MINISTÉRIO DA EDUCAÇÃO UNIVERSIDADE
TECNOLÓGICA FEDERAL DO PARANÁ ESPECIALIZAÇÃO
EM TECNOLOGIA JAVA

LUCIANA MICHELON

JAVA E FLEX – APLICAÇÃO RÁPIDA, CONFIÁVEL E FLEXÍVEL

CURITIBA – PR
2011

MINISTÉRIO DA EDUCAÇÃO UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ ESPECIALIZAÇÃO EM TECNOLOGIA JAVA

LUCIANA MICHELON

JAVA E FLEX – APLICAÇÃO RÁPIDA, CONFIÁVEL E FLEXÍVEL

Monografia apresentada como requisito parcial para conclusão do Curso de Especialização em Tecnologia Java - UTFPR.

Orientador: Prof. Nelson Kashima
Co-Orientador: Marisângela Pacheco Brittes

CURITIBA – PR
2011

RESUMO

A expansão da *web* e seu fortalecimento como uma plataforma de desenvolvimento de aplicações está atraindo cada vez mais usuários e tem como principal diferencial a facilidade de uso que o usuário tem de obter informações em qualquer lugar. No entanto, à medida que cresce a extensão do uso desses aplicativos, aumenta a preocupação com funcionalidades básicas da interação com o usuário, porque a metodologia de desenvolvimento está provando ser muito limitado, tendo uma interface de usuário difícil, baixa velocidade de processamento de informação e difícil manejo. Dentro desse contexto, o objetivo deste trabalho é estudar dois quadros para o desenvolvimento de aplicações *web* (Flex e JSP), e desenvolver uma *Rich Internet Application* (RIA), a fim de avaliar os benefícios que podem ser alcançados utilizando esse recurso tecnológico *web*. Para melhor avaliar os quadros, um exemplo de uma aplicação CRUD (*Create Read Update Delete*) foi desenvolvido, de forma simples e rápida, usando cada uma das duas tecnologias, tentando demonstrar, na prática, as vantagens e desvantagens de cada estrutura de desenvolvimento.

Palavras-chave: Web, Java, Flex, JSP, RIA.

ABSTRACT

The expansion of the web and its strengthening as an application development platform is attracting increasingly users and has as a main differential the ease of use that the user has to obtain information anywhere. However, as it grows the extension of use of such applications, increases the concern about basic functionalities of the user interaction, because the development methodology is proving to be very limited, having an difficult user interface, hard handling and low speed of information processing. Within that context, the goal of this work is to study two frameworks for development of web applications(Flex and JSP), and to develop a Rich Internet Application (RIA) in order to evaluate the benefits that can be achieved using this technological web feature. To better evaluate the frameworks, an example of a CRUD (Create Read Update Delete) application was developed, in a simple and fast manner, using each of the two technologies, trying to demonstrate in practice, the advantages and disadvantages of each development framework.

Keywords: Web, Java, Flex, JSP, RIA.

LISTA DE FIGURAS

LISTA DE SIGLAS

RIA - *Rich Internet Application*

CRUD - *Create Read Update Delete*

MVC - *Model View Control*

HTML - *Hypertext Markup Language*

JVM - Java Virtual Machine JEE

Java EJB - Enterprise JavaBeans

JSP - Java Server Pages

JME - Java Micro Edition

PDA - Personal Digital Assistant

EJB - Enterprise JavaBeans

JMS - Java Message Service

JSF - *Java Server Faces*

XML - *Extensible Markup Language*

MXML - *Macromedia Flex Markup Language*

SWF - *Small Web File*

JVM - *Java Virtual Machine*

HTTP - *Hyper Text Transfer Protocol*

IDE - *Integrated Development Environment*

API - *Application Programming Interface*

CSS - *Cascading Style Sheets*

SUMÁRIO

1 INTRODUÇÃO

A internet possibilita a troca de informações entre as pessoas, independente da distância geográfica. Desde o surgimento de sistemas computacionais, houve uma considerável evolução em sua variedade e escopo de aplicação, revolucionando os mais diversos processos organizacionais, e tornando-se aspecto central de muitas aplicações em diferentes setores da sociedade. Segundo SOUZA (2005), o Desenvolvimento de Sistemas *Web* é caracterizado pela necessidade da otimização do tempo em um ambiente onde a competição depende de quão rápido se atende às necessidades do cliente e da integração de todos os aspectos de negócio.

Com a necessidade de uma internet mais eficaz, surge uma nova geração de ferramentas, dando origem ao que é hoje conhecida como web 2.0, para o desenvolvimento *web* de "Aplicações ricas para Internet", mais conhecidas como RIA (*Rich Internet Application*). O RIA propõe que, através de ferramentas e tecnologias desenvolvidas para esse fim, seja possível a criação de aplicações similares às do ambiente *desktop*, com interfaces de fácil usabilidade e ágeis.

1.1 Objetivo Geral

O objetivo deste trabalho está direcionado a um estudo sobre tecnologia Java utilizando o *Flex* para desenvolver aplicações ricas para a internet, tendo em vista aplicar a usabilidade da ferramenta *Adobe Flex* e apresentar as melhorias e vantagens que se pode ter, utilizando este recurso tecnológico da *web* 2.0.

Serão aplicadas as técnicas de Java e um protótipo de exemplo, para comparar com as limitações encontradas em um aplicativo desenvolvido através da JSP e o outro aplicativo com Java e *Flex*.

Este trabalho demonstrará quais benefícios podem ser obtidos utilizando a metodologia Java para aplicativos *web* através da ferramenta *Adobe Flex*.

1.2 Objetivos Específicos

No desenvolvimento do projeto em RIA, combina a interatividade e a funcionalidade com a abrangência e a interatividade da web formando desta forma, uma única e integrada experiência, rica em conteúdo.

Para demonstrar esta representação, serão aplicadas neste trabalho, as metodologias da web 2.0, onde será realizado um estudo de caso comparando interfaces desenvolvidas em protótipos através da ferramenta *Adobe Flex* e outro protótipo utilizando a tecnologia JSP(*Java Server Page*), e também será realizado um estudo sobre a tecnologia Java com utilização do *Flex* com melhorias e desvantagens em utilizar tais tecnologias.

1.3 Justificativa

Durante muito tempo o uso da *Web* foi limitado à publicação de conteúdo em documentos em HTML(*HyperText Markup Language*). Conforme foi evoluindo a internet, aplicações mais complexas sob plataforma *Web* começaram a despertar o interesse de empresas e usuários comuns. À medida que cresceu a extensão de uso deste tipo de aplicação, aumentou a preocupação com relação às funcionalidades básicas para uma interação entre usuário e aplicação, não satisfatórios, pois apresentavam tecnologias muito limitadas com interfaces pouco amigáveis e de difícil manuseio e agilidade de processamento das informações.

O *Flex* surgiu para desenvolver aplicações ricas para *internet*, sendo uma tecnologia de desenvolvimento que propõe através de ferramentas para *web*, o desenvolvimento de aplicativos similares aos existentes, mas com interfaces ágeis e de fácil usabilidade.

1.4 Estrutura do Trabalho

O presente trabalho foi divido em cinco capítulos, sendo que o primeiro capítulo apresenta a introdução, os objetivos do trabalho e a justificativa.

O segundo capítulo refere-se à revisão bibliográfica das tecnologias estudadas e utilizadas, como JSP, *Flex* e RIA, além das vantagens e desvantagens de cada aplicação.

O terceiro capítulo apresenta as tecnologias para o desenvolvimento dos protótipos e as ferramentas RIA.

O quarto capítulo refere-se ao estudo de caso sobre o desenvolvimento dos protótipos desenvolvidos e a análise comparativa das metodologias e dificuldades encontradas.

E por último, no quinto capítulo são descritas as considerações finais do projeto.

2 FUNDAMENTAÇÃO TEÓRICA

Atualmente existem diversas bibliotecas que ajudam os desenvolvedores a construir interfaces gráficas para aplicações Web, onde que a maioria das bibliotecas são baseadas em *Flash* ou *Javascript*.

A junção de Java e *Flex* traz o melhor de cada uma das tecnologias. Somente o Java, até sobrevive, mas é preciso criar páginas *web* com saída HTML. O *Flex* sozinho não faz sentido, porque precisa persistir dados e realizar operações no servidor. O HTML sozinho não basta, pois necessita de o *Javascript* para mais interatividade ao sistema, ocorrendo assim incompatibilidades de *browsers* e dificuldades na implementação. O *Flex* surge para substituir toda a parte de HTML e *Javascript* do sistema que o JSP utiliza, acabando assim com tais problemas.

O desenvolvimento de aplicativos de *Internet* com servidores JEE, geralmente consiste de uma camada de apresentação, como *Struts* ou *Spring*. Essas ferramentas normalmente seguem o modelo MVC(*Model View Controller*). O modelo comum de programação para desenvolvimento *web* é permitir aos usuários emitir solicitações para um servidor de aplicação para cada ação no aplicativo. Para cada solicitação de pedido, o servidor gera uma resposta nova que permite ao usuário enviar um novo pedido para mais informações.

O *Flex* é usado somente para os desenhos dos *layouts*. As operações, por exemplo, de salvar, alterar, excluir informações no banco de dados ou gerar arquivos são realizados no servidor na linguagem Java.

Segundo SCMITZ (2010), o *framework Flex* e a linguagem Java são considerados líderes em seus respectivos segmentos. O *Flex*, por ser a melhor ferramenta para desenvolvimento de sistemas, sejam eles *web* ou *desktop* e o Java por ser uma linguagem sólida, focada exclusivamente na orientação a objetos.

