UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

TIAGO DE ALCÂNTARA ESMERALDINO

COMPARAÇÃO ENTRE AS PLATAFORMAS ADOBE FLEX, SILVERLIGHT E JAVAFX PARA CRIAÇÃO DE APLICATIVOS RICOS NA WEB

CRICIÚMA, JULHO DE 2011

TIAGO DE ALCÂNTARA ESMERALDINO

COMPARAÇÃO ENTRE AS PLATAFORMAS ADOBE FLEX, SILVERLIGHT E JAVAFX PARA CRIAÇÃO DE APLICATIVOS RICOS NA WEB

Trabalho de Conclusão de Curso apresentado para obtenção do Grau de Bacharel em Ciência da Computação da Universidade do Extremo Sul Catarinense.

Orientador: Profº. Esp. Fabricio Giordani

CRICIÚMA, JULHO DE 2011

TIAGO DE ALCÂNTARA ESMERALDINO

Comparação Entre as Plataformas Adobe Flex, Silverlight e JavaFX para Criação de Aplicativos Ricos na Web

Submetido ao corpo docente do Curso de Ciência da Computação da Universidade do Extremo Sul Catarinense como um dos requisitos para obtenção do grau de Bacharel em Ciência da Computação.

Profa. MSc. Ana Claudia Garcia Barbosa
Coordenadora do Curso de Ciência da Computação

Banca Examinadora:

Prof. Esp. Fabrício Giordani (UNESC)
Orientador

Prof. Esp. Fábio Bif Goularte (UNESC)

Prof. MSc. Gustavo Bisognin (UNESC)

Aos amores da minha vida: meus pais, minha noiva, meus familiares, meus amigos e colegas.

AGRADECIMENTOS

Agradeço primeiramente a Deus por colocar pessoas maravilhosas na minha vida e ter me capacitado desde o inicio.

Agradeço aos meus pais Sergio e Kátia, que lutaram comigo durante todo o tempo. Agradeço novamente ao meu pai, que também é meu patrão, por ter me dado dois meses de férias, pois não teria tempo para concluir esse trabalho.

Agradeço a minha futura esposa, Cristina, que esteve literalmente do meu lado durante quase todo o curso me ajudando. Agradeço a minha cunhada Gesiane que me ajudou a revisar o trabalho e aos meus familiares, amigos e colegas que simplesmente enriquecem a minha vida.

Também agradeço aos meus orientadores, Cristiane Woszezenki e Fabricio Giordani, pela atenção, ajuda e direção que me deram neste trabalho.

"Estando em angústia, invoquei ao SENHOR, e a meu Deus clamei; do seu templo ouviu ele a minha voz, e o meu clamor chegou aos seus ouvidos."

2 Samuel 22

RESUMO

A web ganhou popularidade durante os anos e passou de plataforma de texto para plataforma de aplicativos. Porem o modelo atual com sites construídos em DHTML possui várias limitações, inviabilizando a construção de aplicativos que exigem alta interatividade. Muitas tecnologias foram criadas para suprir essas limitações, elas são conhecidas como plataformas RIA.

As plataformas RIA Adobe Flash, Silverlight e JavaFX são compostas por um *runtime* e uma biblioteca base, são capazes de executar dentro do navegador e elevar o grau de interatividade das aplicações. Apesar das plataformas possuírem similaridades, as plataformas possuem muitos aspectos diferentes que podem refletir na curva de aprendizado de um desenvolvedor ou equipe habituada que já está habituada em desenvolver aplicações desktop ou aplicativos web, no custo de desenvolvimento, na acessibilidade ou resultado final do aplicativo.

Este trabalho apresenta uma comparação entre as plataformas Adobe Flex 4.1, Silverlight 4 e JavaFX 1.3 na criação de aplicativos na internet. A comparação foi realizada por meio do estudo das características gerais de cada plataforma e pela criação de casos de uso, onde foi possível avaliar a acessibilidade, ferramentas disponíveis, linguagens suportadas, controles visuais e outros recursos. Também foi possível experimentar e avaliar as ferramentas, linguagens, a facilidade de consumir *web services*, o consumo de recursos entre outros aspectos.

Palavras-Chave: RIA; JavaFX; Silverlight; Adobe Flex.

ABSTRACT

Over the years the Web has gained popularity and became a applications platform. However the present model with content built in DHTML has several limitations, making it impossible to build applications that require high interactivity. Many technologies have been created to meet these limitations, they are known as RIA platforms.

The RIA platforms Adobe Flash, Silverlight and JavaFX are composed of a runtime and a base library, are capable of running inside the browser and raise the level of interactivity applications. Despite having similarities platforms, the platforms have many different aspects that may reflect on the learning curve of a developer or team that is already accustomed to developing desktop applications or web applications, the development cost, viability or outcome of the application.

This paper presents a comparison between the platforms Adobe Flex 4.1, Silverlight 4, and JavaFX 1.3 for creating web applications. The comparison was performed by the study of the general characteristics of each platform and by creating use cases. It was possible to assess the accessibility, development tools available, supported programming languages, visual controls. It was also possible to experience and evaluate the tools, programming languages, the ease of use web services, resource consumption and other aspects.

Keywords: RIA; JavaFX; Silverlight; Adobe Flex.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS

API	*Application Programming Interface*
ASP	*Active Server Pages*
CLR	*Common Language Runtime*
CSS	*Cascading Style Sheets*
DHTML	*Dynamic HyperText Markup Language*
DOM	*Document Object Model*
HLSL	*High-Level Shader Language*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
HTTPS	*HyperText Transfer Protocol secure*
JRE	*Java Runtime Environment*
JSON	*JavaScript Object Notation*
JSP	*Java Server Pages*
JVM	*Java Virtual Machine*
MVC	*Model-View-Control*
MVP	*Model-View-Presenter*
MVVM	*Model-View-ViewModel*
PDA	*Personal Digital Assistant*
PHP	*Hypertext Preprocessor*
PM	*Presentation Model*
REST	*Representational State Transfer*
RIA	*Rich Internet Applications*
SOAP	*Simple Object Access Protocol*

W3C	*World Wide Web Consortium*
WHATWG	*Web Hypertext Application Technology Working Group*
WSDL	*Web Services Description Language*
XAML	*Extensible Application Markup Language*
XHTML	*eXtensible Hypertext Markup Language*
XML	*eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

Inicialmente, a internet foi criada para ser uma plataforma de acesso a textos dinâmicos ou estáticos construídos em HyperText Markup Language (HTML). Nesse modelo original, os usuários são condicionados a interagir com páginas HTML por meio de links e preenchendo dados em formulários. Essa arquitetura simples tornou-se um padrão universal, onde as páginas são visualizadas em qualquer dispositivo que tenha um sistema operacional que possua um navegador básico.

A interface das aplicações desktop evoluiu conforme o tempo e tornou-se mais rica e interativa proporcionando uma melhor experiência para o usuário. No entanto, o desafio de uma aplicação desktop é a implantação, pois geralmente é preciso ter a base de código para cada plataforma. Além disso, cada atualização e manutenção deve ser realizada localmente na máquina (LAIR, 2009, tradução nossa).

Em contrapartida, as aplicações web rodam em servidores na internet e clientes com quaisquer plataformas podem acessar facilmente por meio de um navegador. Além disso, uma atualização na aplicação é realizada uma única vez no servidor e é aplicada para todos os usuários. Contudo, as aplicações web resultam, em sua grande maioria, em interfaces com o usuário pobres, quando comparadas a aplicações desktop.

Com as tecnologias mais recentes, as linhas entre as aplicações desktop e as aplicações web se cruzaram. Essa nova abordagem é denominada Rich Internet Applications (RIA) (LAIR, 2009, tradução nossa).

O termo RIA não se refere apenas a uma única tecnologia, mas sim a um conjunto heterogêneo de tecnologias que tem como intuito adicionar novos recursos e funcionalidades para a web tradicional (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010, tradução nossa).

De acordo com O'Rourke (2004, tradução nossa), tecnologias RIA nos permitem implantar clientes ricos por meio da Internet com a simplicidade de aplicações web tradicionais. Com isso, as aplicações podem ser mais sofisticadas, interativas, aumentando a usabilidade.

As plataformas de suporte a RIA que são conhecidas e se encontram na mesma categoria são JavaFX, Silverlight e Adobe Flex, das quais, esse trabalho propõe a compará-las. Apesar das plataformas possuírem similaridades, esse trabalho objetiva comparar aspectos como: portabilidade, sistemas operacionais suportados; facilidade de instalação das dependências (cliente); funcionalidades, tipos de mídia (vídeo, áudio, imagens), suporte a gráficos 2D e 3D, acesso a câmera, microfone entre outros recursos da máquina local; capacidade de executar fora do navegador; componentes inclusos na biblioteca padrão entre outros.

