* fornece um ambiente dinâmico, com capacidade de hospedagem de aplicativos compilados no lado do servidor recebidos como arquivos através de HTTP. Os aplicativos no lado do cliente conectam-se de volta aos back-ends de servidores de aplicativos existentes, por meio de uma arquitetura assíncrona de cliente/servidor que oferece segurança, escalabilidade e que é bem adaptada ao novo modelo orientado a serviços que vem ganhando grande atenção recentemente.

A adoção crescente da tecnologia *Rich Client* não é uma etapa evolutiva de substituição a HTML. Consiste mais em uma ampliação da capacidade de suportar interfaces de usuário mais eficazes e dinâmicas, dos browsers e dispositivos<sup>5</sup>.

A maioria dos aplicativos *Rich Client* é executada no contexto de browsers, e muitos são executados dentro das páginas, junto com o conteúdo HTML. Estes aplicativos acrescentam mais recursos à Internet, mas a linguagem HTML continuará a ter um papel fundamental na disponibilização de conteúdo, nas interfaces de usuário e na navegação.

---

<sup>5</sup> Atualmente existe uma série de dispositivos que possuem a capacidade de acessar a Internet, como celulares, palmtops, etc.

Como a tecnologia *Rich Client* pode ser executada tanto em browsers como em dispositivos, ela possibilita criar aplicativos que podem ser disponibilizados uniformemente em uma ampla gama de plataformas com conexão à Internet. Além disso, como a tecnologia *Rich Client* possibilita o uso de elementos gráficos móveis, vídeo, áudio, comunicação bidirecional e formulários complexos, ela constitui um ambiente significativamente mais sólido para a criação de interfaces de usuário de aplicativos.

Segundo (Macromedia, 2002), são necessárias três tecnologias para a disponibilização de aplicativos Web com interfaces ricas: tecnologia *Rich Client*, servidores e ferramentas de desenvolvimento:

**Tecnologia Rich Client:** A tecnologia *Rich Client* fornece a capacidade do lado do cliente, que possibilita o uso de aplicativos "interfaces ricas" aproveitando da melhor forma possível a potência de processamento local, isto é, dos computadores e dispositivos do usuário. Os dois fatores fundamentais na escolha de uma tecnologia *Rich Client* são o seu grau de adoção e a sua capacidade. (Macromedia, 2002)

**Tecnologia de servidor:** Para estabelecer a conexão dos *Rich Client* aos dados e software dos aplicativos, é necessária uma nova tecnologia, que forneça um ambiente veloz de criação de script, integração empresarial, conectividade a cliente, e compatibilidade com os principais padrões do setor. Além dos recursos fornecidos pelos aplicativos comuns de bancos de dados, os aplicativos "interfaces ricas" têm o potencial de incorporar comunicação bidirecional e dados em tempo real, necessitando, para isso, de uma nova geração de recursos de servidores de comunicação. (Macromedia, 2002)

**Ferramentas de desenvolvimento:** A tecnologia de cliente e de servidor não significa nada sem um conjunto de ferramentas poderosas e simples de usar que permitam aos desenvolvedores começar a trabalhar sem demora e disponibilizar soluções avançadas. Devido à sua arquitetura de cliente/servidor, os aplicativos "interfaces ricas" necessitam dispor de uma variedade de ferramentas de desenvolvimento que funcionem em conjunto. (Macromedia, 2002)

## 4.1 Arquitetura

A figura 2 (O'ROURKE, 2004) ilustra a arquitetura típica para uma RIA.

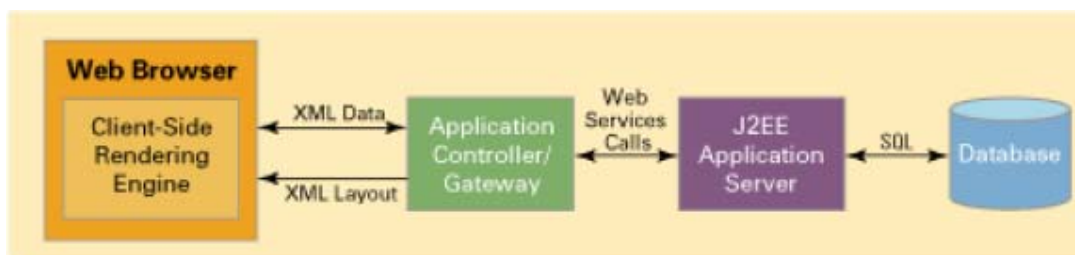


Figura 2- Arquitetura de uma aplicação RIA

No lado do cliente, este deve possuir um *Web Browser* que contenha algum *plug-in* necessário para interpretar os dados que chegam da aplicação hospedada no servidor, onde esses dados podem ser arquivos de vídeo, textos, áudio, entre outros tipos de arquivos. Esta parte da arquitetura se diferencia das aplicações Web tradicionais, onde se armazena a parte gráfica da aplicação (esquema XML, ou binário, por exemplo, SWF), com isto evitam-se as sucessivas renderizações a cada comunicação com o servidor, para a exibição dos dados.

Essa arquitetura também possui um controlador de aplicação e um *gateway*. O controlador da aplicação seria parte da aplicação que se encarrega de interagir com o cliente. Logo que esta parte da aplicação utiliza outras partes para realizar e atender os serviços solicitados. O *gateway*, caminho de passagem da informação, se encarrega de transformar os dados que são manipulados em XML, para que possa ser entendido pelo Web Browser do cliente.

Outro elemento da arquitetura RIA, é o servidor onde se hospeda a aplicação. Este servidor de aplicação compila a aplicação Web, caso necessário, e realiza a intermediação da comunicação da aplicação RIA executando no Web Browser do cliente com outras aplicações ou *servlets*, no mesmo servidor ou mesmo em outros servidores da Internet. Esta parte da arquitetura se encarrega também de obter os dados necessários para a aplicação, onde estes dados podem estar em uma base de dados qualquer ou em arquivos.