2.1 Plataforma Java

A linguagem de programação Java é uma plataforma de computação lançada

pela primeira vez em 1995 pela *Sun Microsystems*, sendo uma tecnologia que possui programas de mais alta qualidade. O código fonte é compilado e transformado em *bycode* que em seguida é interpretado pela JVM (Máquina Virtual Java), sendo uma linguagem interpretada, proporcionando assim, maior portabilidade e flexibilidade para a linguagem, onde um programa escrito uma vez, deverá ser executado em qualquer sistema operacional, sem quaisquer alterações.

A necessidade de Java surge, pois existem muitos aplicativos e sites que funcionam somente com o Java instalado, e muitos outros aplicativos e sites são desenvolvidos e disponibilizados com o suporte a essa tecnologia. O Java é seguro, confiável e rápido. Devido a suas características a tecnologia Java é bem difundida e possui três ambientes de desenvolvimento conhecidos: o JSE, o JEE e JME.

JSE *(Java Standard Edition)* é o ambiente de desenvolvimento mais utilizado, isso porque seu uso é voltado computadores e servidores, onde há bem mais necessidade de aplicações. Além disso, pode-se dizer que essa é a plataforma principal, já que, de uma forma ou de outra, o JEE e o JME tem sua base aqui. (SUN, 2010a).

JEE (*Java Enterprise Edition*): é a plataforma Java voltada para redes, internet, intranets e afins. Assim, ela contém bibliotecas especialmente desenvolvidas para o acesso a servidores, a sistemas de e-mail, a banco de dados, etc. Devido essas características, o JEE foi desenvolvido para suportar uma grande quantidade de usuários simultâneos, sendo que a sua maior preocupação é com relação à segurança desse ambiente (SUN, 2010b). O JEE contém uma série de especificações, cada uma com funcionalidades distintas. Entre elas, tem-se:

- JDBC (*Java Database Connectivity*) é utilizado no acesso a banco de dados;
- JSP (*Java Server Pages*) é um tipo de servidor *Web*. Por ser baseado na linguagem de programação Java, tem a vantagem da portabilidade de plataforma, que permite a sua execução em diversos sistemas operacionais;
- *Servlets* é um componente do lado servidor que gera dados HTML e XML para a camada de apresentação de um aplicativo *Web*. É basicamente uma classe na linguagem de programação Java que dinamicamente processa requisições e respostas, proporcionando dessa maneira novos recursos aos

servidores

JME (*Java Micro Edition*) É um ambiente flexível direcionado para aplicações que serão executadas em dispositivos embarcados, móveis ou portáteis como celulares, PDA, TV, *set-top, boxes* e impressoras (SUN, 2010c). Como a linguagem Java já era conhecida e a adaptação ao JME não é complicada, logo surgiram diversos tipos de aplicativos para tais dispositivos, como jogos e agendas eletrônicas. As empresas saíram ganhando com isso porque, desde que seus dispositivos tenham uma JVM (Java Virtual Machine – Máquina Virtual Java), é possível, com poucas modificações, implementar os aplicativos em qualquer aparelho, sendo o único limite a capacidade do hardware.

A plataforma JME contém configurações e bibliotecas trabalhadas especialmente para a atuação em dispositivos portáteis. Assim, o desenvolvedor tem maior facilidade para lidar com as limitações de processamento e memória, por exemplo.

Conforme demonstra na imagem abaixo, a JEE suporta diversas especificações importantes como: JSP, EJB, *JavaMail*, JMS, *Web Services*, JSF 1.2 entre outras.

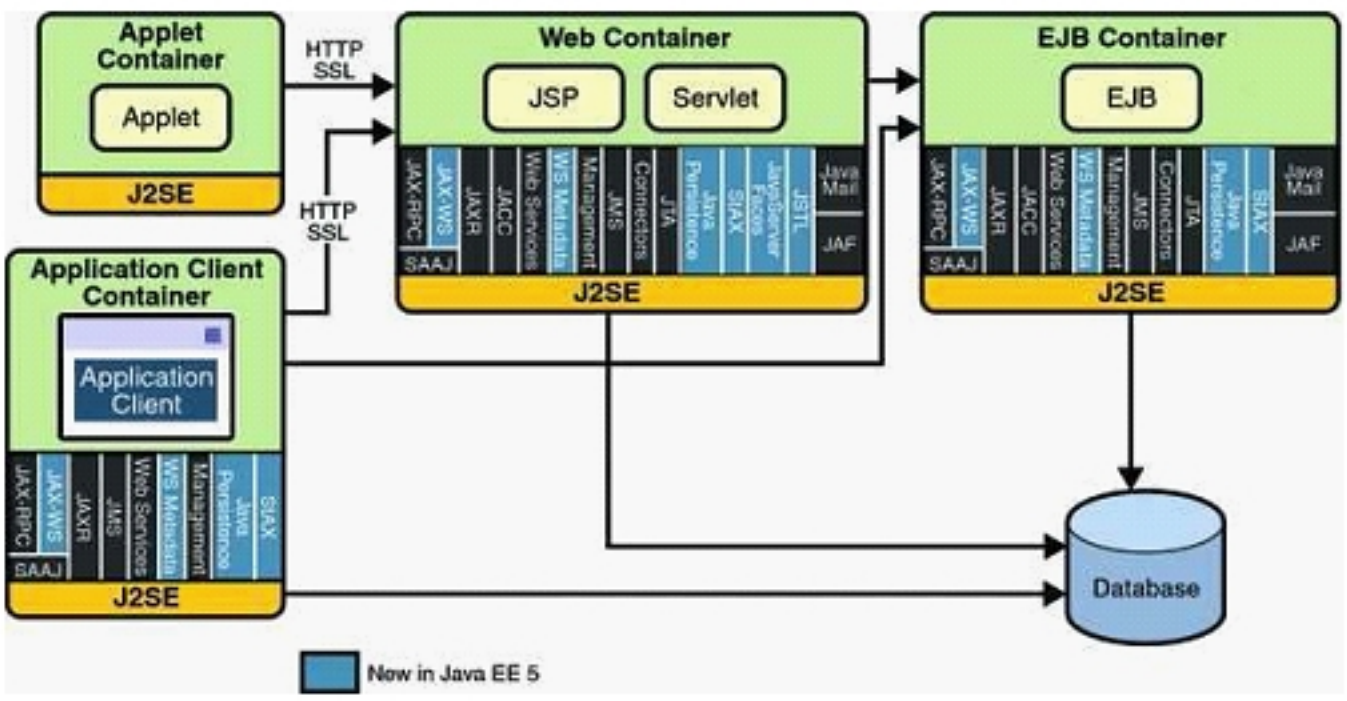

Figura 1 - Arquitetura da plataforma JEE e suas novas características
Fonte: SUN, 2009b.

O presente trabalho foi desenvolvido e focado na plataforma Java JEE, onde foram criados dois protótipos de exemplo para realizar as comparações entre os diferentes *frameworks* JSP e *Flex*.

2.1.1 Java Server Pages (JSP)

O JSP é uma tecnologia orientada a criar páginas *web* com programação em Java. A tecnologia JSP permite juntar o conteúdo estático HTML com o conteúdo dinamicamente gerado das *servlets*. JSP (*Java Server Pages)*, e em português é algo como Páginas de Servidor Java.

Com a ajuda do JSP podemos criar aplicações *web* que se executam em servidores *web*, de múltiplas plataformas. Por se tratar código HTML/XML misturado com etiquetas especiais para programar *scripts* de servidor em sintaxe Java, pode-se usar qualquer editor de texto para programar em JSP.

Para sinalizar a um servidor *Web* que processe e codifique os elementos presentes em tal página, necessitam que tenham a extensão *.jsp* ou *.jspx* com as *tags* específicas e combinadas com outras *tags* estáticas (HTML ou XML).

As *tags* XML e *scriptlets* são usadas pelo JSP e escritas na linguagem Java para encapsular a lógica que gera o conteúdo para a página, enviando as *tags* de formatação (HTML ou XML) de volta à página de resposta. Sendo assim, páginas JSP separam a lógica da página de seu *design* e de sua exibição.

Através de um formulário HTML são enviadas informações e são armazenadas em um objeto *request*, que as envia do cliente para um *container* JSP, conforme representa a figura abaixo.