Também, esta pesquisa propõe a criação de protótipos com o intuito de avaliar o consumo de processamento de cada plataforma.

Com isso, o objetivo desta pesquisa é comparar as plataformas de forma a mensurar o quanto cada uma consegue se beneficiar da web e, ao mesmo tempo, usufruir das suas respectivas características de interação podendo, assim, guiar os desenvolvedores a escolha da melhor opção para implementação de sua aplicação.

1.1 OBJETIVO

Realizar uma comparação entre as plataformas Adobe Flex, Silverlight, JavaFX de aplicações ricas na internet (RIA).

1.2 OBJETIVOS ESPECÍFICOS

A fim de atingir o objetivo geral dessa pesquisa foram definidos os seguintes objetivos específicos:

a) documentar as funcionalidades de cada plataforma;

b) analisar e documentar a acessibilidade das plataformas;

c) documentar os fatores que podem influenciar na curva de aprendizado;

d) criar aplicação de testes em cada plataforma a fim de constatar o consumo de recursos;

e) analisar casos de uso em aplicações de negócios.

1.3 JUSTIFICATIVA

A web está cada vez mais sendo vista como uma plataforma. O termo "Web 2.0" é construído com foco no usuário, colocando-o no centro dos aplicativos, como pode ser constatado em redes sociais, *wikis*, vendas virtuais, buscadores entre outros. Nessa nova era de aplicativos, existe a necessidade de um alto grau de interação e usabilidade. A arquitetura HTML tradicional não suporta muitos aspectos exigidos nessa nova era de aplicativos em áreas como apresentação, comunicação e lógica de negócios. Muitas tecnologias estão moldando a web tradicional e acrescentando funcionalidades para corrigir os aspectos em que ela falha. Dentre estas tecnologias, as plataformas RIA se destacam desempenhando um papel fundamental (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010, tradução nossa).

O HTML5 é a quinta revisão majoritária da linguagem HTML, que suportará várias funcionalidades para trazer mais riqueza para a web sem necessidade de instalação de

clientes. Contudo, a especificação dessa nova tecnologia ainda se encontra em rascunho sujeito a alterações (HICKSON; HYATT, 2010, tradução nossa).

Por outro lado, as plataformas RIA têm evoluído trazendo soluções mais maduras para a criação de aplicativos ricos. Um exemplo disso é a plataforma Silverlight que, em torno de três anos, lançou quatro versões majoritárias. No entanto, cada plataforma tem suas peculiaridades como funcionalidades, maneiras de acessar dados de forma assíncrona em um servidor, linguagens suportadas, ferramentas para codificação e design, consumo de memória e processamento, navegadores e sistemas operacionais suportados, entre outros aspectos.

Muitos desses fatores podem influenciar na curva de aprendizado de um desenvolvedor ou de uma equipe que já está habituada a desenvolver aplicações desktop ou aplicativos web; no custo de desenvolvimento geral do projeto; na acessibilidade do aplicativo; no resultado final do sistema.

De acordo com Taivalsaari (2009, tradução nossa), as plataformas JavaFX, Silverlight e Adobe Flex se encontram na mesma categoria de tecnologia RIA, onde as mesmas possuem *plugins*/*runtimes* para executar aplicações dentro ou fora do navegador. Diferentemente de outras tecnologias como Ajax, Google Web ToolKit e Sun Labs Lively Kernel que utilizam o navegador para executar as aplicações.

Uma vez que estas plataformas encontram-se na mesma categoria tecnológica, se faz necessária uma comparação aprofundada de suas características de forma a avaliar seus prós e contras. Esta análise comparativa poderá nortear os desenvolvedores na escolha da plataforma mais adequada para suas necessidades.

É importante enfatizar que o conceito de computação nas nuvens vem sendo cada vez mais explorado e empresas portam ou criam versões de aplicativos existentes no desktop oferecendo-os como serviços na web. Um exemplo dessa tendência é a Microsoft, que disponibilizou o pacote Office online (Office Web Apps). Outro exemplo é a Google, que

lançará um sistema operacional para netbook baseado em um navegador web para executar com mais velocidade aplicativos para a web. Percebe-se que o próprio termo netbook enfatiza a importância dos aplicativos web e que eles devem suprir as experiências de usuário. Neste caso, a utilização de uma plataforma RIA se faz necessário.

1.4 ESTRUTURA DO TRABALHO

Este trabalho está dividido em cinco capítulos. No Capítulo 1 é apresentada uma introdução ao tema proposto, os objetivos gerais e específicos e a justificativa.

É apresentado no Capítulo 2 o que é a web, sua popularidade como uma plataforma e um modelo novo de negócios, arquitetura e evolução.

A abordagem do navegador como um cliente leve para execução de aplicações, as tecnologias atuais e futuras e também as limitações do mesmo são descritas no Capítulo 3.

No Capítulo 4 é conceituado aplicações ricas na internet; o cliente rico; as plataformas Adobe Flex e Flash, Silverlight e JavaFX; padrão de desenvolvimento; e a integração dos clientes com o servidor por meio dos serviços.

Estão descritos no Capítulo 5 alguns trabalhos correlatos a nível regional e mundial que contem um teor semelhante a este trabalho.

Por fim, o Capítulo 6 apresenta a comparação entre as plataformas RIA, a descrição das metodologias, a criação dos protótipos e recursos utilizados.

2 A WEB

A W*eb* consiste em milhares de clientes que interagem por meio de navegadores; servidores executando aplicações; conectados por meio de redes com fio e wireless (BASHAN; SIERRA; BATES, 2005). Ela foi criada inicialmente para ser uma plataforma de acesso a textos dinâmicos ou estáticos construídos em HyperText Markup Language (HTML). Os *web sites* baseados apenas nessa tecnologia permitem que o usuário leia textos, veja imagens, baixe arquivos e interaja preenchendo formulários e clicando em links.

Devido aos poucos requisitos para navegar em uma página HTML esse modelo tornou-se universal, pois é necessário apenas de um dispositivo que possua um navegador básico, já que não há necessidade de baixar nenhum *plugin* para visualizar as páginas construídas em HTML (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010, tradução nossa).

2.1 A BOLHA PONTO-COM

Em meados da década de 1990 houve um crescimento explosivo da internet e da web que levou a adoção de um novo modelo de conteúdo e aplicativos. (ALLAIRE, 2002, tradução nossa).

Em meio a esse crescimento algumas empresas de comercio eletrônico obtiveram sucesso na Web, como Amazon e Ebay, que atraíram investidores e em pouco tempos passaram a valer milhões. Esse sucesso repentino estimulou a criação de várias empresas *.com*, com capital aberto e com muitos investidores comprando suas ações. No entanto as empresas não geravam o lucro esperado ou até mesmo nada de lucro. Esse processo acabou

gerando uma bolha econômica, conhecida como bolha da internet ou bolha *ponto-com* (THE TRUE..., 2008).

De acordo com O'Reilly (2005, tradução nossa), apesar de muitas empresas quebrarem e os investidores perderem muito dinheiro, o estouro da bolha causou uma virada na web em 2001. Muitos concluíram que a web recebeu uma publicidade exagerada, mas na verdade bolhas e reorganizações normalmente fazem parte de qualquer revolução tecnológica.

2.2 A WEB 2.0

O termo *Web 2.0* foi criado durante uma conferencia de *brainstorming* entre a O'Reilly e a MediaLive International. O presidente e o vice-presidente da O'Reilly notaram que a web não havia explodido junto com a bolha econômica, mas estava com muito mais importância, com uma quantidade regular de aplicativos novos. Foi observado também que as empresas que sobreviveram tinham algo em comum e a partir dessas características é que foi definido o que seria a *Web 2.0* (O'REILLY, 2005, tradução nossa).

De acordo com O'Reilly (2005, tradução nossa), conhecido como o criador do termo mencionado, uma das mudanças que essa revolução trouxe foi a quebra de paradigma do modelo de negócios de software tradicional, isso é observado na Google e sua ferramenta de busca:

a) software como um serviço. Não é necessário licenças ou vendas, apenas o uso. Mesmo que o usuário pague direta ou indiretamente;
b) não há prazos para o lançamento do software, apenas contínuos aperfeiçoamentos;
c) sem conversões para diferentes plataformas, mesmo que internamente possa utilizar sistemas operacionais e softwares que são transparentes para o cliente.