A arquitetura também engloba um gerenciador de banco de dados, que habitualmente é uma base de dados onde se manipulam os dados relacionados ao serviço oferecido pela aplicação. Esta parte da arquitetura nem sempre é necessária, já que existem casos de aplicações que não utilizam uma base de dados própria, mas utiliza serviços disponíveis na Internet.

É importante frisar que na arquitetura de aplicações RIA, a parte correspondente ao cliente contém toda a parte gráfica da aplicação, com isto torna-se possível a redução da comunicação entre cliente/servidor a poucos dados, somente o mínimo necessário (normalmente os dados são em formato XML). (CABRERA, 2006)

## **4.2 Utilidade**

Segundo (LÓPEZ, 2005), as utilidades e possibilidade de uso de uma aplicação Web tradicional, nas tecnologias RIA, este número de utilidades e possibilidades de uso aumentaram consideravelmente, sendo estas desde *Webmails*, fóruns, *address book*, agendas, apresentações, jogos, carrinhos de compras entre outras possibilidades de uso.

A tabela 3 mostra uma grande quantidade de aplicações e serviços que a tecnologia RIA pode oferecer, tanto para a Internet como para Extranets e Intranets.

Internet			Extranet		Intranet
Interactive Marketing	eBusiness Apps	Customer Apps	Partner Apps	Enterprise IT Apps	Department IT Apps
Presentations Media Streaming Online ads Games Offline Kiosks	Catalogs Product Tours <b>FootJoy</b> Simulations Configurators <b>MINI USA</b> Shopping Carts Wireless	Customer Portals Self-service <b>Broadmoor Hotel</b> Customer Service <b>E*TRADE Financial</b> Collaboration eLearning	Channel Portals <b>Yankee Candle</b> B2B Supply Chain Exchanges Customer Tracking	Employee Portals CRM Data Visualization <b>Charles Schwab</b> HR Systems Sales Reports	Business Visualization eLearning Data Reporting Productivity Applications <b>FleetBoston Financial</b>

Tabela 3 - Lista de utilidade de aplicações RIA. (DUHL, 2003)

### 4.3 Tecnologias

Existem diversas tecnologias que propõem o desenvolvimento de RIA's, algumas delas são:

AJAX – Sigla para *Asynchronous JavaScript and XML* (JavaScript Assíncrono e XML), AJAX não é um programa, mas sim uma técnica de desenvolvimento de sistemas web mais interativos com o usuário (GARRET, 2005).

Java – Pode-se dizer que Java é o pioneiro em ambiente de aplicações web, iniciada com os famosos *Java Applets*. Atualmente esta tecnologia apresenta variações para todos os tipos de ambientes de desenvolvimento, desde aplicações para celulares até sistemas de grande porte (mainframes). Todos eles dispoñdo de inúmeros recursos para se desenvolver interfaces agradáveis e amigáveis ao usuário (SUN, .2005).

Outra inovação de Java é o JWS – *Java Web Start*, visto com mais detalhes no capítulo 5 deste trabalho.

Flash – O termo “Flash” é associado a pequenas animações com a intenção de deixar um web site mais bonito. No entanto, o desenvolvimento de aplicações web completas utilizando-se desta ferramenta tem crescido gradativamente.

XUL – sigla para *XML User Interface Language*, que basicamente é uma linguagem multiplataforma para desenvolver interfaces gráficas de usuários (GUI – *Graphical User Interface*) em aplicações, inicialmente foi criado para tornar o desenvolvimento do navegador Mozilla mais fácil, rápido e portátil. Utiliza-se da tecnologia XML como sua base, e como o XML tem as vantagens de outras linguagens de marcação como XHTML, MathML, SVG, etc. (VIEIRA, 2005).

Oracle Forms – Produto comercial para construção de aplicações web centradas no banco de dados Oracle (O’ROURKE, 2004). Utiliza-se de todo o poder deste banco de dados e da linguagem PL/SQL.

Dentre as tecnologias listadas acima, AJAX e JWS serão o alvo principal da pesquisa. A primeira pelo fato de ser atualmente a mais usada e apresenta uma série de casos de uso de grande qualidade. A segunda por permitir realmente que uma aplicação desenvolvida para o ambiente desktop seja distribuída e utilizada via web.

## 5. Java Web Start - JWS

O usuário pode baixar e executar aplicações sem passar por procedimentos complicados de instalação. Caso a aplicação não esteja presente na máquina do cliente, o *Java Web Start* irá automaticamente baixar todos os arquivos necessários para execução da aplicação. Caso exista alguma versão da aplicação na máquina do cliente, esta aplicação estará sempre pronta para ser executada a qualquer hora que o usuário necessitar, através de um ícone em seu desktop ou através de um link em um *Web Browser*. Independente do modo que aplicação for invocada, uma versão mais recente da aplicação sempre estará presente para o usuário. (SUN, 2005)

Java Web Start suporta:

- Versionamento e atualização incremental;
- Operações Off-line;
- Integração com Desktop;
- Sandboxing (Ambiente protegido com restrições de acesso a disco ou recurso de rede);
- Code-signing;
- Instalação automática de JRE e pacotes adicionais da aplicação.

### 5.1 Como funciona o JWS – Perspectiva do Usuário

Java Web Start utiliza a tecnologia *Java Network Launching Protocol & API* (JNLP). A tecnologia JNLP define, além de outras coisas, um formato padrão de arquivo (arquivo JNLP) que descreve como a aplicação será executada.

JWS é projetado para que o usuário acesse de maneira fácil suas aplicações e trabalhe com aplicações Java robustas. É de fácil instalação e uso pois é gerenciado pela JRE (*Java Runtime Environment*) e integrado com qualquer sistema operacional (Windows, Solaris, Linux). (SUN, 2005)

### 5.1.2 Instalação e uso de uma aplicação JWS

Fazer *download* de uma aplicação JWS é bastante simples, basta clicar no link disponibilizado em um *Website*. A figura 3 ilustra o *website* com acesso a uma aplicação JWS.