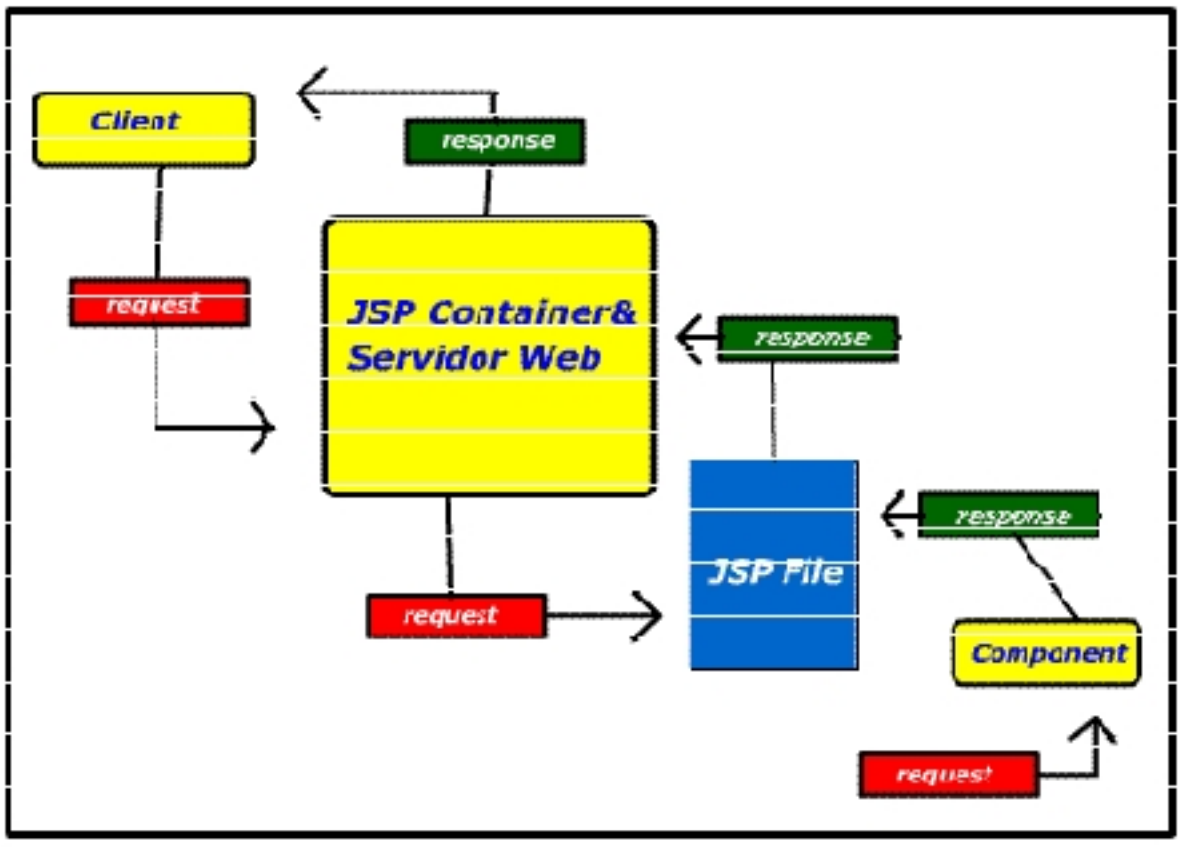

Figura 2 - Funcionamento do JSP e página Web
Fonte: Sierra, 2008

A figura acima mostra o container JSP, que envia o objeto *request* para o componente (*beans, enterprise bean ou servlet*), conforme especificado pelo arquivo JSP.

O *request* (informação chegando ao servidor *Web* a partir do navegador) recupera dados de um banco de dados ou de outro tipo de repositório, e envia um objeto *response* (informação saindo do servidor para o navegador) de volta para o container JSP. O container JSP passa o objeto *response* para a página JSP, onde os dados do objeto *response* são formatados de acordo com o *design* da página HTML. O *container* JSP e o servidor *Web* enviam a página JSP, já formatada, de volta para o usuário, onde o ele pode ver os resultados em seu *browser Web*, usando um protocolo de comunicação qualquer ou o HTTP.

A imagem abaixo mostra um exemplo simples de uma página JSP, e conversão dessa página em um *servlet*.

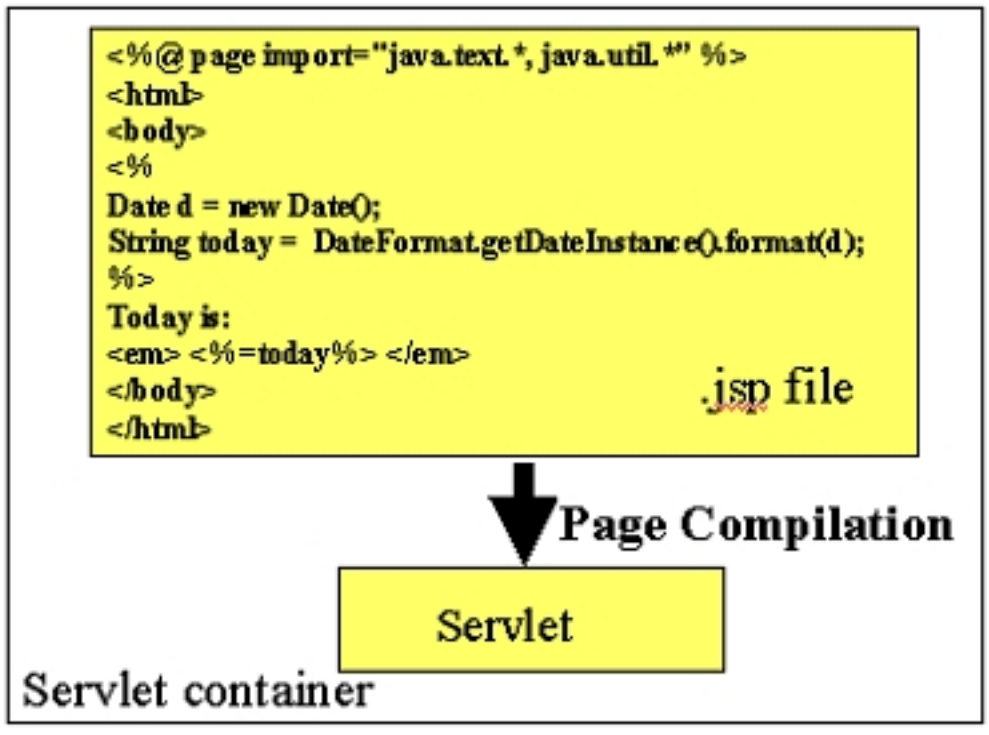

Figura 3 - Exemplo página JSP
Fonte: Sierra, 2008

Enfim, o JSP se aproveita de todas as vantagens e característica de uma linguagem orientada a objetos, por ser baseado na linguagem Java, como por exemplo, tratamento de exceções, herança, encapsulamento, tendo um código flexível e robusto.

2.2 Flex

Flex é um *framework* de desenvolvimento de aplicação que permite a criação de interfaces ricas para *web* e *desktop*. [SCHIMITZ, 2009]. É um *framework* de desenvolvimento de aplicações *web*, que utiliza o *Flash Player* do navegador para poder ser renderizado dentro do *browser*. É uma tecnologia *open sorce*, usado essencialmente para desenhar um sistema, substituíndo o *html* e o *Javascript*, trazendo mais recursos e bibliotecas prontas para o uso.

O *Adobe Flex* é uma plataforma de desenvolvimento de RIA's tendo como principal base de distribuição das suas aplicações o *Adobe Flash Player*. É um *framework* que permite a criação dinâmica de aplicações ricas e interativas para a internet. Pode-se usar aplicações usando o *Flex* SDK que é *Free* ou *FlexBuilder* que é um *software* IDE

para criação em modo visual no estilo arrastar e soltar, baseada no *Eclipse*. O *Flex* amplia metodologias já consagradas tais como XML, *Web Services*, HTTP, *Flash Player* e *ActionScript (*WELTER, 2008).

O Flex pode ser instalado em um servidor JEE ou em um *Servlet Container,* por ter muitas biblioteca de componentes, que podem ser programados através da MXML (*Macromedia Flex Markup Language*), uma linguagem baseada em XML e orientada a objetos. O *framework Flex* é divido conforme segue abaixo.

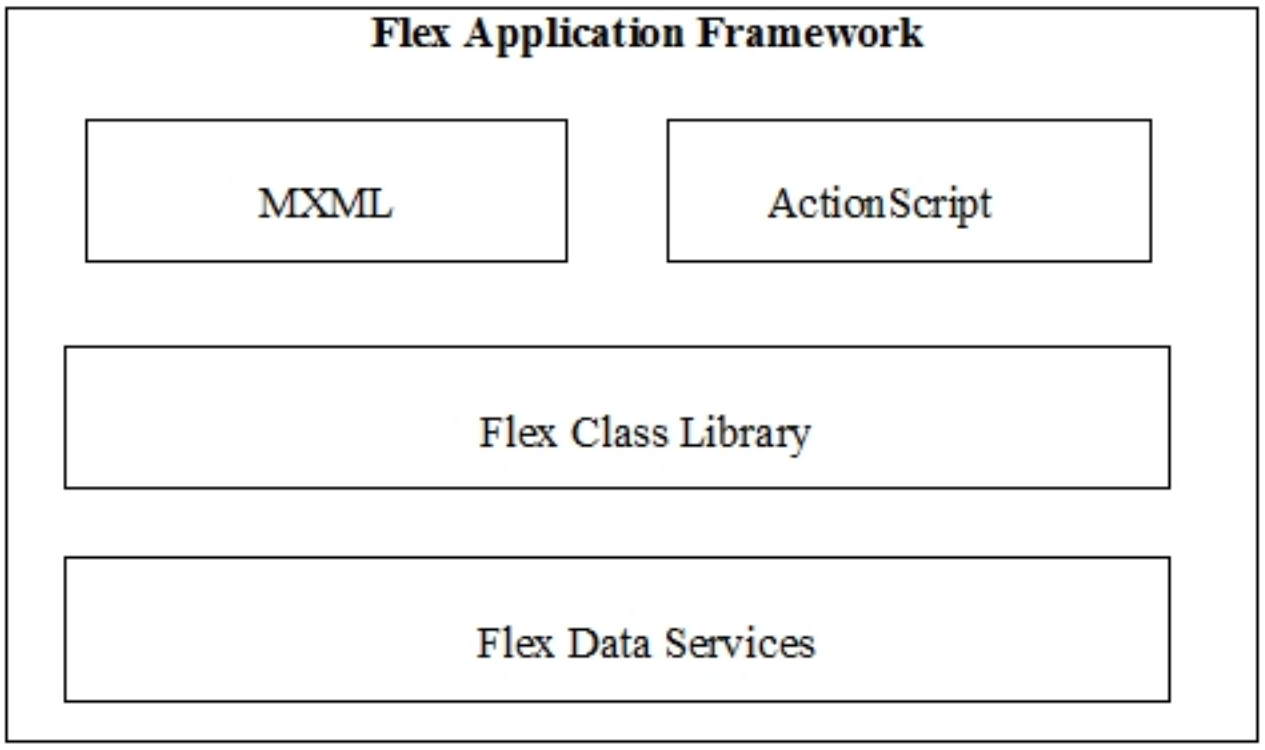

Figura 4- Framework de aplicação Flex
Fonte: Schimitz, 2009

2.2.1 Vantagens e Desvantagens do Adobe Flex

Além das vantagens do *Flex*, devem-se levar em consideração as desvantagens. Segue abaixo as vantagens e desvantagens de tal tecnologia levando em conta o levantamento teórico e a aplicação na prática, e a sua evolução.