Além do modelo de negócios os aplicativos tiveram em comum:

a) confiança no usuário, colocando o mesmo como colaborador. Alguns exemplos: *Wikipédia*, uma enciclopédia onde os usuários fornecem as informações; Ebay e Mercado Livre, site de vendas que colocam os usuários como vendedores; Flickr, usuários hospedam fotos; Youtube, para compartilhamento de vídeos; entre muitos outros aplicativos que de alguma maneira colocam o usuário como colaborador;
b) alcance para o maior número de pessoas ou sites também chamado de "Cauda Longa". Um exemplo é a Google AdSense que pode ser inserido em qualquer site, acabando com a idéia de que apenas grandes sites podem anunciar;
c) software para mais de um dispositivo;
d) experiência rica do usuário, como Gmail e Google Maps que fornecem um nível elevado de interatividade na web utilizando tecnologias como AJAX.

Sendo assim, Web 2.0 não é um termo para rotular apenas um tipo de aplicativo ou site, mas equivale a vários conceitos diferentes: sites construídos com um determinado tipo de tecnologia; redes sociais; sites que incentivam o usuário a criar conteúdo na forma de texto, vídeo ou foto, juntamente com os comentários, *tags* e avaliações; ou simplesmente os sites que ganharam popularidade nos últimos anos e estão sujeitos a especulações relacionado à bolsa de valores (CORMODE; KRISHNAMURTHY, 2008, tradução nossa).

2.3 ARQUITETURA TRADICIONAL DA WEB

Em uma arquitetura tradicional pode-se definir alguns pontos que ocorrem em uma interação do usuário com a web (BASHAN; SIERRA; BATES, 2005):

a) o usuário por meio do navegador faz requisições, seja pela barra de endereço ou clicando em links nas páginas;
b) o servidor recebe a requisição, encontra o recurso que muitas vezes pode ser uma página html, uma imagem, um arquivo de som, um pdf ou qualquer outro arquivo binário. e envia de volta além do recurso, informações para o navegador lidar com ele;
c) caso o servidor não encontre o recurso, será enviado um código de erro;
d) o navegador processa a resposta e a exibe ao usuário. caso a resposta seja um conteúdo html, o navegador irá formatar a página e exibir ao usuário; no caso de um pdf, poderá abri-lo; ou outro tipo de arquivo opção para download.

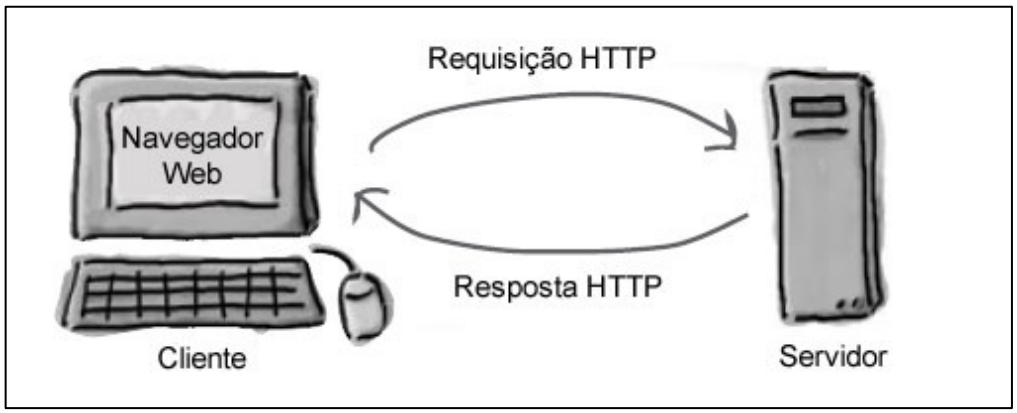

Figura 1. Requisição e Reposta HTTP.
Fonte: Adaptado de BASHAN, B; SIERRA, K; BATES, B (2005)

Toda comunicação é feita por meio do protocolo HyperText Transfer Protocol (HTTP), sendo que o conteúdo das requisições e respostas são encapsuladas dentro dessas mensagens. Ele depende de outros protocolos que são responsáveis por enviar com integridade as mensagens para o destino, mesmo que elas estejam divididas em blocos (BASHAN; SIERRA; BATES, 2005).

2.4 O LADO DO SERVIDOR

O servidor é quem recebe as requisições HTTP e as processa devolvendo o recurso encontrado ou o código de erro. Quando solicitado uma página web o servidor apenas a encontra em uma pasta e a devolve, sendo que a mesma está estática. Esse é o papel dos servidores como Apache, Tomcat, IIS entre outros.

De acordo com o Cox (2008, tradução nossa), no inicio a web era basicamente uma plataforma de compartilhamento de arquivos e leitura de textos. Entretanto surgiu a necessidade de criar páginas dinâmicas para processar, por exemplo, as requisições dos formulários. Para auxiliar nesse processo é que surgiram tecnologias capazes de trabalhar em conjunto com os servidores para gerar os recursos solicitados dinamicamente, assim dando vida a páginas que antes eram meros textos estáticos, viabilizando os aplicativos web.

Para facilitar o desenvolvimento de páginas dinâmicas nasceram aplicativos e linguagens voltadas à web que auxiliam a criação de aplicações que recebem as requisições como ASP.NET, JSP, PHP entre outras. Sendo que as páginas não existem até o momento que o cliente faz a solicitação, o servidor recebe e os aplicativos processam a requisição e devolvem uma resposta para o cliente dinamicamente, atuando em conjunto com o servidor. Essas abstraem o protocolo HTTP tornando transparente esse processo, como uma camada adicional auxiliando o desenvolvedor (BASHAN; SIERRA; BATES, 2005).

2.5 A EVOLUÇÃO DA WEB

A web tem sofrido uma serie de fases evolutivas. De acordo com Taivalsaari (2009, tradução nossa) a evolução está divida em três fases como é ilustrado na Figura 2:

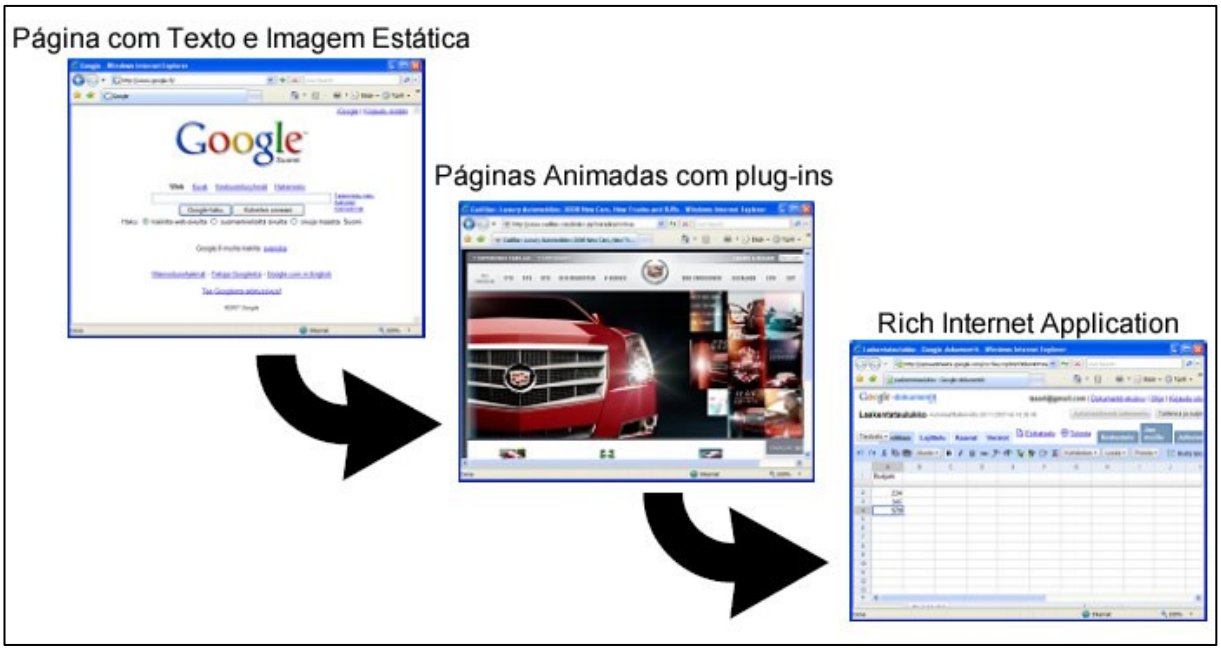

Figura 2. A evolução do uso da web
Fonte: Adaptado de TAIVALSAARI, A (2009)

a) na primeira fase, as páginas web eram simples documentos de textos com algumas imagens estáticas intercaladas, sem animação ou qualquer outro conteúdo interativo. A navegação entre páginas foi baseada simplesmente em hiperlinks e uma nova página era carregada a partir do servidor da web a cada vez que o usuário clicou em um link. Não houve necessidade de uma comunicação assíncrona de rede ou protocolos de comunicação mais avançadas entre o navegador e servidor web. Algumas páginas apresentavam formulários com botões e outros componentes simples para realizar a entrada de dados. Essas páginas ainda são muito comuns;

b) gradualmente com a introdução do DHTML[1] se tornou possível a criação de páginas web cada vez mais interativas com gráficos animados e *plugins* que permitiram um conteúdo mais dinâmico e rico para ser apresentado. Esta fase coincidiu com a decolagem comercial da web, quando as empresas perceberam o valor comercial de um site, podendo colocar anúncios, oferecer serviços e