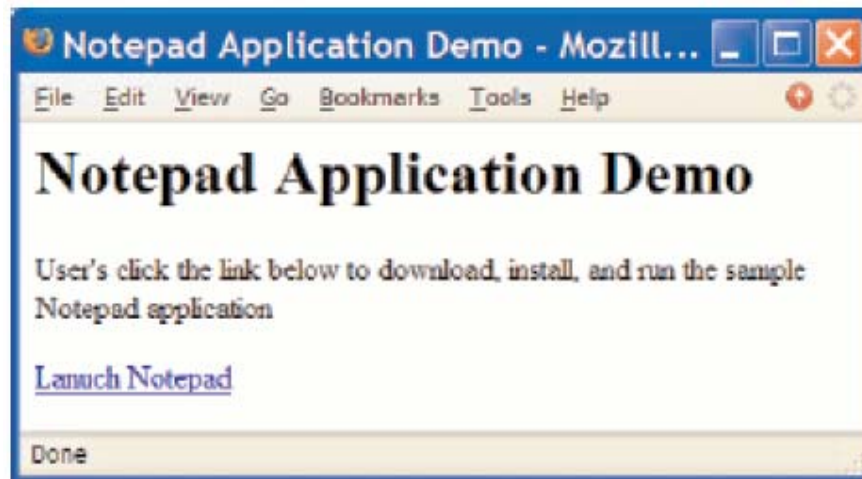


Figura 3- Página para aplicação exemplo Notepad

A figura 4 ilustra a aplicação Notepad.

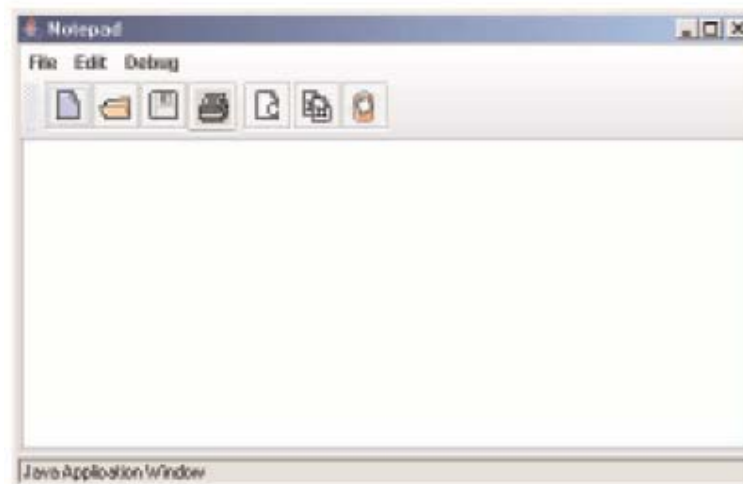


Figura 4 - Aplicação Notepad

Duas ações importantes acontecem quando se tenta executar uma aplicação deste tipo:

- Se o usuário não possui a versão correta da JRE, o JWS baixa da Internet e instala.
- O JWS adiciona a aplicação no computador que acessou o link, assim o usuário não precisa retornar ao *Website* para utilizar a aplicação novamente.

## **5.2 JWS – Perspectiva do Desenvolvedor**

JWS é projetado para fazer a distribuição de maneira fácil tanto para desenvolvedores quanto para usuário. O layout e os pacotes da aplicação para a distribuição com JWS é feito da mesma forma que em qualquer programa em JAVA. (SUN, 2005).

### **5.2.1 Distribuindo uma Aplicação JWS**

Para distribuir uma aplicação JWS alguns passos devem ser seguidos:

1. Configuração do *WebServer*
2. Criação do arquivo JNLP
3. Colocar a aplicação no servidor
4. Criar o *Website*

Algumas vantagens podem ser observadas nas aplicação JWS:

- Fácil instalação
- Independência de plataforma;
- Integração com o *desktop*
- Fácil atualização
- Segurança
- Desempenho

## 6 AJAX

Desenvolvida desde os inícios dos anos 90 e com apoio sustentado de empresas de renome internacional como a Microsoft ©, este tipo de arquitetura (Web como plataforma) foi ganhando forma e robustez, ganhando a confiança do mercado quando surgiram os primeiros produtos de utilização massiva nos últimos 2 anos (Google Maps, Gmail, entre outros). Esta filosofia em nível de arquitetura constitui sem dúvida uma revolução nas aplicações que hoje em dia conhecemos. À medida que se vão explorando as maravilhas do AJAX, vão sendo disponibilizados novos serviços via web que de outro modo seriam impraticáveis. (ROMÃO, 2005)

### 6.1 O que é AJAX?

“*Asynchronous JavaScript and XML*” (JavaScript Assíncrono e XML), mais conhecido por AJAX, em termos tecnológicos não é algo revolucionário, mas trata-se do conjunto de várias tecnologias já existentes.

GARRETT (2005) define AJAX da seguinte forma:

“O AJAX não é uma tecnologia. São na realidade várias tecnologias, cada uma progredindo de forma independente, e que se juntaram de forma a possibilitar uma melhora nas formas de interação com os utilizadores em aplicações Web”.

É apenas um estilo de desenho, que junta todas as características dos Web Browsers atuais de forma a produzir aplicações que se assemelham menos à Web e mais ao Desktop.

O AJAX utiliza as seguintes tecnologias:

- Apresentação baseada em padrões, utilizando XHTML e CSS;
- Interação e apresentação dinâmica utilizando o *Document Object Model* (DOM);
- Formato padrão para troca e manipulação de dados – XML;

- Comunicação assíncrona com o servidor utilizando *XMLHttpRequest*
- Utiliza Javascript na interligação de todas estas tecnologias.