A seguir segue algumas vantagens do Flex:

- Possui um ambiente de desenvolvimento integrado(IDE), que aumenta a produtividade no desenvolvimento de aplicações, pois com o uso da IDE o

programador arrasta e solta componentes na tela, programando apenas a funcionalidade da aplicação e não o componente;

- Produz uma interface rica para o usuário final da aplicação, ou seja, interfaces mais completas, funcionais e mais amigáveis ao usuário;
- Uma aplicação feita em *Flex* é semelhante às feitas para *desktop*, visando um uso mais fácil para o usuário com uma interface mais completa do que as aplicações *web* tradicionais, unindo a interatividade e a funcionalidade do *desktop* com a abrangência e flexibilidade da *web*;
- Possui uma comunicação direta com diversos *back-end*, permitindo que o desenvolvedor escolha qual linguagem a ser utilizada do lado do servidor;
- É orientado a objeto assim como o Java, utilizando assim objetos e todos os recursos do Java, tais como: encapsulamento, herança, entre outras e utilizando boas práticas de programação;
- Por ser compilado e executado pelo *Flash Player*, a aplicação não sofre alterações com as variações de padrões dos navegadores;
- Não há necessidade de atualizar da página, após uma submissão, como acontece no HTML (sem o uso do *Ajax*);
- Aplicações multiplataforma, compatíveis com várias plataformas e suporte para Windows, Mac e Linux.

Abaixo estão listadas algumas desvantagens com a utilização do *Flex*:

- É necessário a instalação de um plugin do Flash Player no navegador do cliente;
- Para que o Flash Player possa executar sem problemas, existe uma configuração mínima da máquina do cliente;
- Devido a necessidade de aprender outra linguagem de programação, o *ActionScript*, a curva de aprendizado é maior;
- Dependendo da conexão e da aplicação, poderá ocorrer um problema de lentidão no primeiro acesso do usuário, porque o arquivo *swf* da aplicação *Flex*, será gravado na máquina do cliente;
- O usuário pode estar visualizando uma versão antiga da aplicação, se não

for realizado um *download* do novo swf modificado, devido ao fato do arquivo *swf* gerado, ser armazenado em *cache* do navegador.

2.2.2 MXML - Macromedia Extensible Markup Language

O MXML é uma linguagem de marcação baseada em XML usada para definir a interface da aplicação, através de inclusão de componentes. Poderá ser usada para definir aspectos não visuais da aplicação como acesso a dados no servidor, e inclui *tags* para componentes visuais como *dataGrids, inputText, comboBox* e *menu.*

Dentro de uma aplicação *Flex*, o MXML é utilizado para o desenvolvimento e criação da interface e componentes da aplicação, assim como, o HTML fornece *tag* que define a interface com o usuário. Ela é muito explorada por ser uma linguagem simples e familiar aos desenvolvedores *web*. No conceito dessas aplicações utilizando *Flex*, ela representa a linguagem primária, seguida pelo *ActionScript.*

Utilizando apenas código é possível criar objetos e animações. Todas as aplicações criadas em *Flex* são compiladas e transformadas em formato *swf*, o mesmo do *Flash.* Para visualização de uma aplicação MXML é feita da seguinte forma: o usuário efetua uma requisição ao seu servidor, que compila o arquivo MXML para *swf* e este será executado no *Flash Player* que será salvo na máquina do cliente e visualizado no navegador.

As *tags* MXML são mapeadas diretamente para os objetos do *ActionScript*, e todo atributo dessas *tags*, remetem às propriedades do *ActionScript.* Portanto, tudo o que é possível realizar em uma aplicação *Flex* usando o MXML, é possível fazer as mesmas coisas usando o *ActionScript*, porém a facilidade que o MXML proporciona para criar componentes é melhor que o *ActionScript.*

2.2.3 ActionScript

ActionScript é uma linguagem de programação orientada a objetos usada para definir a parte lógica da aplicação *Flex*. Com base em *ECMAScript*, a linguagem de programação de padrões internacionais para *scripting*, o *ActionScript* avança ainda mais sua linguagem para oferecer aos desenvolvedores um modelo de programação para aplicativos aprimorados de Internet (RIA's). O *Flex* fornece aos programadores um novo caminho de desenvolvimento, com um fluxo de trabalho e um modelo de programação familiar aos desenvolvedores.

Para declarar script dentro de um arquivo MXML, ele deve estar dentro da tag <mx:Script> e com isso é possível realizar as seguintes funcionalidades:

- A interface definida pelo *Flex* é orientada a eventos, onde esses eventos, que podem ser tratados com funções em *ActionScript*. Um exemplo simples de um evento, é o clique de um botão;
- O *ActionScript* permite que o desenvolvedor manipule os erros ocorridos em tempo de execução e identificar erros nas validações de dados, enviando uma mensagem para o usuário ou qualquer outro evento que tenha sido definido na aplicação;
- Com os componentes base do *Flex* é possível desenvolver componentes específicos para as necessidades da aplicação que está sendo desenvolvida, os componentes do *Flex* são extensíveis, o que facilita o trabalho;
- Personalizar a aplicação a partir de métodos ou propriedades;
- Possibilidade de criar novos métodos em componentes da aplicação que já existem, adicionando neles novos comportamentos que são necessários para sua aplicação;
- Dentro de uma declaração MXML é possível fazer uma ligação de dados dos objetos para o *Flex*: recurso este, utilizado para o preenchimento dos dados das classes de modelo da aplicação. Essa ligação é a transferências dos dados inseridos pelo usuário para as classes definidas na aplicação *Flex*, que está ligada as classes do Java ou qualquer outra tecnologia de *back-end*.

Após a criação de todos os componentes em ambiente de desenvolvimento, os arquivos são compilados, empacotados e enviados para o servidor onde os usuários terão acesso à aplicação.

2.2.4 Flex Class Library e Flex Data Services

Flex Class Library e Flex Data Services são bibliotecas de classes do *Flex*. Inclui bibliotecas de classe pré-criada e serviços de aplicativos que ajudam os desenvolvedores a montar e criar aplicativos utilizando mais de 100 componentes avançados e pré-criados de aplicativos, além de novos componentes específicos. Tais serviços incluem o gerenciamento de arrastar e soltar, o vínculo de dados, o sistema de exibição que gerencia o layout da interface, o sistema de estilo que gerencia a aparência dos componentes de interface, e o sistema de efeitos e animação que gerencia o movimento e as transições.

Para que o *Flex* faça um uso intenso desses dados, necessita que o *Flex Data Service* ofereça um conjunto de avançados recursos de gerenciamento de dados no lado do servidor.

3 TECNOLOGIAS PARA O DESENVOLVIMENTO

A seguir serão apresentadas as características de algumas ferramentas para o desenvolvimento de RIA e o conjunto de novos conceitos e práticas que podem ser sempre utilizadas a favor das empresas e em prol de uma boa relação com os internautas. Na verdade, se bem utilizadas, só traz benefícios como: possibilidade de oferecer uma melhor experiência de navegação e uma interface mais rica, a possibilidade de o internauta participar efetivamente do site e estreitar o relacionamento com o cliente e fidelizá-lo.

3.1 Aplicativos Web

A *Web* trata-se de um conjunto de programas que é executado em um servidor de HTTP (*Web Host*). O desenvolvimento da tecnologia *web* está relacionado, entre outros fatores, a necessidade de simplificar a atualização e manutenção mantendo o código-fonte em um mesmo local, de onde ele é acessado pelos diferentes usuários. A *web* é um ambiente que possui uma problemática a entrega de aplicações, que exigem interfaces mais robustas e níveis mais altos de interatividade, levando a custos maiores de transmissão de dados. Define-se *web* como uma aplicação de software que utiliza a *web*, através de um *browser* como ambiente de execução.

Tudo que se é processado em algum servidor pode ser definido como *web*, exemplo: quando você entra em um site de *e-commerce* a página que você acessa, antes de vir até seu navegador é processada em um computador ligado a internet que retorna o processamento das regras de negócio nele contidos.

Outros aspectos favorecem a escolha de se desenvolver um aplicativo *web*, um deles seria, por exemplo, o custo de instalação do *software*.

Quando os aplicativos *desktop* são utilizados em centenas ou milhares

de computadores possuem um custo de instalação alto. Em aplicações empresariais, temos uma redução considerável no custo de *deployment* (nomenclatura utilizada para o *start* de uma aplicação *web*), na medida em que basta colocar a aplicação num servidor HTTP, e a partir daí milhares de terminais ficam com a aplicação disponível através do navegador, sem precisar que seja instalado em cada *desktop*.

Outra redução considerável é na manutenção, pois os custos podem ser reduzidos, visto que basta dar manutenção a um único ponto: o servidor HTTP, onde é disponibilizada a aplicação. Ao atualizar essa aplicação no servidor, todos os terminais são atualizados simultaneamente assim que acessados, sem precisar ir máquina por máquina atualizando.

Os aplicativos tradicionais cliente-servidor são substituídos pelos aplicativos *web*, porém, os resultados finais são limitados, decorrentes dos problemas de tempo de reposta dependendo da velocidade da aplicação e da rede. O RIA surgiu para tentar suprir essas limitações encontradas.