[1] DHTML é a combinação de HTML, Cascanding Style Sheets (CSS), JavaScript e Document Object Model (DOM).

vender mercadorias. A navegação já não é mais limitada a hiperlinks e a comunicação entre o navegador e o servidor se tornou mais complexa;

c) a terceira fase é caracterizada pelas aplicações ricas. As tecnologias para o desenvolvimento de aplicações ricas são também conhecidas como "Tecnologias Web 2.0". Essas tecnologias tendem a combinar a colaboração e a interatividade que são características da Web 2.0. Elas permitem construir web sites que se comportam como aplicações desktop, por exemplo, permitindo atualizar apenas parte de uma página ao invés de atualizar ela completamente a cada vez que algo muda. Também é evitada a utilização de links fazendo com que o usuário interaja com elementos que remetem a uma aplicação desktop.

3 NAVEGADOR: O CLIENTE LEVE

Tradicionalmente as aplicações desktop são distribuídas em um pacote de instalação, mesmo em uma arquitetura cliente/servidor, o aplicativo vem acompanhado de uma interface gráfica para o usuário interagir, que seria um *cliente pesado*. Esse cliente possibilita uma interface gráfica com imagens, som, e controles interativos e o mesmo possui afinidade com o sistema operacional.

Com a web ganhando popularidade como uma plataforma, o navegador acabou se tornando um cliente leve para a interface gráfica, enquanto o lado do servidor é composto por máquinas poderosas e escaláveis capazes de criar várias páginas HTML dinamicamente de acordo com a necessidade do aplicativo (DRIVER; VALDES; PHIFER, 2005, tradução nossa).

3.1 LINGUAGENS E TÉCNOLOGIAS DO NAVEGADOR

Ao longo dos anos, inúmeras tecnologias e metodologias de programação foram criadas com o único propósito de melhorar a experiência do usuário final de um website. O navegador não ficou limitado em processar apenas o HTML puro, surgiu e aprimoraram-se as linguagens que são executadas com ele.

3.1.1 O Universal HTML

O HTML é uma linguagem de marcação criada por Tim Berners-Lee e que ganhou popularidade pelo navegador Mosaic, tornando-se a linguagem universal da web. O HTML utiliza *tags* (por exemplo, *<h1> </h1>*) para estruturar o texto em cabeçalhos,

parágrafos, listas, links de hipertexto etc. (RAGGETT; HORS; JACOBS, 2010, tradução nossa).

Por ser uma linguagem de marcação não é possível desenvolver lógica de negócios em HTML, mas o mesmo fornece uma *tag* para anexar linguagem de scripts. Também existe uma *tag* para anexar declarações de estilos que possibilita a separação das formatações das páginas do código HTML.

3.1.2 Linguagens de Scripts

O navegador não é apenas responsável pela exibição de páginas construídas em HTML, mas executa também códigos chamados de scripts como JavaScript e VBScript, sendo que JavaScript é a principal, e se tornou uma das linguagens mais populares do mundo.

JavaScript é uma linguagem de programação leve e interpretada que implementa características da orientação a objetos, e possui uma sintaxe que lembra C e Java. A linguagem teve alta adoção e o seu núcleo foi incorporado em vários navegadores. O JavaScript que foi incorporado nos navegadores permite a inserção de código fonte em páginas web - isso significa que uma página da web não precisam mais ser HTML estático, mas podem incluir códigos que controlam o navegador a até criar dinamicamente o conteúdo HTML afim de interagir com o usuário (FLANAGAN, 2001, tradução nossa).

3.1.3 Cascading Style Sheets

Cascading Style Sheets (CSS) é um mecanismo simples para adicionar estilo (fontes, cores, espaçamento) a documentos web. O CSS proporciona o desacoplamento da formatação das páginas HTML, isso permite que várias páginas utilizem o mesmo arquivo

CSS facilitando a construção e manutenção. Também pode ser inserido o código CSS dentro das páginas (BOS et al, 2010**,** tradução nossa).

3.2 AS LIMITAÇÕES

Essa abordagem do navegador como um cliente leve se saiu bem ao longo dos anos, no entanto esse modelo também sofreu com desvantagens significativas e limitações, especialmente em torno da riqueza das interfaces de aplicativos, mídia e conteúdo, e da sofisticação geral das soluções que poderiam ser construídos e entregues ao usuário (ALLAIRE, 2002, tradução nossa). Em implementações típicas, os elementos tradicionais do cliente de aplicativos baseados em navegador são limitadas à lógica de interface com pequenas quantidades de código de script (como JavaScript) para pequenas validações de dados e lógica de controle.

Outro problema deve-se ao fato de que programar para navegadores pode não ser uma tarefa fácil:

a) o JavaScript apesar de ser uma das linguagens mais conhecida do mundo também é uma das mais desprezada devido a complexidade em se trabalhar com o Document Object Model (DOM) que é mal especificado e implantando de forma inconsistente (CROCKFORD, 2008, tradução nossa);
b) apesar do JavaScript ser utilizado por pessoas que não sabem programar, ele ainda pode ser frustrante para fazer tarefas não triviais (FLANAGAN, 2001, tradução nossa);
c) o que pode ser um dos piores problemas é a falta de padronização de algumas API's dos navegadores para o JavaScript. É possível ficar horas construindo uma página e funcionar em um navegador e mais tarde vê-la falhar em outro. E

muitas vezes isso só pode ser contornado com trabalho duro, “hacks” e ajustes bizarros. Ainda assim a aplicação web não consegue ser tão interativa quanto uma aplicação desktop (LECRENSKI, 2010, tradução nossa);

d) existe uma diversidade de navegadores. Cada navegador pode exibir da maneira que bem entender uma determinada formatação. Essa falta de padronização visual resulta as vezes em pequenas diferenças em relação ao design da tela (ALLAIRE, 2010, tradução nossa);
e) o armazenamento local é limitado a *cookies*.

3.3 O FUTURO HTML5

Em seus primeiros cinco anos (1990-1995) o HTML passou por muitas revisões e experimentou uma série de extensões, inicialmente hospedado no CERN e depois no IETF. Logo após a criação do W3C o HTML evoluiu com sua versão *3.2* que foi concluída em 1997 e com a quarta versão que foi concluída em 1998. Depois da quarta versão a W3C decidiu não aprimorar mais o HTML e trabalhou em uma reformulação da linguagem em XML, conhecido como XHTML concluída em 2000. Ainda as *APIs* dos navegadores (DOM) tiveram três versões até 2003.

A idéia de que a evolução do HTML deve ser reaberta foi testado em um workshop do W3C, em 2004, onde alguns dos princípios que fundamentam o trabalho HTML5 foram apresentados pela Mozilla e Opera. Porém os membros da W3C não aprovaram, pois entrava em conflito com suas decisões anteriores que era uma linguagem sem compatibilidade com o HTML/XHTML chamada de XHTML2.

Pouco tempo depois a Apple, Mozilla e Opera anunciaram sua intenção de continuar o desenvolvimento do HTML5 formando um novo grupo chamado Web Hypertext

Application Technology Working Group (WHATWG). Em 2007 a W3C se juntou ao grupo e publicou uma especificação sob os direitos autorais da W3C (HICKSON; HYATT, 2010, nossa tradução).

3.3.1 Funcionalidades do HTML5

Mesmo ainda com a especificação em *draft*, sujeito a alterações, vários navegadores já dão suporte parcial ou completo ao HTML5. Essa nova tecnologia não acrescenta apenas *tags* novas para o HTML, mas há uma melhoria no CSS que chega sua terceira versão e também em novas APIs para o JavaScript (GOOGLE, 2010, tradução nossa):

a) novas APIs para JavaScript: Armazenamento de dados no cliente com Web Sql Database, App Cache, Web Storage;
b) novas *tags* HTML para melhorar a semântica, acessibilidade, novos controles para formulários, capacidade multimídia (som e vídeo), desenhos 2D e 3D com a *tag canvas*;
c) o CSS3 traz também melhoria na escolha da tipografia, novos efeitos visuais, transições, transformações e animações aos elementos do HTML.