## 6.2 Comparação entre o modelo clássico de aplicações web e o modelo AJAX

A figura abaixo (GARRET, 2005) ilustra as diferenças entre as aplicações clássicas Web e as que usam o AJAX. No modelo clássico das aplicações Web, a maior parte das interações na interface (UI - *User Interface*) passa por um ciclo no qual são efetuados pedidos HTTP ao servidor Web, esperar pela resposta, e depois de recebida o cliente redesenha por completo a página HTML. Este modelo baseado no modelo original da Web está adaptado para “hipertexto” (páginas com ligações para outras páginas), mas não para a criação de aplicações.

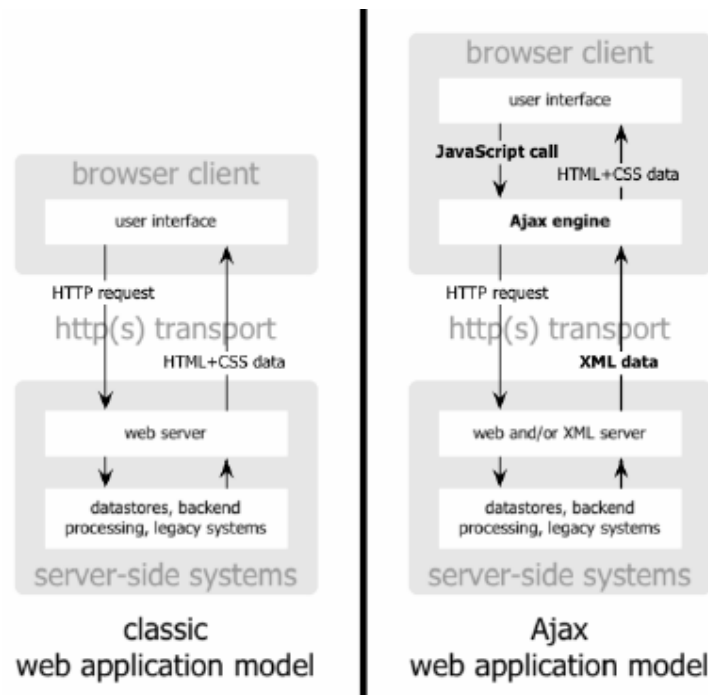


Figura 5 - Modelo tradicional de aplicações web comparado com o modelo AJAX

Segundo (TRINDADE, REIS, 2005) as aplicações que sigam o modelo AJAX dão um grande passo quando se tenta comparar com a complexidade e riqueza das aplicações desktop tradicionais. Tal tecnologia apresenta algumas vantagens, que são:

**Maior Interatividade nas aplicações:** É uma das principais vantagens e uma das principais razões para o crescente uso da tecnologia AJAX. Como a maior parte da aplicação corre no lado do cliente, isto permite que uma aplicação AJAX se comporte como uma aplicação Desktop e não seja tão limitada pela rede em termos de respostas do servidor, dando a sensação de continuidade nas ações efetuadas.

Além disso, as aplicações AJAX são mais reativa e permitem uma maior riqueza em termos gráficos (efeitos visuais que nos indicam o estado de um envio ou poder arrastar e colar elementos tal como se estamos habituados numa aplicação desktop).

**Redução das comunicações:** Na maioria das aplicações Web que utilizam o modelo AJAX devido a sua natureza de atualizações granulares, consegue-se reduzir a quantidade de informação trocada com o servidor (não é necessário reenviar vezes sem conta toda a estrutura HTML ou partes de uma página que não foram alteradas).

**Redução da carga de processamento do servidor:** Em casos de aplicações que envolvam efetuar muitos cálculos no lado do servidor, através de aplicações Ajax é possível em alguns casos trazer parte desses cálculos para o cliente (por exemplo, a ordenação de tabelas de produtos), permitindo desta forma reduzir o processamento necessário por parte do servidor.

**Não é proprietário:** AJAX não é um nome de nenhuma marca ou produto, apenas uma designação de um método de desenho para aplicações Web, utilizando um conjunto de tecnologias padronizadas já existentes.

**Portabilidade:** Como o AJAX é constituído por tecnologias que são suportadas pela maioria dos browsers existentes no mercado, ela não se restringe a um browser, nem a uma plataforma. Além disso, não requer a instalação de qualquer plug-in no browser ou software do cliente.

Segundo (TRINDADE, REIS, 2005) o Ajax traz algumas preocupações. Boa parte dessas preocupações tem a ver com o comportamento dos browsers. Essas limitações são:

**Capacidades limitadas:** Algumas aplicações Ajax conseguem fazer coisas inimagináveis á algum tempo na Web, mas existem muitas restrições nesta plataforma. Por conseqüente o Ajax ao se basear nas tecnologias existentes, herda as suas limitações. Exemplos de limitações são: pouca capacidade multimídia (*streaming de video*), incapacidade de usar armazenamento local no cliente, gráficos em tempo real, interação com hardware (impressoras, *webcams*). Algumas destas limitações têm vindo a ser solucionadas nos browsers mais recentes através da instalação de plug-ins específicos (como por exemplo o Flash), mas existem algumas que serão muito difíceis de ultrapassar devido á natureza não-aplicacional da Web.

**Performance do cliente:** Numa aplicação Ajax transfere-se muito do processamento do servidor para o cliente. Essa mudança tem custos porque estaremos delegando para o cliente a responsabilidade por realizar determinadas operações para as quais não estaria inicialmente destinado. Poderemos sobrecarregar o cliente caso não se tomem as devidas precauções durante a fase de desenvolvimento deste tipo de aplicação *Web*.

**Comportamento dos botões *Back* e *Forward*:** Como as atualizações das páginas passam a ser dinâmica, por exemplo, os botões “*back*” e “*forward*” dos browsers podem ter comportamentos que não são os pretendidos. Para resolver este problema começam a aparecer algumas soluções, onde a maioria passa pela criação ou utilização de *Frames* invisíveis que guardam os diversos estados da aplicação e preenchem o histórico utilizado pelos botões “*back*” e “*forward*”.

***Bookmarking (Unique URL)*:** Outro problema associado ao anterior é o fato de se tornar difícil a tarefa do usuário de guardar um link para um determinado estado da aplicação. Para este problema também existem algumas soluções, entre elas a utilização do identificador de fragmento do URL (“*anchor*”), que permite aos usuários voltarem a um determinado estado da aplicação.