A segmentação da *web* pode ser dividida em: a *Web* 1.0, a implantação e popularização da rede e as convencionais páginas HTML. A *Web* 2.0, referencia-se o conteúdo que se tem disponível nos dias atuais, centrada nos mecanismos de busca como os *sites* de busca, os *sites* de colaboração. A *web* 2.0 surgiu pela necessidade natural de evolução, visando principalmente aproximar o usuário dos aplicativos através da fácil comunicação e interatividade entre estes.

3.1.1 WEB 1.0

A *Web* 1.0 se estende desde a década de 90 e pode ser definida como a primeira fase da *Web*. É caracterizada por um "*read-only web*", em que o internauta pertence ao papel de mero espectador, sem ter condições de desenvolver o conteúdo dos sites visitados.

As aplicações *Web* tradicionais se enfrentam com um desafio substancial: elas sofrem com a inabilidade de poder de maneira adequada, visualmente representar as

necessidades requeridas pelos aplicativos atuais. (CABRERA *apud* LASZLO SYSTEMS, 2006).

A limitação do modelo foi usado para levar à necessidade de se desenvolver uma variedade de algoritmos sofisticados para trabalhar com *cache*, assim como mecanismos e práticas de codificação, buscando melhorar a transferência de conteúdo das páginas, para reduzir o tempo de *refresh* e utilização da banda de conexão. Com o passar dos anos foram proporcionadas novas características dinâmicas às páginas *Web* para melhorar a interatividade com o usuário como a capacidade de conter mídias, animações e acesso a base de dados, mas mesmo assim se viu limitada no que diz respeito ao que pode ser apresentado.

Segundo COPPARI (2006), apesar dos recursos limitados da *web* 1.0, este fator não influenciou no crescimento dos aplicativos *web* devido às vantagens do desenvolvimento baseado em servidor.

3.1.2 WEB 2.0

A *Web* 2.0 define-se como segunda geração da internet, marcada pela interatividade, pelos conteúdos gerados por usuários e pela personalização de serviço. Representa a segunda década da *web* (2000-2009), que é caracterizada por uma mudança no uso da *web*, não só para se conectar a uma empresa ou componentes do produto, dando-lhes informação, mas também permite aos utilizadores ligar para a empresa e um para o outro.

Com a *web* 2.0, o conteúdo dos *sites* sofreu grande impacto, permitindo ao usuário a possibilidade de interatividade gerando e organizando informações. O usuário possui também a flexibilidade para enriquecer conteúdos através de personalização, comentários e avaliações.

Qualquer pessoa pode usar uma parte do programa para fazer outro programa, pois os programas são abertos. São utilizadas APIs para deixar que outros *sites* utilizem partes dos seus dados nos serviços deles. Em vez de grandes servidores provendo uma

enorme quantidade de arquivos, descobriram-se as redes P2P (*Peer-to-peer)*, na qual cada usuário é um servidor de arquivos e os arquivos (áudio, vídeo, texto e outros formatos) são trocados diretamente entre eles sem possuírem relação de cliente/servidor (COSTA, 2008).

3.2 RIA (Rich Internet Application)

RIA é a abreviação de Aplicações Ricas para *Internet* (*Rich Internet Application*). É uma aplicação com interface *web* que contêm características e funcionalidades de uma aplicação *desktop* tradicional, sendo, entretanto distribuídas do lado servidor. Esta tecnologia utiliza técnicas que fornecem uma nova classe de sites mais interativos e que atendem a exigência do usuário, fazendo parte do processamento da interface da aplicação do lado do cliente (no navegador *web*).

Com o uso da tecnologia RIA algumas características, tais como a de "arraste e solte" ou a utilização de uma barra para alterar dados ou a realização de cálculos que não precisam ser enviados de volta para o servidor, são consideráveis benefícios da tecnologia.

Outra característica positiva do uso do RIA é ter o melhor tempo de resposta nas ações executadas, levam ao usuário a sensação de estarem utilizando uma aplicação *desktop*. A carga de processamento entre o cliente e servidor torna-se mais equilibrada, visto que o servidor *Web* não necessita realizar todo o processamento e enviar para o cliente, permitindo que o mesmo servidor possa lidar com mais sessões de clientes.

A redução do fluxo de dados na rede ocorre, porque um *client engine* pode ter uma inteligência embutida maior do que um navegador da *Web* padrão quando decidir quais os dados que precisam ser trocados com os servidores, podendo acelerar a solicitações individuais ou reduzir as respostas, porque muitos dos dados só são transferidos quando é realmente necessário, e a carga global da rede é reduzida. Entretanto, o uso destas técnicas pode neutralizar, ou mesmo reverter o potencial desse benefício. Isto porque o código não pode prever exatamente o que cada usuário irá fazer

em seguida, e é comum que tais técnicas como baixar dados extras, para muitos ou todos os clientes, cause um tráfego desnecessário (WELTER, 2008).

A imagem abaixo apresenta um comparativo entre as aplicações baseadas em *desktop* e *web*, onde as setas coloridas em verde demonstram as vantagens das aplicações e as setas de cor laranja demonstram as desvantagens, segundo (AHMED, 2009).

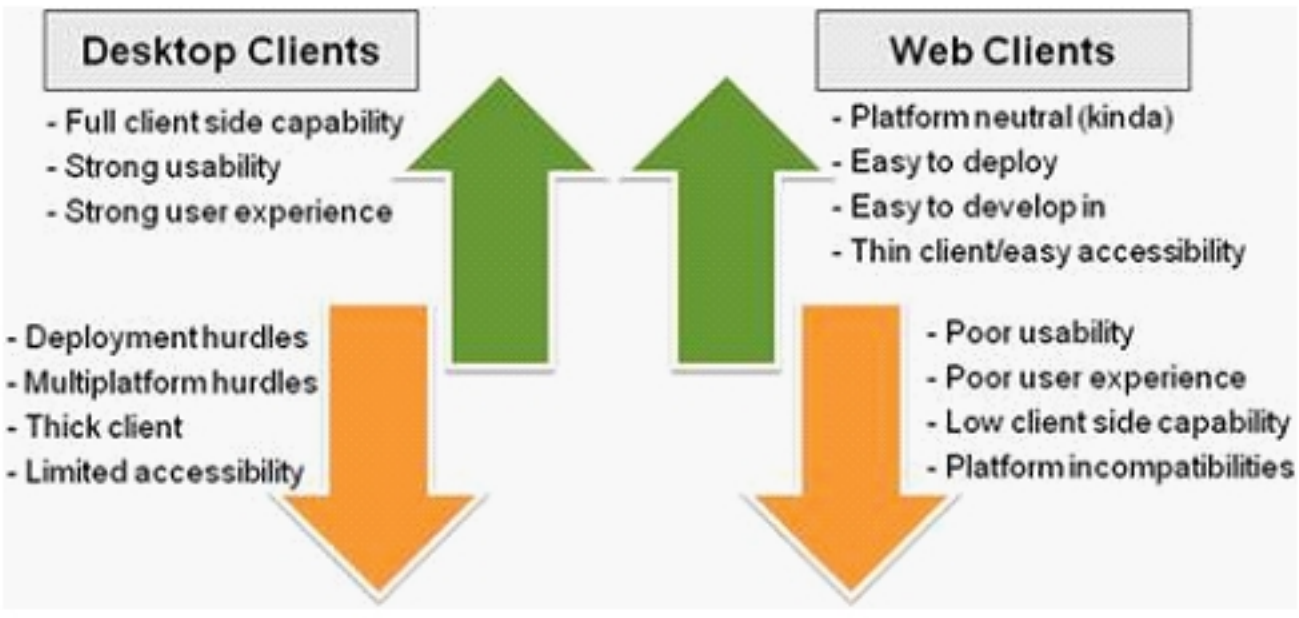

Figura 5 - Vantagens e desvantagens de aplicações desktop e web
Fonte: Macoratti, 2011

Conforme figura acima, as vantagens do ambiente *web* podem ser relacionadas como:

- A grande maioria dos usuários já estão acostumados com o funcionamento dos navegadores e da interface;
- Para alterações e manutenções de aplicações, basta colocá-las no servidor e os usuários já estarão com a versão da aplicação correta sem que, para isso tenha que ir a diversos computadores diferentes atualizar;
- Caso exista a necessidade de aumentar o poder de processamento, basta fazer isto no servidor.

A falta de padronização entre os diversos navegadores pode ser um problema da aplicação *web*, pois a aplicação poderá 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.

Outro problema do uso da aplicação web ocorre devido à grande quantidade de navegadores, a aplicação pode não ficar tão elegante como você imagina. A *interface* HTML não é rica em controles gráficos e peca no quesito posicionamento.

3.3 Ajax

O *Ajax* é metodologia de desenvolvimento para criar *softwares web* com interação, e facilita a comunicação síncrona e assíncrona entre uma página e o usuário, ou seja, navegue em vários conteúdos ao mesmo tempo, sem ter que esperar enquanto o *site* processa as informações para executar outra ação.

O *Ajax* usa combinações de linguagens para que juntamente com *Javascript* possa dinamizar a exposição das informações e interagir com a apresentação da informação realizando a troca e manipulação dos dados assincronamente com um servidor *web*. O *Ajax* é uma metodologia de grande valia para auxiliar no desenvolvimento *web*, mas possui um grande problema, por ser *Javascript*, há o risco de que algumas funcionalidades não sejam interpretadas por todos navegadores.