É evidente que as novas funcionalidades trazem mais riqueza a web suprindo muitas limitações existentes no HTML4.

4 RICH INTERNET APPLICATIONS

As interfaces das aplicações desktop evoluíram conforme o tempo e tornaram-se mais ricas e interativas proporcionando uma melhor experiência para o usuário. No entanto, um desafio de uma aplicação desktop é a implantação, pois geralmente é preciso ter a base de código para cada plataforma. Além disso, cada atualização e manutenção devem ser realizadas localmente na máquina (LAIR, 2009, tradução nossa).

Essas desvantagens dos aplicativos desktop são facilmente resolvidas em aplicações web, pois a implantação é simples necessitando apenas de um dispositivo que tenha um navegador, além disso, atualizações e manutenções são realizadas apenas no servidor diminuindo a complexidade.

O termo Rich Internet Application (RIA) foi criado pela Macromedia quando ela deu inicio a uma nova abordagem para o Flash. O mesmo poderia fazer muito mais que apenas animações, fornecendo um cliente rico para execução de aplicações na internet. Nessa nova abordagem uniu-se o melhor dos dois mundos (O'REILLY, 2005, tradução nossa).

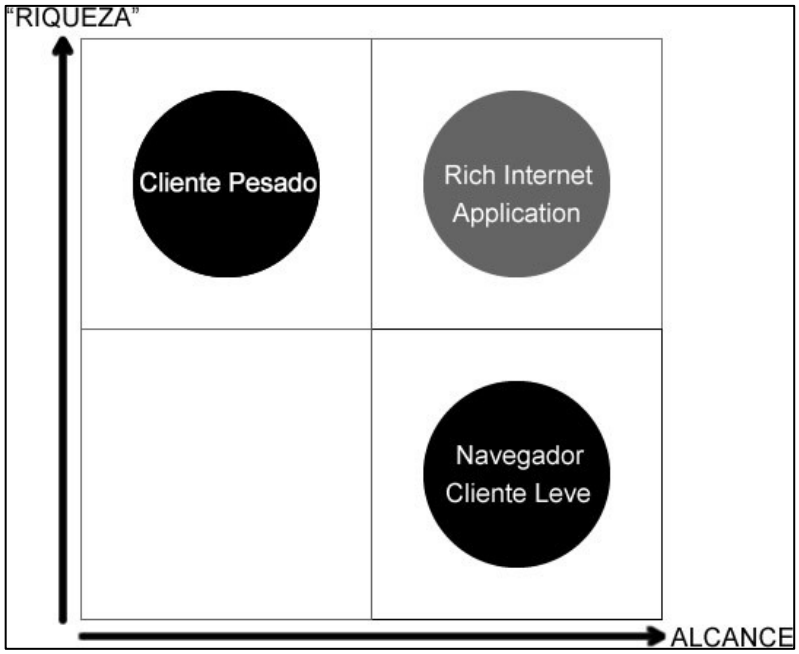

Figura 3. A abordagem das aplicações ricas
Fonte: Adaptado de DRIVER, M; VALDES, R; PHIFER, G (2005)

Existem algumas categorias de tecnologias RIA como demonstra a Figura 4. Sendo que Adobe Flash/Flex, Silverlight e JavaFX se encontram na mesma categoria, todas utilizam um cliente rico ou seja, são baseadas em *plugins* e com possibilidade de execução no desktop. Enquanto que as tecnologias baseadas no navegador acabam tendo um alcance maior, mas perdendo na riqueza da aplicação (TAIVALSAARI, 2009, tradução nossa).

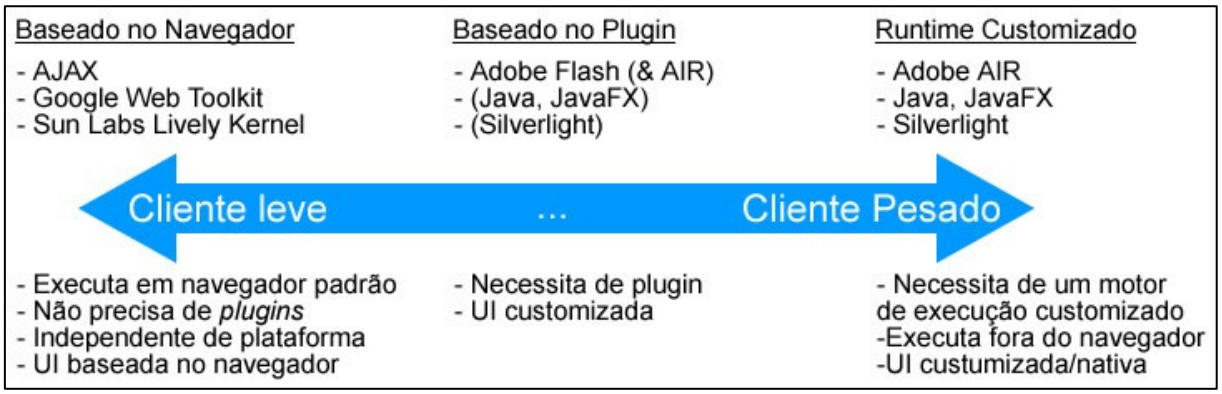

Figura 4. As categorias das tecnologias que proporcionam riqueza na Web
Fonte: Adaptado de TAIVALSAARI, A (2009)

4.1 MELHORANDO A WEB COM O CLIENTE RICO

Um cliente rico é uma plataforma para execução de aplicativos inseridos nos navegadores por meio de um *plugin*. É composto normalmente por um *runtime*[2] e uma biblioteca base capaz de ser executado em vários navegadores e sistemas operacionais.

A Macromedia estabeleceu que um cliente rico devesse (ALLAIRE, 2002, tradução nossa):

a) proporcionar eficiência e alto desempenho para execução do código, conteúdo e comunicação. O principio é a experiência do usuário final, os aplicativos web tradicionais sofrem com vários desafios relacionados ao desempenho. Inclui o modelo de *requisição-resposta* para a visualização da página; a necessidade de criar textos para transmissão de dados simples; falta de armazenamento no lado do cliente; a incapacidade de invocar com facilidade e usar lógica de negócios remota; e até os modelos gráficos do HTML. Isso tudo deve ser melhorado;
b) fornecer modelos de objetos poderosos e extensíveis para interatividade. Enquanto os navegadores têm progredido em termos de suporte a interatividade por meio do DOM, JavaScript e DHTML, eles ainda não tem a riqueza necessária para a construção de determinados aplicativos. Clientes ricos precisam fornecer um poderoso modelo baseado em objeto e eventos para aplicações. Esse modelo comum deve ser integrado com a interface do usuário, comunicação e serviços do sistema;
c) permitir o desenvolvimento rápido de aplicações por meio de componentes e reutilização. Os clientes ricos devem suportar a construção de componentes poderosos. Fornecendo facilidade a terceiros e desenvolvedores corporativos

[2] Runtime é um agente que gerencia o código em tempo de execução, fornecendo serviços principais como gerenciamento de memória, de thread e comunicação enquanto promove restrição de tipos e outras formas de garantir a segurança e a robustez (MICROSOFT COPORTATION, 2010a).

na reutilização de componentes visuais para acelerar o desenvolvimento. Estes componentes devem se integrar perfeitamente no ambiente de design para a facilidade de desenvolvimento;

d) proporcionar o uso da web e serviços de dados fornecidos pelos servidores de aplicação. A promessa de clientes ricos inclui a capacidade de separar a lógica de apresentação de forma limpa e interfaces de usuário a partir da lógica do aplicativo hospedado na rede. Os clientes ricos devem fornecer um modelo que facilite a utilização de serviços providos por componentes de *back-end*, hospedado em um servidor de aplicativos ou acessado por *web services*;

e) adotar clientes conectados e desconectados. Enquanto muitos usuários costumam estar online e utilizar o navegador para trabalhar, a realidade é que a maioria das aplicações se beneficiam da capacidade de ser usado offline em dispositivos conectados ocasionalmente. Os clientes ricos devem permitir ambos os tipos de aplicativos que podem ser facilmente construídos e implantados;

f) possibilitar fácil publicação em múltiplas plataformas e dispositivos. Uma premissa da web é possibilitar acesso ao conteúdo e aplicações de qualquer lugar, independentemente da plataforma e dispositivo. Os clientes ricos devem abraçar e apoiar os sistemas operacionais populares, bem como a mais ampla gama de plataformas de dispositivos emergentes, tais como *smart phones, PDAs,* consoles de jogos entre outros.