**Latência da rede:** Outra preocupação das aplicações AJAX é a latência de rede. Deverá existir a preocupação de quando for efetuado um pedido ao servidor, ser dado algum “*feedback*” para que o utilizador não pense que a aplicação falhou, caso o pedido

demore entre a indicação para enviar e ser mostrada a resposta. Este problema tem tendência a diminuir com o aumento do uso de banda larga, mas nunca deixará de existir devido á natureza da comunicação instável na Internet.

**Requer conectividade permanente:** O navegador busca informações constantemente no servidor, para efetuar as atualizações nas informações apresentadas pelo site.

### **6.3 Casos de sucesso com AJAX**

Existe atualmente inúmeros casos de sucesso que utilizam a tecnologia AJAX no desenvolvimento de sistemas, são alguns deles:

Google Mail <http://mail.google.com>

Serviço de E-Mail com capacidade superior a 2,5 GB, que tira proveito de um poderoso motor de busca do Google para encontrar mensagens. Mostra todas as respostas a um e-mail quando este é aberto, permitindo desta forma acompanhá-lo como uma conversação.

A9.com <http://www.a9.com>

Motor de busca inovador desenvolvido pela Amazon.

Writely <http://www.writely.com>

Editor de texto online.

Flickr <http://www.flickr.com>

Serviço do Yahoo, para armazenar, procurar, organizar e partilhar fotos na web.

Meebo <http://www.meebo.com>

Aplicação de mensagens instantâneas que se liga às redes ICQ/AIM, Yahoo, MSN e Jabber/Google Talk.

## 7. MVC

*Model View Controller* ou Modelo-Visão-Controlador é um padrão de arquitetura de aplicações que visa separar a lógica da aplicação (*Model*), da interface do usuário (*View*) e do fluxo da aplicação (*Controller*). Permite que a mesma lógica de negócios possa ser acessada e visualizada por várias interfaces.

MVC também é utilizado em padrões de projetos de software, entretanto, MVC abrange mais da arquitetura de uma aplicação do que é típico para um padrão de projeto.

### 7.1 Funcionamento MVC

Em um projeto de software baseado no padrão MVC, define-se em uma arquitetura básica com 3 camadas possivelmente abstratas:

- **Model:** Implementa o modelo representando a estrutura de baixo nível do projeto, podendo ser o modelo objeto-relacional que implementa a camada de dados, e ou num caso de MVC de Interface poderia guardar informações de estado dos controles.
- **Controller:** Implementa a camada responsável pelo gerenciamento de eventos no projeto, tais como clique do usuário, chamando a camada *Model* para processar os eventos, também pode manter informações de estado do usuário na aplicação.
- **View:** Gera a interface com usuário de modo que esta somente requisiite o processamento de eventos pelo *Controller*.

Para uma implementação correta, as camadas *Model*, *Controller* e *View* devem ser implementadas de forma que a inversão da ordem não acarrete problemas por dependência, ou seja a camada de interface (*View*) depende de controle (*Controller*) que implementa um Modelo (*Model*), mas nunca o inverso.

Os padrões de projeto de software ou padrões de desenho de software, também muito conhecido pelo termo original em inglês: *Design Patterns*, descrevem soluções para

problemas recorrentes no desenvolvimento de sistemas de software orientados a objetos. Um padrão de projeto estabelece um nome e define o problema, a solução, quando aplicar esta solução e suas conseqüências.

Os padrões de projeto visam facilitar a reutilização de soluções de desenho - isto é, soluções na fase de projeto do software, sem considerar reutilização de código. Também acarretam um vocabulário comum de desenho, facilitando comunicação, documentação e aprendizado dos sistemas de software.

A interface tem a função de melhorar e facilitar o contato entre o homem e a máquina e para entendermos esse estudo, devemos ficar atentos a vários aspectos psicológicos, informativos, estéticos e ergonômicos.

## **8. Estudo de Caso**

### ***8.1 Estrutura e Descrição da aplicação***

O estudo de caso em questão trata-se do desenvolvimento de uma aplicação Web com duas implementações distintas para a IU (Interface com o usuário)

Para o desenvolvimento da aplicação em Java, é utilizado o padrão MVC (Model-View-Controller).

A interface da aplicação JWS é feita utilizando-se da API Swing da linguagem Java, onde é possível encontrar todos os componentes encontrados em qualquer IDE de desenvolvimento de software para desktop.

A interface da aplicação feita utilizando a tecnologia AJAX é feita em HTML puro.

O protótipo trata-se de um sistema de controle de preços de produtos (parte de um sistema de controle de estoque) para servir de apoio ao administrador de empresa para saber exatamente o custo dos produtos e os lucros que podem ser obtidos com o mesmo, ou seja, um cadastro de produtos onde os campos dos valores possuem alguns tratamentos.

### ***8.2 Requisitos do sistema:***

O sistema deverá prover o cadastro de produtos no estoque e o controle de preços dos mesmos.

O sistema deverá, mediante o preço de custo do produto, realizar todos os cálculos necessários para a obtenção da sugestão de venda, podendo o usuário alterar o valor de acordo com o seu entendimento.

### ***8.3 Atores***

- Administrador – tem a função de prover a manutenção do cadastro de produtos, entradas, saídas e alterações.
- Usuário – tem a habilidade de fazer consultas no sistema.