Algumas vantagens do *Ajax*:

- Interatividade: com *Ajax* a maior parte da aplicação e realizada na parte no lado do cliente, isso faz com que a aplicação se torne interativa para o usuário, com ações como efeitos visuais, arrastar e soltar e outros se assemelham com os aplicativos *desktop*.
- Redução de consumo de banda: *Ajax* reduz a quantidade de requisição enviada entre cliente e servidor, pois o *Ajax* não reenvia toda a página de *interface*, e sim apenas dados da aplicação.
- Redução de processamento no servidor: algumas aplicações precisam realizar alguns procedimentos do lado do servidor, mas com o *Ajax* é possível realizar alguns desses processos para o lado do cliente.
- Não é proprietário: *Ajax* não é um produto ou marca, é apenas o nome dado à utilização de um conjunto de tecnologias *Web* padrões existente.

- Portabilidade: Não preciso instalar nenhum *software* ou *plugin* no cliente, pois é um conjunto de tecnologias, que não se restringe a um *Web Browser*.

Algumas das desvantagens do *Ajax*:

- Código fonte desprotegido: se a página for simples, através da opção mostrar código fonte da página poderá ver e copiar métodos importantes;
- Parte da Lógica de Negócios visível: como a parte lógica do Ajax é movida para o navegador, ficará visível para o usuário;
- Não funciona se o *Javascript* estiver desabilitado, na configuração do navegador do usuário;
- Desempenho do cliente: Com *Ajax* os muitos procedimentos são transferidos do servidor para o cliente, podendo sobrecarregar o cliente.

3.4 SilverLight

A tecnologia *Silverlight* é uma ferramenta criada pela empresa *Microsoft* com o propósito de entrar no mercado da *web* 2.0. Foi lançado em 2007 para fazer frente com a metodologia do *Adobe Flex*. É formado por páginas .XAML e arquivos de comportamento para o *Code Behind* (em VB.NET ou C#), implementando ações disparadas pelo usuário. A solução, uma vez compilada, gera um *assembly* .NET, que deve ser executado no *browser* cliente, através do *plug-in* de *Silverlight* que deve ser instalado no navegador para que a aplicação *silverlight* possa ser executada.

Esta ferramenta suporta a construção de interfaces ricas para internet, com interatividade e conteúdo multimídia. Outro aspecto do *Silverlight* é sua portabilidade através de um *plug-in*. As aplicações *Silverlight* podem ser executadas em praticamente todas as versões de *browsers* do mercado, como *Internet Explorer, Firefox* (para Mac e PC), *Safari*, entre outros. A figura abaixo é um exemplo do processo de criação e execução do *Silverlight*.

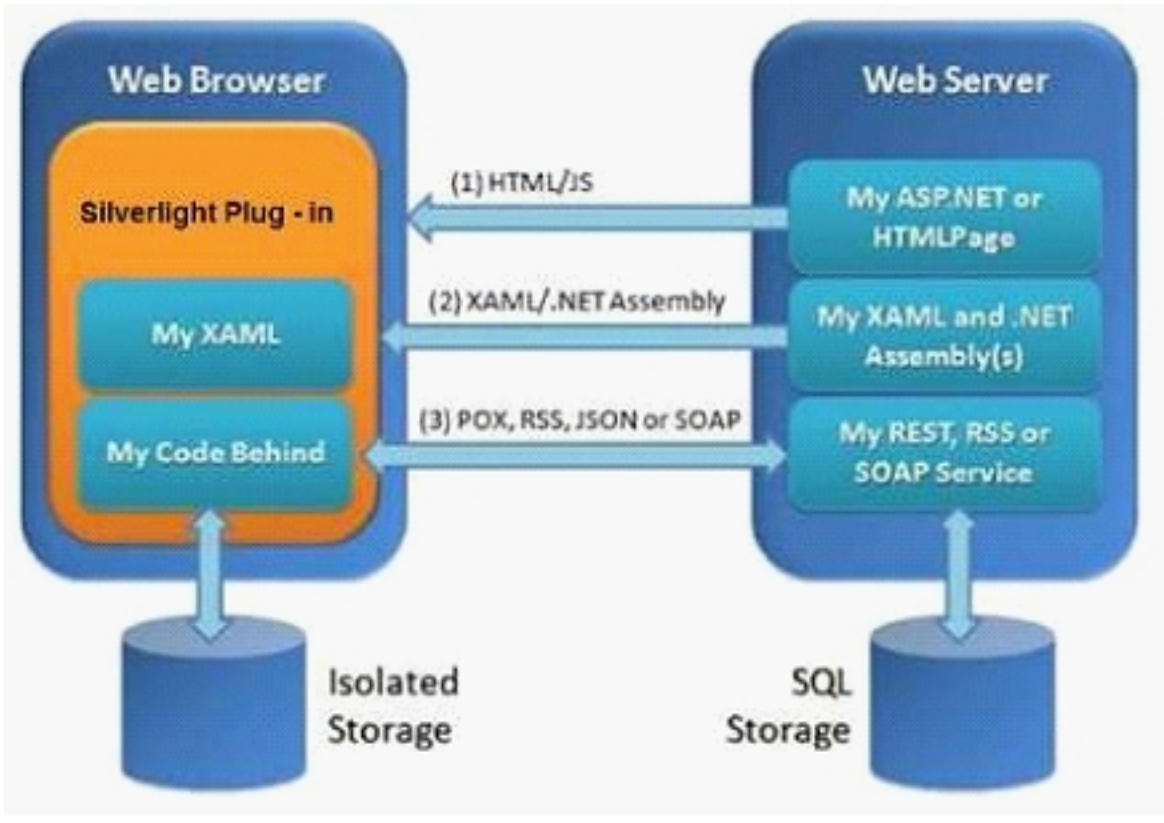

Figura 6 - Interações e Componentes com Silverlight
Fonte: Cambiucci, 2008.

3.5 Flash Builder

Adobe Flash Builder é um ambiente de desenvolvimento integrado (IDE), construído sobre a plataforma *Eclipse* que acelera o desenvolvimento de aplicações ricas para Internet (RIA's).

É um *framework* que oferece várias facilidades ao programador e têm como característica acelerar o desenvolvimento de aplicativos devido aos recursos oferecidos, além de ser bastante intuitivo. Seus componentes podem ser arrastados até a tela, configurando os parâmetros necessários e depois desenvolvendo a programação via código. De certa forma, pode-se assemelhar ao Borland *Delphi* que possui essas funcionalidades de componentes prontos e programáveis, mas no caso, com componentes voltados para a *Web*.

O *framework Flash Builder* pode ser configurado para compilar o código automaticamente e gerar o arquivo *SWF*, além de também criar outros códigos relevantes ao cliente. O maior problema é que esta ferramenta possui um custo, mas pode ser

adquirido gratuitamente para estudantes e desempregados.

3.6 JavaFx

A metodologia *JavaFx* é uma nova estratégia da empresa *Sun*, para levar o Java ao desenvolvimento interfaces ricas com o usuário, que atendam as exigências dos mesmos tendo interfaces amigáveis e de fácil acesso para dispositivos móveis, como celulares, televisores e navegadores. De acordo com a *Sun*, a família de produtos *JavaFX* tem como visão "a capacidade de criar conteúdo interativo, aplicações e serviços para *desktop* e aplicações para dispositivos móveis".

Com a utilização de *scripts* declarativos simples, podem-se acessar as APIs do *Swing* e do AWT para criar interfaces ricas de forma mais prática do que hoje. Apesar de ter *scripts* declarativos, a linguagem *JavaFX* é totalmente orientada a objetos, com métodos (chamada a operações e funções em *JavaFX*) e atributos (WEAVER, 2007).

Os conceitos de outras linguagens são incorporadas pelo *JavaFX*, incluindo *Java, Javascript, XML, SQL, ActiveScript*, SVG e *AspectJ*. Na definição de *GUI* (*Graphical User Interface*), o forte da *JavaFX Script* é o estilo declarativo, em que a estrutura visual se reflete diretamente na linguagem de programação.

3.7 Adobe Flex

O *Adobe Flex* é uma metodologia que possibilita a construção rápida de aplicações que são executadas no *Flash Player*, um ambiente que permite a execução de programas de interfaces com interfaces de usuário e interatividade sofisticadas.

É uma plataforma de desenvolvimento de RIA's tendo como principal base

de distribuição das suas aplicações o *Adobe Flash Player*. É um *framework* que permite a criação dinâmica de aplicações ricas e interativas para a internet. Pode-se usar aplicações usando o *Flex SDK* que é livre ou *FlexBuilder* que é um *software* IDE para criação em modo visual no estilo arrastar e soltar. O *Flex* amplia metodologias já consagradas tais como XML, *Web Services*, HTTP, *Flash Player* e *ActionScript*. (WELTER, 2008)

O *Flex* é uma forma de apresentar os recursos de interação rica entre o usuário e aplicação, ou seja, é uma camada de apresentação.

Usando o *Flex*, não há necessidade de testar o navegador do usuário, pois o código é executado através de uma máquina virtual. No *Flex* não sofre nenhuma mudança de *layout*, não importa qual navegador esteja sendo utilizado, pois o código é compilado em arquivos swf. Caso este que geralmente não acontece no desenvolvimento com AJAX, pois a compatibilidade do navegador em algumas implementações é difícil de obter e, para isso, deve-se geralmente efetuar testes para cada navegador.

4 O PROTÓTIPO – ESTUDO DE CASO

Para realizar o comparativo entre as metodologias foram desenvolvidos dois protótipos, que consistem em uma implementação de um sistema de CRUD, definido como as quatro operações básicas utilizadas em um banco de dados, como: inserção, exclusão, pesquisa e alteração de dados. Os protótipos possuem as funcionalidades básicas, pois o objetivo deste trabalho é a comparação das tecnologias entre si, com uma visão prática e crítica de todas as adversidades encontradas no desenvolvimento de uma aplicação com cada uma das tecnologias, lembrando que o foco é o desenvolvimento de interfaces ricas para a *internet* (*Rich Internet Applications*) e não os recursos fornecidos pelo protótipo em si.