4.1.1 Adobe Flex

O Adobe Flex é um framework integrante da plataforma Flash que facilita o desenvolvimento de aplicações ricas na internet, contendo diversos tipos de componentes visuais e uma poderosa linguagem de programação chamada ActionScript. Ele utiliza o *runtime* do Flash Player, sendo assim ele é baseado em suas funcionalidades que cuidam da lógica do lado do cliente e que também permite interagir com o conteúdo HTML e JavaScript do navegador. O Flash Player possui uma alta taxa de adoção e é de natureza multi-plataforma que permite poupar muito tempo que seria gasto em testes e depuração em diferentes versões do navegador e sistemas operacionais diferentes (SCHMITZ, 2008).

Os aplicativos Flex são compostos de arquivos MXML e ActionScript. O MXML é uma linguagem de marcação que se baseia no padrão Extensible Markup Language (XML), ela é utilizada para definir a interface do usuário por meio de *tags* que correspondem a elementos visuais e aos aspectos não visuais da aplicação (como *data-binding* e recursos do lado servidor). Cada *tag* é baseada em classes criadas em ActionScript e tudo é compilado em um arquivo SWF. O ActionScript é uma linguagem de programação baseada no padrão ECMA Script 262, ela permite que o desenvolvedor programe o comportamento da aplicação e como o usuário pode interagir com os elementos da interface (CASARIO, 2007, tradução nossa). A Figura 5 apresenta o processo de compilação do Adobe Flex.

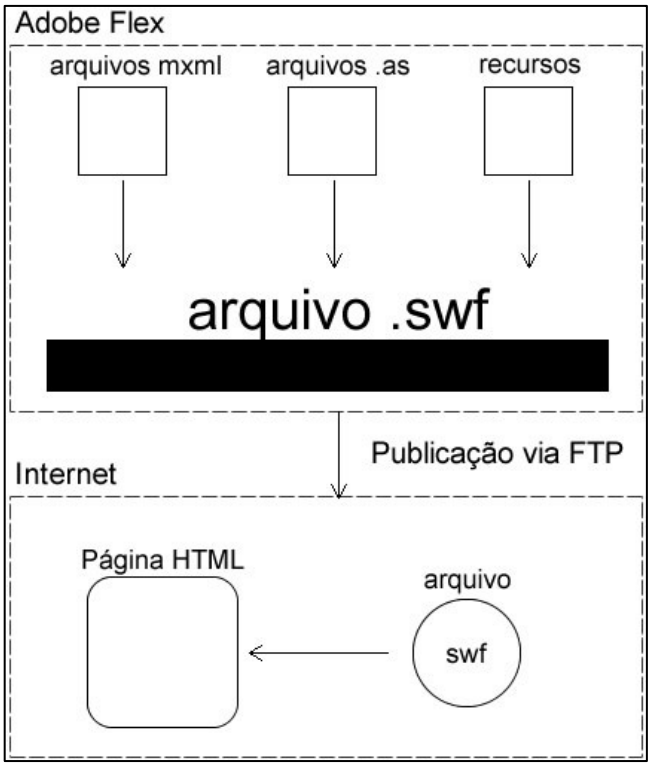

Figura 5. Compilação do Adobe Flex
Fonte: Adaptada de SCHMITZ, D (2008)

4.1.1.1 A Plataforma Adobe Flash

A plataforma Adobe Flash é líder em web design e plataforma de desenvolvimento para criar aplicações expressivas, conteúdo e vídeo que executam de forma consistente em muitos sistemas operacionais e dispositivos, com uma base instalada de 98% de usuários desktop conectados na internet. A Figura 6 apresenta uma visão geral da plataforma Flash, que é composta de um conjunto integrado de tecnologias, incluindo *runtimes*, ferramentas, *frameworks*, serviços e servidores, que são capazes de entregar aplicativos, conteúdo e vídeo de alta definição para o maior público possível (ADOBE SYSTEMS INCORPORATED, 2010b, tradução nossa).

Figura 6. Visão geral da plataforma Adobe Flash
Fonte: ADOBE SYSTEMS INCORPORATED (2010b)

4.1.2 Silverlight

O Microsoft Silverlight (ou apenas Silverlight) é uma implementação do *.Net Framework* para criação e entrega de aplicativos ricos na internet em múltiplos navegadores e sistemas operacionais. O Silverlight utiliza uma versão compacta do *.NET CLR* (*runtime* da Plataforma .Net) que permite escrever o código em linguagens suportadas pela plataforma .NET como C# ou Visual Basic, além de linguagens dinâmicas (MICROSOFT CORPORATION, 2010d, tradução nossa). A Figura 7 apresneta a arquitetura do Silverlight.

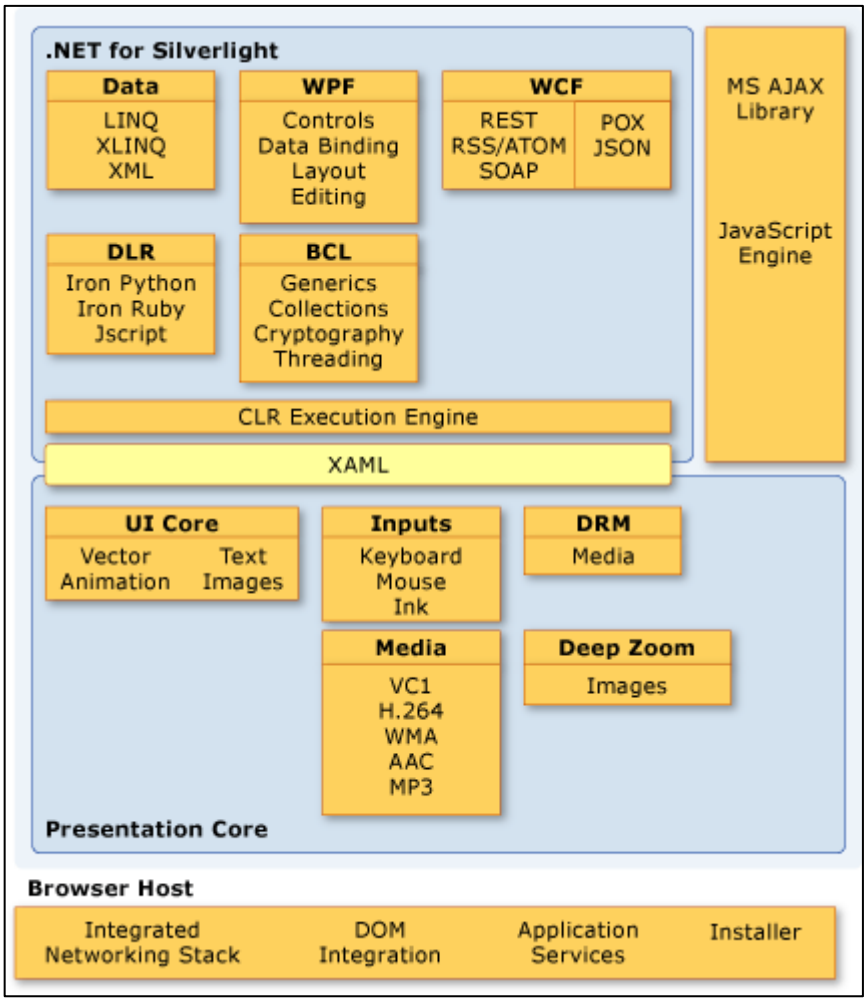

Figura 7. Arquitetura do Silverlight
Fonte: MICROSOFT CORPORATION (2010b, tradução nossa)

A Figura 8 apresenta um exemplo de código em Extensible Application Markup Language (XAML), que é a linguagem utilizada para definir a interface do usuário e tem sua base em XML. Com XAML é possível definir elementos visuais como controles ou formas além de animações, transformações e eventos. O código em linguagem de programação gerenciado ou dinâmico define o comportamento do aplicativo (MICROSOFT CORPORATION, 2010c, tradução nossa).