## 8.4 Casos de Uso

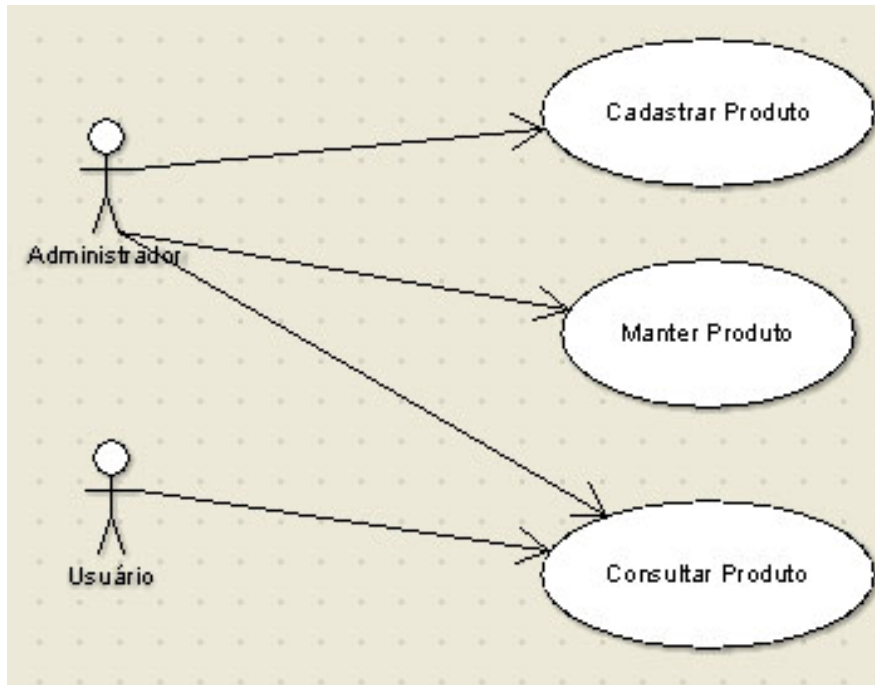


Figura 6 - Diagrama de Casos de Uso

### 8.4.1 Cadastrar Produto

Caso de uso onde o ator (Administrador) efetua o cadastro de um novo produto. Caso o produto já exista na base de dados do sistema, o sistema efetua a busca do produto e apresenta os dados atuais ao usuário, evitando assim a duplicação de informações.

### 8.4.2 Manter Produto

Caso de uso onde o ator (Administrador), tem acesso a lista de produtos cadastrados no sistema, podendo verificar na lista mostrada, ou realizar uma busca filtrando os itens da lista apresentada pela digitação das letras iniciais dos produtos, uma vez localizado o produto o usuário pode editar as informações necessárias, sem comprometer o cadastro desse produto ou apagar o produto.

### 8.4.3 Consultar Produto

Caso de uso onde os atores podem fazer consulta na base de dados de produtos.

Os produtos podem ser consultados por código, descrição, data e valor.

## 8.5 Diagrama de Classe da Implementação em Java

O Diagrama de Classe da aplicação exemplo é bastante simples. A aplicação é constituída de 3 pacotes principais. O pacote de classe Visão, que possui as classes de interface com o usuário, o pacote modelo que apresenta as classes de acesso ao banco de dados e o pacote controle onde estão as classes que fazem a ligação entre as classes que manipulam a base de dados e a interface do sistema. Existem outras classes utilizadas no desenvolvimento da aplicação, que são omitidas no diagrama de classes, são classes de controle da interface com o usuário.

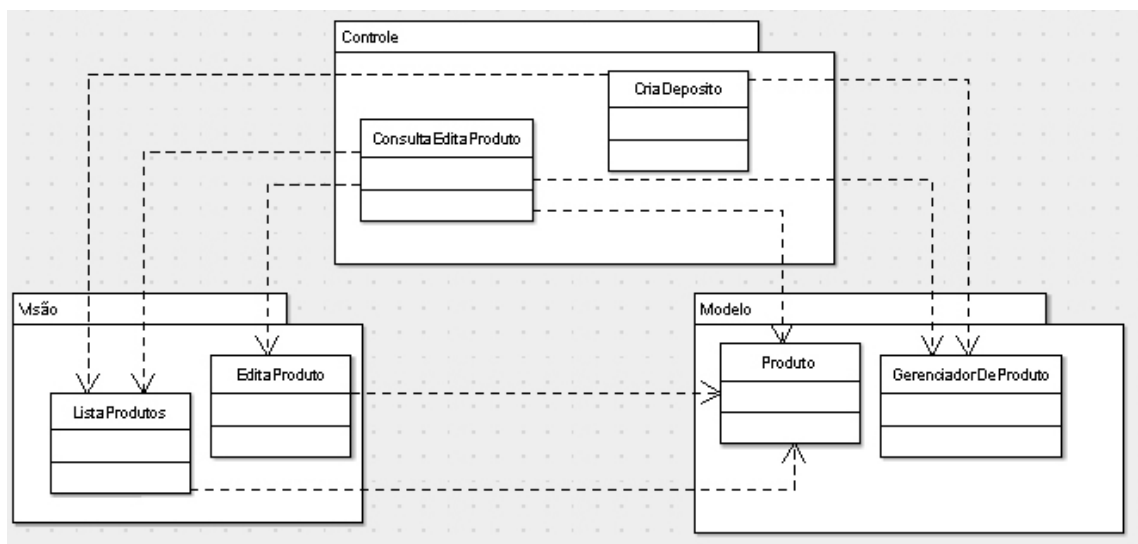


Figura 7 - Diagrama de Classes

### 8.5.1 Pacote de Classes Controle

Neste pacote estão presentes as classes responsáveis por fazer a ligação das classes do pacote modelo e das classes do pacote visão.

- ConsultaEditaProdutos – classe responsável por obter as informações junto a classe GerenciadorDeProduto e repassar para as classes do pacote Visão. É o principal controlador da aplicação.
- ProdutoDAO – classe que acessa base de dados, especificamente a tabela produto.
- CriaDeposito – gerencia qual depósito esta sendo utilizado. Cria um novo arquivo de produtos, para um novo depósito.
- AlterarProdutoDAO – classe que altera o produto.
- ListaProdutoDAO – classe que lista os produtos armazenados na base de dados.
- InserirProdutoDAO – classe que insere um novo produto.
- RemoverProdutoDAO – classe que remove o produto.
- ControladorDeConexões – classe responsável por controlar a conexão com a base de dados, assim fica mais simples caso seja necessário a mudança do banco.