4.1 Características dos Protótipos

Os protótipos desenvolvidos foram baseados em estudos sobre as duas tecnologias estudadas (*Flex* e Java), onde serão demonstradas as diferenças entre cada tecnologia e também as melhorias adquiridas, vantagens e desvantagens de cada tecnologia.

Tanto o protótipo 1 usando a tecnologia JSP, quanto o protótipo 2, utilizando Java e *Flex*, foram desenvolvidos com as seguintes funcionalidades:

- Tela de cadastro: responsável por armazenar todos os dados;
- Tela de Pesquisa: responsável por buscar um dado cadastrado no banco de dados;
- Tela de Exclusão: responsável por excluir um dado do banco de dados;
- Tela de Alteração dos dados: responsável por alterar os dados do cadastro.

O protótipo 1, foi desenvolvido utilizando a tecnologia JSP. O protótipo 2 foi desenvolvido utilizando a tecnologia *Flex*, sempre mantendo as característica e regras da aplicação. Os protótipos foram desenvolvidos apenas para fins de pesquisa, não contemplando a análise e uma estrutura com regras de negócio de um sistema real.

Para o desenvolvimento dos protótipos, foi utilizado o *Eclipse*, que é um ambiente de programação e desenvolvimento livre (usa-se IDE ou ambiente de desenvolvimento integrado) utilizado extensivamente por programadores, especialmente desenvolvedores Java [JACOBS, 2008]. O *Eclipse* é muito utilizado por desenvolvedores Java, a sua verdadeira eficácia é a capacidade de acomodar plugins para uma variedade de linguagens de programação. O *plugin Flex* SDK foi adicionado e usado para o desenvolvimento do protótipo dois.

4.1.1 Protótipo 1 – CRUD com Tecnologia JSP

O protótipo um(1), foi desenvolvido utilizando a tecnologia JSP (*Java Server Pages*), sendo uma tecnologia orientada para criar páginas *Web*. Junto com JSP pode ser usado AJAX (*Asynchronous Javascript* e XML) para as páginas mais interativas com o usuário. O grande problema é que com o aumento da aplicação, essas bibliotecas começam a falhar, a manutenção de telas fica difícil, páginas com códigos enormes e *Javascript* no JSP, e também incompatibilidade de *browsers*.

A figura 7 e a 8 apresentadas abaixo mostram a tecnologia JSP empregada no protótipo. A imagem 7 mostra um exemplo da aplicação JSP para listar os dados cadastrados e a manipulação de 'Editar' e 'Excluir'.

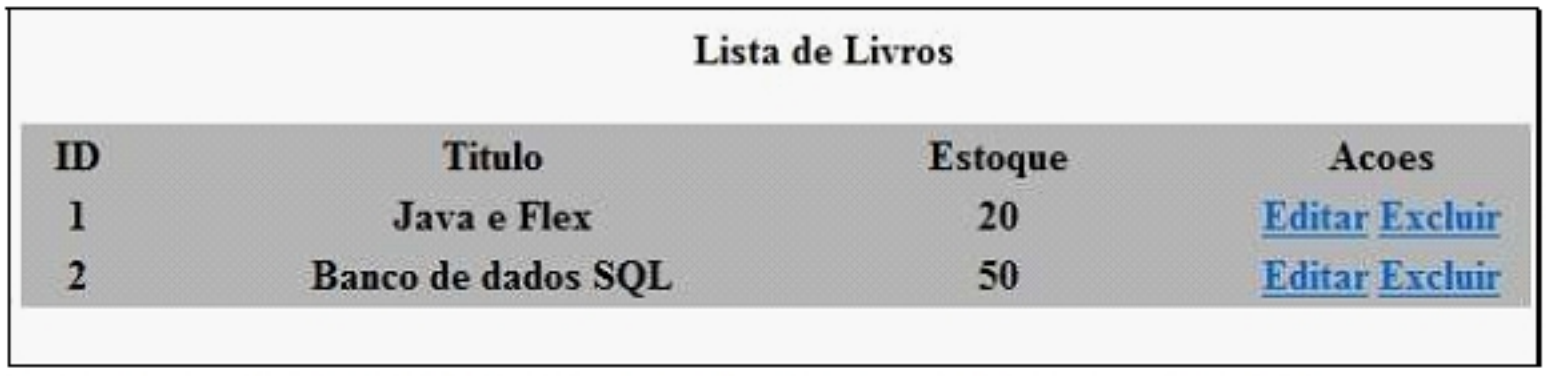

Figura 7 - Exemplo de Tela de Lista de dados usando tecnologia JSP
Fonte: o autor

A imagem 8 abaixo, mostra uma tela de Cadastro de dados, utilizando o JSP com o uso de *CSS* para melhor aparência da tela.

Figura 8 - Exemplo de Tela de Cadastro de dados
Fonte: o autor

Os arquivos Java do protótipo em JSP são arquivos acessados através de páginas dinâmicas, ou seja, através do JSP, onde é feita toda a manipulação do banco de dados.

4.1.2 Protótipo 2 – CRUD com Tecnologia Java e Flex

No protótipo dois, foi utilizado a tecnologia *Flex* para o desenvolvimento do CRUD. O uso do *Flex* facilita muito o desenvolvimento, pois existem muitas opções de componentes prontos, componentes que o JSP não possui, e nem o *Ajax* possui. E mesmo com os componentes o *Ajax* leva mais tempo em relação ao *Flex*, a manutenção é mais difícil, já que *Ajax* é baseado em componentes para criar componentes, o que deixa as páginas em JSP com muito código em *Javascript*.

O protótipo desenvolvido em *Flex* está mais padronizado e organizado e o tempo desenvolvimento foi relativamente menor do que o protótipo em JSP, pelo fato do *Flex* ter uma IDE de desenvolvimento, facilitando assim a produtividade do programador. O tempo varia dependendo do tamanho e da complexidade do projeto. As telas a seguir são do protótipo desenvolvido em *Flex*.

Figura 9 - Tela de Cadastro em Flex
Fonte: o autor

Figura 10 - Tela de Busca dos dados em Flex
Fonte: o autor

Figura 11 - Tela de Editar Dados em Flex
Fonte: o autor

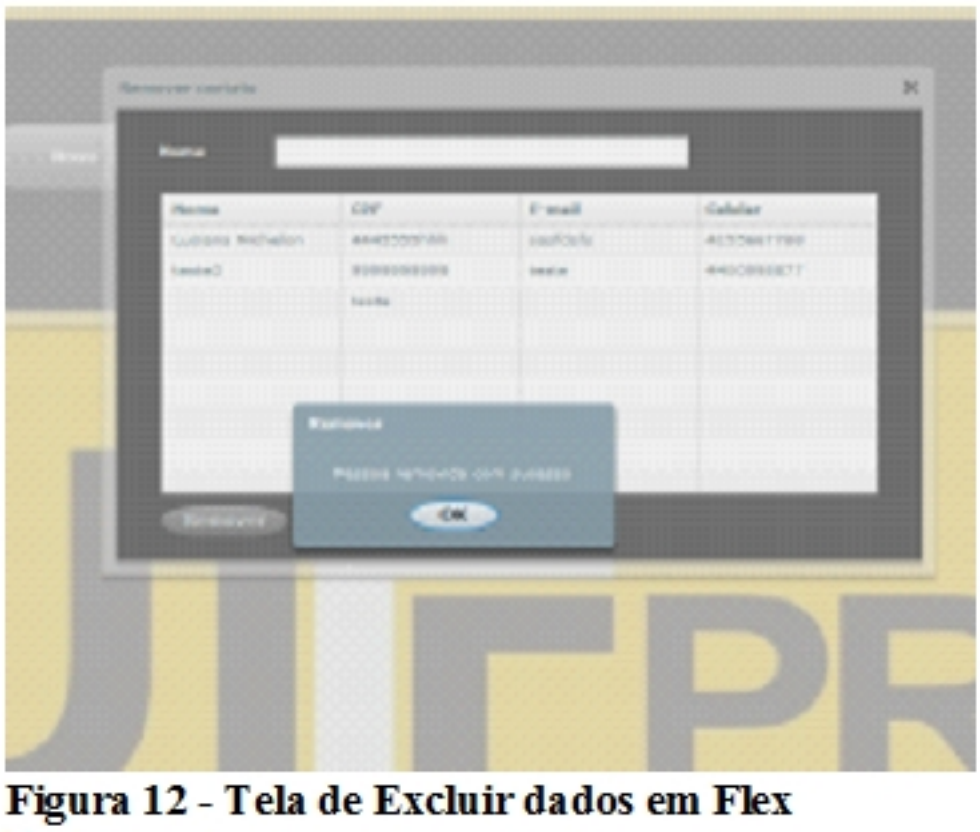

Figura 12 - Tela de Excluir dados em Flex
Fonte: o autor

As imagens acima representam o protótipo de integração *Flex* com Java. O *Flex* surge para facilitar a vida do programador através da utilização da IDE e o uso de componentes já existentes, inserido em poucas linhas de código, trazendo rapidez e organização do projeto.

4.2 Vantagens e Desvantagens das Tecnologias

Conforme estudo realizado nesta monografia, as características na prática exemplificadas no protótipo serão mostradas e comparadas a seguir.

Levando em conta as tecnologias utilizadas para o desenvolvimento dos protótipos, pode-se observar que com a mudança de tecnologia de JSP para *Flex*, ocorrem as seguintes vantagens:

- Facilidade para a criação de interface: com a ajuda do *Flex*, é muito mais fácil e prático de criar interfaces elaboradas, ao contrário do JSP, que

necessita de muitas alterações no HTML e nos estilos (CSS) para obtenção dos mesmos recursos.