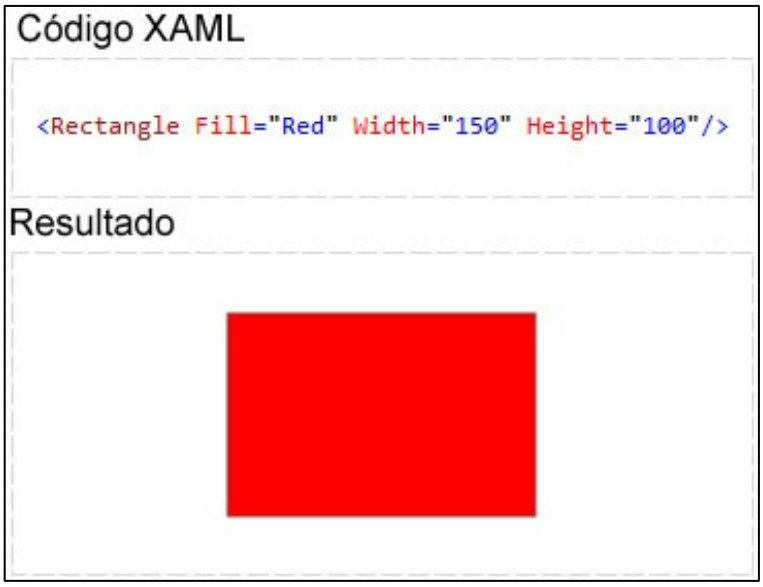

Figura 8. Exemplo de código XAML
Fonte: Adaptado de MICROSOFT CORPORATION (2010c)

4.1.3 JavaFX

O JavaFX é uma plataforma para criação e distribuição de aplicativos ricos na internet, no desktop, televisões e outros dispositivos. JavaFX utiliza o *Java Runtime Environment* (JRE), que é o mesmo *runtime* da plataforma Java. Sendo assim a aplicação é executada dentro do navegador pela *Java Virtual Machine* (JVM) (ORACLE CORPORATION, 2010b, tradução nossa). A Figura 9 apresenta a arquitetura da plataforma JavaFX.

Figura 9. Arquitetura da Plataforma JavaFX
Fonte: ORACLE CORPORATION (2010b)

Para o desenvolvimento de aplicações, o JavaFX utiliza uma nova linguagem de programação chamada *JavaFX Script*. Essa linguagem foi construída do zero para modelagem e animação de aplicações multimídia. Ela é uma linguagem compilada e orientada a objetos, com uma sintaxe independente do Java tradicional, porém é capaz de trabalhar com classes Java (MORRIS, 2009, tradução nossa).

4.2 PADRÃO DE ARQUITETURA

No contexto de software, geralmente um sistema com arquitetura modular é melhor do que o monolítico. No entanto algumas arquiteturas modulares são melhores que outras, as características comuns entre as boas arquiteturas são o baixo acoplamento e a alta coesão de seus módulos (HALL, 2010, tradução nossa).

Quando não se trabalha com um padrão modular normalmente a lógica de negócios e até o acesso a dados está fortemente acoplado na interface do usuário. Por não separar o sistema em partes o desenvolvimento acaba se tornando mais rápido e fácil. Porém por conseqüência do alto acoplamento, trechos de código são repetidos em várias partes do

sistema aumentando a complexidade de compreensão e manutenção (SANDERSON, 2010, tradução nossa).

É amplamente aceita a separação das funções de um software (normalmente usando representações orientadas a objetos) devido aos graves impactos negativos que o alto acoplamento causa em um projeto. A separação da lógica de negócios, também conhecido de modelo (ou *model*), da interface gráfica consiste na identificação e associação com entidades do mundo real, operações e regras existentes em um determinado assunto da qual é o alvo do software. Essa divisão e delegação de funções aos seus respectivos modelos aumentam o reuso do código sem repetições aumentando a compreensão da lógica e velocidade de realizar manutenção, também permite uma linguagem unificada entre especialistas de negócios e dos programadores.

4.2.1 Presentation Model ou Model-View-ViewModel

Alguns padrões se tornaram populares em realizar a separação do modelo da interface gráfica como o Model-View-Controller (MVC) e outras variações como Model-View-Presenter (MVP) e o mais recente Model-View-ViewModel (MVVM). Algo em comum nos padrões Model-View-"x" é que a separação do modelo e da visão (interface gráfica ou representação visual dos dados) é sempre feita por um mediador (HALL, 2010, tradução nossa).

O padrão PresentationModel (PM) foi criado com a premissa de que o mediador seria uma abstração da visão, definindo seu comportamento e estado, esse objeto que abstrai a visão é chamado de PresentationModel, sendo que ele e a visão estão em constante sincronia. Em 2005 John Gossman que é um arquiteto da Microsoft que trabalha nas plataformas Silverlight e WPF, adotou o padrão PM nas plataformas citadas e padronizou sua

implementação, dando o nome de MVVM. No MVVM o objeto PresentationModel (o mediador) é chamado de ViewModel. O padrão MVVM é recomendado pela Microsoft na construção de aplicativos utilizando Silverlight e o PM é recomendado pela Adobe na construção de aplicativos na plataforma Flex (WILLIANS 2007; SMITH 2009, tradução nossa).

4.3 INTEGRAÇÃO RIA COM O SERVIDOR

Os aplicativos ricos atuam na máquina do cliente, no entanto a base de dados continua no lado do servidor. De acordo com a pesquisa realizada, os clientes ricos possuem limitações e não podem acessar diretamente uma base de dados no servidor.

Para realizar a integração entre as tecnologias que atuam no servidor se faz necessário o uso de *web services* (ou serviços web). Os serviços podem se conectar diretamente com a base de dados ou com a camada de acesso a dados e enviar o resultado para o cliente. Os dados não precisam necessariamente vir de uma base de dados, podem vir de um arquivo texto ou de um resultado de um calculo do servidor (CLEEREN, 2010, tradução nossa).

A maioria das pessoas está familiarizada com acesso à web por meio de um navegador, que fornece uma interface visual para humanos. Ele apresenta textos formatados, imagens e todos os outros tipos de conteúdo, dessa maneira o protocolo HTTP atua de forma transparente para o usuário.

Os serviços web são componentes de *software* distribuídos para fornecer informações para aplicações em vez de seres humanos, por meio de uma interface orientada a aplicação. Geralmente a informação é estruturada utilizando XML, para que ele possa ser

analisado e processado facilmente ao invés de ser formatado e exibido (SRINIVASAN; TREADWELL, 2005, tradução nossa).

Os serviços publicam detalhes de suas funções/métodos e interfaces, mas eles mantêm os seus detalhes de implementação privado, portanto o cliente e o serviço suportam os mesmos protocolos de comunicação e interagem independente da plataforma ou linguagem de programação.

4.3.1 Especificações mais Utilizadas em *Web Services*

Os *web services* podem suportar qualquer protocolo de conexão, porém o mais comum é o SOAP sobre o HTTP ou HTTPS, por serem protocolos simples, são suportados na maioria dos dispositivos e normalmente não possuem bloqueio de firewall (SRINIVASAN; TREADWELL, 2005, tradução nossa). E também a estrutura das informações podem obedecer outros formatos além do XML como JavaScript Object Notation (JSON) que é um formato mais sucinto (MORRIS, 2009, tradução nossa).

Sendo assim, as especificações mais utilizadas para expor os serviços são: o XML, para formatação e intercâmbio de dados estruturados; SOAP (Simple Object Access Protocol), que é um protocolo baseado XML para especificação das informações do envelope, conteúdo e informações de processamento de uma mensagem; WSDL (Web Services Description Language) é uma linguagem baseada em XML utilizada para descrever os atributos, interfaces e outras propriedades de um web service. (W3C, 2000, tradução nossa; W3C, 2001, tradução nossa).

Além do SOAP, nos últimos anos os web services RESTful tem ganhado popularidade. Esses serviços funcionam por meio de simples requisições HTTP para interagir com os recursos do servidor (WEAVER et al, 2009). Nesse contexto os métodos do HTTP são

utilizados para sinalizar o servidor qual operação será realizada com o determinado recurso. Os métodos de acesso são: *PUT*, que indica criação; *GET* que indica leitura; *POST* que indica atualização; *DELETE* que indica exclusão. Os dados resultantes das operações são geralmente enviados em XML ou JSON.

5 TRABALHOS CORRELATOS

Essa seção relaciona alguns dos trabalhos com conteúdo semelhante a esta fundamentação teórica utilizados no desenvolvimento deste trabalho de pesquisa.

5.1 ESTUDO DE VIABILIDADE DO HTML5 PARA DESENVOLVIMENTO WEB

Trabalho de Conclusão de Curso de Daniel Fuverki Hey, apresentado ao Curso de Ciência da Computação do Centro de Tecnologia, em 2010, da Universidade Estadual de Maringá.

O trabalho apresenta um estudo que aborda os atributos gerais da linguagem HTML5, descrevendo o funcionamento dos novos elementos estruturais de marcação, conteúdo multimídia e aplicações disponíveis nos navegadores. O trabalho também apresenta uma comparação das tecnologias atuais, instigando desenvolvedores a considerar HTML5 em futuros projetos.