### 8.5.2 Pacote de Classes Modelo

As classes deste pacote são responsáveis por fazer a ligação do sistema com a base de dados, abstraindo o significado das características dos produtos.

- Produto – classe que representa o produto, apresenta os getters e setters.
- GerenciadorDeProdutos – classe que gerencia o objeto produto, sua inclusão, exclusão, consulta.
- ValidacaoException – indica um erro de validação na digitação dos dados de um produto.
- ModeloException – superclasse para todas as classes de exceção do pacote modelo.

### 8.5.3 Pacote de Classes Visão

O pacote de classes visão possui as classes responsáveis por fazer a captura dos dados necessários para o usuário, tanto no ato de efetuar um cadastro quanto no momento de efetuar uma busca.

- ListaProdutos – classe responsável por listar os produtos cadastrados no sistema;
- EditarProdutos – classe que gera a interface para manter o produto (cadastrar, alterar e remover).
- ActionSupport – superclasse abstrata das classes do pacote visão, fornece métodos utilitários para a geração dos eventos *actionPerformed* para as classes de controle, seguindo o modelo de eventos do Swing/AWT
- ProdutosColumnModel – customiza um *JTable* para exibir cinco colunas (Código, Produto, Quantidade, Valor, Imprimir).
- ProdutosTableModel – customiza um *JTable* para exibir as informações a partir de uma lista de objetos Produto (`List<Produto>`).

## 8.6 Implementação em AJAX

### 8.6.1 Estrutura da implementação

A implementação em AJAX possui uma estrutura simples, com poucos arquivos é possível fazer o cadastro e a consulta dos dados armazenados no banco.

Diferente da implementação em Java, que utiliza o padrão MVC, esta não adota nenhum padrão de projeto específico.



Figura 8 - Arquitetura da Implementação em AJAX

### **8.6.2 Descrição dos arquivos**

- index.jsp – arquivo que contém a chamada para as telas de cadastro de pesquisa da aplicação em AJAX
- produtos.jsp – arquivo responsável pela conexão com o banco e a geração do arquivo xml que será utilizado como referencia para as pesquisas. Este arquivo contém os comandos SQL de consulta, inserção, remoção e atualização do banco de dados.
- pesquisar\_produto.html – tela de busca e visualização dos produtos armazenados em estoque.
- cadastrar\_produto.html – tela de cadastro de produtos.

### **8.7 Infra-Estrutura utilizada**

Foi utilizado como infra-estrutura para o desenvolvimento e testes a seguinte configuração:

- Notebook HP/Compaq nx9010
- Windows XP Professional em inglês
- Pentium 4, 2.80 GHz, 1Gb Memória Ram, HD 60 Gb
- Servidor local.

## 9 – Resultados e Observações

### 9.1 Curva de aprendizado

Para um desenvolvedor que já possua conhecimento na linguagem Java, a familiarização da API Swing da tecnologia Java é bastante rápida tornando a curva de aprendizado em JWS curta. Para aqueles que estão o inícios dos estudos sobre a tecnologia Java e para aqueles que nunca tiveram contato com a linguagem, entender os conceitos pertinentes ao funcionamento e a como tornar uma aplicação Java funcional através do JWS não apresenta dificuldades.

A curva de aprendizado em AJAX é mais longa e trabalhosa, mesmo que a aplicação desenvolvida seja simples, pelo fato de AJAX ser uma união de diversas tecnologias diferentes.

Para se desenvolver uma interface com o usuário rica e atrativa em AJAX é necessário que o desenvolvedor tenha amplos conhecimentos em CSS, Javascript e DHTML, já para se fazer uma interface amigável com Swing o conhecimento da em orientação a objetos e na linguagem Java já se faz suficiente.

### 9.2 Limitações

As interfaces desenvolvidas utilizando-se da tecnologia Java (Swing) seguem um padrão onde podem ser utilizados tabelas, *grids*, imagens entre outros componentes encontrados em qualquer tecnologia de desenvolvimento de aplicações *desktop*. Não é possível inserir vídeos, sons, animações, ou seja, recursos mais interativos.

Já as implementações utilizando-se da tecnologia AJAX, possui a possibilidade de apresentarem maior interação com o usuário, através de sons, imagens animadas, ou seja, todo o poder que pode ser encontrado em sites web.

### **9.3 Carregamento**

A comparação do carregamento e performance da aplicação foi realizado na mesma máquina de desenvolvimento.

A aplicação desenvolvida em Swing inicialmente apresenta um tempo de carregamento maior, pelo fato de precisar ser feito o *download* da aplicação completa no computador do usuário, após isso ser realizado a aplicação torna-se mais rápida fazendo apenas pequenas cargas de classe caso seja necessário.

Já a aplicação em AJAX apresenta um tempo de resposta bastante eficiente, pois trata-se de uma aplicação simples com recursos visuais leves.

### **9.4 Comparação Geral**

Como pode ser observado, no desenvolvimento das aplicações, a diferença no número de arquivos é bastante significativa, para se fazer o tratamento de uma tabela em Swing é necessário a criação de várias outras classes de apoio, enquanto q em ajax basta criar uma função javascript que capture os dados e os formate em uma tabela html, isso torna o tempo de desenvolvimento e por conseqüência o tempo de manutenção significativamente menor.

Para se ter uma noção da comparação do tempo gasto em ambas as implementações, em toda a implementação feita utilizando a tecnologia AJAX é encontrado cerca de 600 (seiscentas) linhas de código, divididos em todos os arquivos. Já na implementação feita de Java, apenas no pacote visão, é encontrada cerca de 1.000 (mil) linhas de código.

Ambas as aplicações são independentes de plataforma e utilizam a mesma base de dados, tornando assim a distribuição mais fácil e sem complicações.