- Não ocorre incompatibilidade de navegadores: toda a interface é executada pelo *plugin Flash*, ou seja, basta ter o *plugin Flash* instalado no navegador se têm certeza que o programa vai possuir exatamente a mesma aparência projetada, ao contrário de JSP, que ao utilizar *Ajax* ou *Javascript*, pode possuir aparências diferentes dependendo do navegador onde são executadas.
- Estrutura do projeto bem definida: a interface e a regra de negócio podem ser separadas de forma mais simples, facilitando a manutenção. Com o JSP, a união de código em HTML e *Javascript* tornam o código mais carregado de informações, ficando mais difícil o manuseio.
- Manutenção do sistema mais fácil: como o *Flex* possui muitas opções de componentes prontos, torna-se mais simples a manutenção, ao contrário do JSP que não possui essas facilidades, a não ser com o uso do *Ajax* e/ou *Javascript*.
- Produtividade maior: com o uso de componentes prontos e IDE de desenvolvimento, fica mais fácil e rápida a implementação, aumentando assim a produtividade. Não é necessário refazer funcionalidades que já existem em componentes prontos.
- Códigos mais enxutos: por utilizar componentes já existentes, o código fica mais limpo, menor e mais fácil de entender.
- Não utilização de bibliotecas externas: o *Flex* só utiliza componentes para algumas funcionalidades, o que não necessita adicionar bibliotecas externas, o que ocasionariam páginas complexas, falta de padronização e difícil manutenção.
- O *Flex* é rico em recursos visuais: retorno para o usuário com o que está acontecendo, facilidade de criar layouts como efeitos especiais e a utilização de *skins* personalizados, que servem para deixar a sua aplicação com um visual único.

Apesar de todas as vantagens descritas acima, deve-se considerar também

algumas desvantagens da mudança de tecnologia:

- Necessidade de *plugin*: O *Flex* tem a necessidade de instalação de um *plugin* chamado *flash player* no navegador do cliente, mas alguns navegadores não têm suporte ao *plugin,* ao contrário do JSP que não necessita de *plugin*.
- Alto Processamento: necessita de uma máquina de grande capacidade de processamento para que consiga trabalhar sem que haja problemas de lentidão, compilação e execução. Para as máquinas dos clientes exige uma configuração mínima para que o *plugin* possa executar sem ocorrer problemas.
- Necessita de aprendizado maior: pela necessidade de se aprender outras linguagem como o *ActionScript*, o que ocasiona tempo maior na curva de aprendizagem.
- *Cache* em navegadores: em alguns casos o arquivo *swf* gerado é armazenado no *cache* do navegador, no momento do primeiro acesso do usuário. Se for feita alguma alteração neste arquivo, a nova funcionalidade só será visualizada pelo usuário quando o mesmo fizer a limpeza do *cache* de seu navegador.
- Custo da IDE do *Flex*: Para ter um desenvolvimento mais produtivo e rápido em *Flex*, é importante usar a IDE *Flash Builder,* que facilita de todas as formas possíveis o desenvolvimento, e por sua vez é paga. O SDK não é pago, porém a dificuldade de implementação é maior, levando um maior tempo de desenvolvimento.

Portanto, a demonstração dos protótipos das tecnologias foram suficientes para mostrar as facilidades de cada tecnologia, onde o projeto torna-se padrão, as manutenções dos códigos são mais fáceis e a produtividade aumenta, pois no Flex, os componentes de layout estão quase prontos.

As dificuldades encontradas durante o estudo e desenvolvimento dos protótipos foram:

- Falta de documentação disponível: o *Adobe Flex* , não possui muitos materiais como: livros, tutoriais, artigos, disponíveis para consulta,

principalmente na língua portuguesa, já para o JSP encontra-se muito material.

- Aprendizagem: como o JSP é parecido com o HTML, a aprendizagem foi rápida, porém com o *Flex* foi mais lenta, devido à necessidade de aprender uma nova linguagem de programação.
- Dificuldade de configuração: a dificuldade de configuração do *plugin* no *Eclipse* para o desenvolvimento do protótipo em *Flex*. Dependendo da versão do *Eclipse* a *SDK* do *Flex* não funciona.
- IDE paga: devido ao fato da IDE ser paga, foi utilizado o SDK, configurado direto no *Eclipse*, porém ao ultrapassar o período de 60 dias(período Trial) para o desenvolvimento, seria necessário pagar pela ferramenta.

5 CONCLUSÕES

Devido ao grande crescimento de aplicações web, a procura com aplicações desenvolvidas voltadas para ela aumentou significativamente. A plataforma JEE foi uma das que mais se destacaram, e com ela surgiram diversos frameworks para a criação da camada de apresentação dessas aplicações e com eles a complexidade para o desenvolvimento também aumentou.

Os frameworks apresentados vêm para suprir essas necessidades por interfaces cada vez mais interativas e intuitivas para os usuários, os protótipos desenvolvidos ajudaram na avaliação das tecnologias utilizadas por cada um.

Ambas as tecnologias, JSP e *Flex*, tem suas características em particular. Para melhor aproveitamento, a melhor utilização é tirar proveito dos componentes e interatividade do *Flex* e também, da simplicidade na apresentação de grandes quantidades de conteúdo em texto do JSP e AJAX.

Levando em conta o desenvolvimento deste trabalho pode-se afirmar que o conceito de aplicações ricas para a Internet é uma alternativa válida para se criar aplicativos elegantes, com interfaces amigáveis e fáceis de serem utilizadas, baseando-se na plataforma *Web*.

A ferramenta *Adobe Flex Builder* mostrou-se ser um editor visual que beneficia o desenvolvedor no quesito produtividade, pois permite a construção de telas com a utilização apenas do *mouse*, arrastando componentes e configurando suas propriedades. Sendo assim, acelera o desenvolvimento da aplicação, mostrando ser uma opção de um novo mercado de trabalho especializado.

Como sugestão para trabalhos futuros, recomenda-se o desenvolvimento de módulos, como por exemplo, módulo de compras, módulo *ecommerce*, módulo financeiro, para posteriormente desenvolver um sistema *Flex* altamente complexo utilizando somente os arquivos swf.

6 REFERÊNCIAS

ALUR Deepak; CRUPI, John; MALKS, Dan. **Core J2EE Patterns: Best Practices and Design Strategies.** Palo Alto: Pearson Education, 2001. 496 p.

ARNOLD, Ken; GOSLING, James; HOLMES, David. **A Linguagem de Programação Java**. 4.ed. Porto Alegre: Bookman, 2007. 800 p.

CAMBIUCCI , Waldemir. Microsoft Silverlight: uma arquitetura RIA. Disponível em: http://blogs.msdn.com/wcamb/archive/2008/05/12/microsoft-silverlight. Acesso em: 06 de outubro de 2008.

COPYRIGHT, Adobe Systems Incorporated – **Estrutura do Flex**. Disponível em http://www.adobe.com/br/products/flex/flex_framework/. Acesso em: 18 de maio de 2011.

COPPARI, Jose Eduardo Rojas – **Desenvolvimento de uma Aplicação Web Utilizando a Plataforma Laszlo seguindo os Conceitos Rich Internet Application** – Monografia – UTFPR – Universidade Tecnológica Federal do Paraná, Campus Medianeira. Paraná: 2006.

JACOBS, Sas; De Weggheleire, Koen. **Fundation Flex for Developers: Data-Driven Applications With PHP, ASP .NET, Coldfusion And Lcds**. United States, Friendsof, 2008.

LABRIOLA, Michael; TAPPER, Jeff; TALBOT, James e BOLES, Matthew – **Adobe FLEX 3 : Treinamento Direto da Fonte**. Editora Alta Books, 2009.

LISBOA, Natalia - **Breve comparação dos termos Web 1.0, 2.0.**. Disponível

http://turma4b20092.bligoo.com/content/view/641553/Breve-compara-o-dos-termos-Web-1-0-2-0-3-0-e-4-0.html. Acesso em: 23 de maio 2011.

MACORATTI, José Carlos -**Desenvolvendo para desktop ou para Web** – Disponível em http://www.macoratti.net/vbn_dkwb.htm. Acesso em: 23 de maio 2011.

RIACENTER. 11 Razões para Utilizar o Flex. Disponível em: http://riacenter.com/blog/?p=71. Acesso em: 05 de maio de 2011.

SCHIMITZ, Daniel Pace - **Dominando Adobe Flex 4.** Livro – Bauru, SP:Canal6, 2009.

SCHIMITZ, Daniel Pace - **Dominando Flex e Zend. Livro** – Bauru, SP: Canal6, 2010.

SILVA, C.T. **Detalhando o projeto arquitetural no desenvolvimento de software orientado a agentes: O caso Tropo**, 182 p. Dissertação – Centro de Informática, Universidade Federal de Pernambuco. Recife: 2003.

SILVA, C.T. **Detalhando o projeto arquitetural no desenvolvimento de software orientado a agentes: O caso Tropo**, 182 p. Dissertação – Centro de Informática, Universidade Federal de Pernambuco. Recife: 2003.

SIERRA, Kathy; BASHAM, Bryan. **Use a Cabeça! Servlets & JSP**. Editora Alta Books, 2008.

WELTER, Rodrigo – **COMPARATIVO DE TECNOLOGIAS PARA APLICAÇÕES RICAS: JAVAFX E ADOBE FLEX.** Dissertação – UTFPR – Universidade Tecnológica Federal do Paraná, Campus Medianeira. Paraná: 2008.