## **10. Considerações Finais**

### **10.1 Conclusão**

O desenvolvimento de aplicações que tenham o ambiente Web como foco principal é tarefa delicada e dispendiosa, visto que o desenvolvedor não pode se restringir a um único tipo de plataforma, como é o caso de uma aplicação desenvolvida de maneira tradicional (*desktop*), onde é possível determinar a configuração mínima necessária para “rodar” a aplicação.

Após estudar diversas tecnologias diferentes para se desenvolver aplicações, tanto Web quanto para *Desktop*, uma coisa fica bem definida. Que a forma que o programa interage com o usuário é o mais importante, não importa se o sistema é de pequeno ou grande porte. A palavra final sempre será daquele que tem de usar o sistema diariamente.

Esperamos que com este estudo leve a novos estudos comparativos mais aprofundados entre as tecnologias de desenvolvimento web existentes no “mercado”. Este tipo de estudo se faz necessário ao ponto de mostrar a comunidade os pontos positivos e negativos de cada tecnologia para que seja um referencial aos desenvolvedores na hora da sua escolha.

### **10.2 Trabalhos Futuros**

Este trabalho teve por objetivo fazer um levantamento das tecnologias existentes atualmente no mercado no que diz respeito ao desenvolvimento de aplicações voltadas ao ambiente web. Faz-se necessário realizar pesquisas mais aprofundadas quanto ao desempenho de tais tecnologias e onde elas se encaixam melhor.

## 11. Referências bibliográficas

CABRERA, **OpenLaszlo – Uma nova proposta para o desenvolvimento da camada de apresentação de sistemas Web**. Centro de Ensino Superior: Foz do Iguaçu, 2006.

GARRETT, Jesse J. **A New Approach to Web Applications**. Adaptive Path: 2005.  
Acessado em: Abril de 2005. Disponível em:

<http://adaptivepath.com/publications/essays/archives/000385.php>

LÓPEZ, Xavier Farré. Trabalho de Conclusão de curso: **Rich Internet Application**. Universidade Politécnica da Catalunia, Espanha, 2005. Disponível em <http://bibliotecna.upc.es/PFC/arxiu/migrats/40624-4.pdf> Acessado em: setembro de 2006.

MACROMEDIA, **Desenvolvimento de Aplicações “rich” para a Internet com o Macromedia MX**. Abril de 2002.

Disponível em <http://www.macromedia.com>

MACROMEDIA, **Flash Professional and Flex**. Novembro de 2005.

Disponível em [http://www.macromedia.com/devnet/flash/articles/flex2\\_flash.html](http://www.macromedia.com/devnet/flash/articles/flex2_flash.html)

MACROMEDIA, **Providing a Flex Front End to Your Struts Applications**. Novembro de 2005. Disponível em <http://www.macromedia.com/devnet/flex/articles/struts.html>

Mozilla Developer Center. **JavaScript** Disponível em:

<http://developer.mozilla.org/pt/docs/JavaScript>

MURRAY, Greg. **Asynchronous JavaScript Technology and XML (AJAX) With Java 2 Platform, Enterprise Edition**. Sun Developer Network, 2005.

Disponível em: <http://java.sun.com/developer/technicalArticles/J2EE/AJAX/>

NODA, Tom, HELWIG, Shawn. **Rich Internet Application**. Madison: University of Wisconsin-Madison, 2005. Acessado em: Abril de 2005. Disponível em:

<http://www.uwebc.org/opinionpapers/archives/docs/RIA.pdf>

O'REILLY, Tim **What is the web 2.0? Design Patterns and Business Models for the Next Generation of Software**. Publicado em: 30/09/2005. Disponível em:

<http://www.oreillynet.com/pub/a/oreilly/tim/news/2005/09/30/what-is-web-20.html>.

Acessado em: junho de 2006.

O'ROUKE, Cameron. **A Look at Rich Internet Application**. Oracle, 2004. Acessado em:

Abril de 2005. Disponível em: [http://www.oracle.com/technology/oramag/oracle/04-jul/o44dev\\_trends.html](http://www.oracle.com/technology/oramag/oracle/04-jul/o44dev_trends.html)

ROMÃO, João, et al. Tecnologia Ajax. Universidade do Porto, 2005. Disponível em:

<http://paginas.fe.up.pt/~ei01061/trabalhos.htm>

SILVA, Edna L., MANEZES, Estera M. **Metodologia de Pesquisa e Elaboração de Dissertação**. Florianópolis: Universidade Federal de Santa Catarina, 3ª edição, 2001.

SUN MICROSYSTEMS. **Java Web Start Overview**. Maio de 2005. Disponível em

[http://java.sun.com/developer/technicalArticles/WebServices/JWS\\_2/JWS\\_White\\_Paper.pdf](http://java.sun.com/developer/technicalArticles/WebServices/JWS_2/JWS_White_Paper.pdf)

SUN MICROSYSTEMS. **The Java Tutorial**. Novembro de 2005. Disponível em

<http://java.sun.com/docs/books/tutorial/>

TRINDADE, Luis, REIS, Ricardo. **AJAX: Introdução**. Portugal: IPLNet, 2005. Disponível

em: <http://pwp.net.ipl.pt/alunos.isel/24138/AJAX/IntroducaoAJAX.pdf>

VIEIRA, Marcio J. XUL – **Interfaces de Usuários Portáteis com Tecnologia XML**.

Curitiba: Universidade Federal do Paraná, 2005.

Acessado em: Abril de 2005. Disponível em:

[http://www.ambientelivre.com.br/xulwarehouse/xul\\_interfaces\\_de\\_usuarios\\_portateis.pdf](http://www.ambientelivre.com.br/xulwarehouse/xul_interfaces_de_usuarios_portateis.pdf)

Web 2.0 Conference. Conferência Mundial sobre web 2.0.

WIKIPEDIA. **User interface**. 2006. Acessado em: Julho de 2005. Disponível em:

[http://en.wikipedia.org/wiki/User\\_interface](http://en.wikipedia.org/wiki/User_interface)

**XML User Interface Language (XUL)**. Mozilla. Disponível em:

<http://www.mozilla.org/projects/xul/>