5.2 UMA COMPARAÇÃO ENTRE DUAS TECNOLOGIAS QUE PODEM SER UTILIZADAS PARA O DESENVOLVIMENTO RIA

Dissertação de Geraint Huw Davies, apresentada em cumprimento parcial das exigências para o grau de Mestre em Ciências da Internet & Sistemas Distribuídos, em 2009, da Universidade de Aberystwyth do Reino Unido.

O trabalho apresenta um estudo que identifica e compara as capacidades das tecnologias Adobe Flex 3 e Microsoft Silverlight 2 no suporte para o desenvolvimento de aplicações ricas na internet.

5.3 COMPARAÇÃO TÉCNICA E ESTUDOS DE CASO DE AJAX, FLASH E JAVA BASEADO EM RIA

Relatório de Tom Nada e Shawn Helwig, em 2005, da Universidade de Wisconsin–Madison, do Estados Unidos.

O trabalho apresenta uma comparação entre as tecnologias Java, Flash e Ajax na criação de aplicativos ricos na internet. Também explica o que é RIA e suas diferenças entre um aplicativo comum e apresenta estudos de caso.

6 COMPARAÇÃO ENTRE AS PLATAFORMAS

As plataformas Adobe Flex, Silverlight e JavaFX possuem um objetivo comum de trazer riqueza aos aplicativos web. Cada uma delas possui suas particularidades e tiveram sua origem por meio de plataformas já conhecidas: Adobe Flex faz parte da plataforma Adobe Flash que é amplamente utilizada em web sites, e é a pioneira em RIA; Silverlight se originou da plataforma *Microsoft .NET*; e JavaFX é parte da plataforma Java, sendo que as plataformas Java e *Microsoft .NET* são utilizadas para desenvolvimento de aplicações desktop, web e outros dispositivos.

6.1 METODOLOGIA

Para a realização da comparação foi estudado as características gerais de cada plataforma. Além do estudo foram criados cenários a fim de testar: o uso das ferramentas de codificação e design; linguagens de programação; facilidade de manipular dados na web; utilização de um padrão de arquitetura; a capacidade de alterar o visual dos elementos gráficos; animações e efeitos visuais.

Foram escolhidas para realizar a avaliação as seguintes versões estáveis de cada plataforma: Adobe Flex 4.1; Silverlight 4 e JavaFX 1.3.

6.2 ACESSIBILIDADE DAS PLATAFORMAS

A web se tornou extremamente acessível por meio dos navegadores, bastando o mesmo exibir o resultado do HTML. Mas existem algumas barreiras para as plataforma RIA

como os navegadores suportados, sistemas operacionais, o tamanho e a dificuldade de instalação de seus *plugins*.

6.2.1 Navegadores e Sistemas Operacionais Suportados

O navegador é o principal meio de acesso à web. O suporte aos diferentes navegadores existentes interfere diretamente na acessibilidade de um aplicativo e pode afetar o conceito da "cauda longa" da Web 2.0. A Tabela 1 apresenta o suporte oficial de cada plataforma.

Tabela 1. Navegadores Suportados

	Adobe Flex	Silverlight	JavaFX
Internet Explorer	Sim	Sim	Sim
Firefox	Sim	Sim	Sim
Chrome	Sim	Sim	Parcial
Safari	Sim	Sim	Sim
Opera	Sim	Não	Parcial
AOL	Sim	Não	Parcial

O JavaFX suporta todos navegadores, porém a API JavaFX Media responsável pela execução de vídeo e áudio possuem implementações diferentes para cada sistema operacional, devido a essa característica é oferecido suporte oficial nos navegadores Internet Explorer, Firefox e Safari (KONCHADY; REDKO; BARANOV, 2011).

Os navegadores são executados dentro de um sistema operacional, a Tabela 2 apresenta o suporte aos sistemas operacionais *desktop*.

Tabela 2. Sistemas Operacionais Suportados Oficialmente

	Adobe Flex	Silverlight	JavaFX
Windows	Sim	Sim	Sim
Mac OS X	Sim	Sim	Sim
Linux	Sim	Com Moonlight	Sim
Solaris	Sim	Não	Sim

O Silverlight tem foco nos sistemas operacionais Windows e Mac OS, no entanto existe o Moonlight que é uma implementação de código aberto para os sistemas operacionais baseados em Unix criado pela Novell com colaboração da Microsoft (NOVELL, 2009, tradução nossa).

6.2.2 Tamanho dos *Plugins* e Dificuldade de Instalação

O tamanho dos *plugins* e a dificuldade de instalação são fatores que interferem diretamente na acessibilidade da aplicação. Outro ponto é taxa de penetração que aumenta as chances de um usuário conseguir acessar a aplicação com facilidade, pois se o usuário já tiver instalado as dependências poderá acessar normalmente a aplicação.

Como demonstrado na Tabela 3, para executar dentro do navegador o Adobe Flex se destaca pelo tamanho do *plugin*, porém ele necessita do Adobe AIR *runtime* para executar fora do navegador. O Silverlight e o JavaFX necessitam apenas de um único *runtime* para executar dentro ou fora do navegador, no entanto a dependência do JavaFX é a maior.

Tabela 3. Características dos *Plugins* e *Runtimes*

Descrição	Flash Player 10.3	Adobe AIR 2.6	Silverlight 4	JRE 1.6u24 x86
Plataforma	Adobe Flex no Navegador	Adobe Flex fora do Navegador	Silverlight	JavaFX
Tamanho	~3,22MB	~12,20MB	~6MB	~15,77MB

Para ter uma base da taxa de adoção foram utilizados como fonte os sites Stat OWL e RIA Stats que fornecerem estatísticas de uso e de penetração das plataformas RIA, como apresentado na Tabela 4. Apesar do JRE mostrar uma boa taxa de adoção apenas a versão JRE 1.6 oferece suporte ao JavaFX. Além do Flash Player ter uma alta taxa de adoção, o *plugin* já vem embutido no navegador Google Chrome.

Tabela 4. Taxas de Penetração das Plataformas RIA

Descrição	Flash Player	Silverlight	JRE
Stat OWL	~96.18%	~71.62%	~65,57%
RIA Stats	~95,86%	~61.42%	~78.13%
Média	~96,02%	~66,52%	~71.85%

6.2.2.1 A Dificuldade de Instalação

Quando o usuário não tem instalado o *plugin*, o mesmo pode encontrar dificuldades na hora da instalação, precisando de um suporte técnico. Para mensurar a dificuldade de instalação foram usados como métricas a quantidade de passos que o usuário precisa interagir com o sistema para efetuar a instalação de cada plataforma, e também a exibição automática do conteúdo após a instalação e mensagens em idioma estrangeiro.

Os testes foram realizados em uma maquina virtual com o Windows XP SP2 32bits com idioma em português do Brasil e com o navegador padrão Internet Explorer 6. O teste consistiu em acessar um conteúdo criado em cada plataforma. As páginas que hospedam o conteúdo são as páginas padrões gerado por cada ferramenta, sem qualquer customização de imagens e instruções de download do *plugin*. Diante desse cenário obtivemos os seguintes resultados:

a) o Flash Player apresentou uma janela requisitando a instalação e precisou de apenas 3 interações com o usuário para completar a instalação, logo após a instalação o conteúdo foi carregado e exibido automaticamente e não teve nenhuma mensagem em idioma estrangeiro.
b) o Silverlight apresentou uma imagem no idioma do sistema operacional com instruções. No total foram seis interações para concluir a instalação. No entanto ao termino o instalador avisa que talvez seja necessário uma atualização da

página para o conteúdo ser exibido, e de fato o conteúdo não foi exibido necessitando a atualização da página ou o reinicio do navegador, nenhuma mensagem estava em idioma estrangeiro.

c) o JavaFX apresentou uma imagem de uma pequena xícara sem instrução alguma para clicar nela. Depois de clicar o usuário é redirecionado à página de download em idioma estrangeiro. No total foram necessárias seis interações para concluir a instalação, ao termino é redirecionado à página inicial e o conteúdo é exibido automaticamente. Apenas a página de download estava em idioma estrangeiro.

A Tabela 5 apresenta os resultados de forma resumida:

Tabela 5. Dificuldade de Instalação dos *Plugins*

Descrição	Flash Player	Silverlight	JRE
Passos	3	6	6
Mostra conteúdo no final	Sim	Não	Sim
Mensagem em idioma estrangeiro	Não	Não	Sim

6.3 FERRAMENTAS DISPONÍVEIS

Para o desenvolvimento do protótipo em Adobe Flex foi utilizado o Adobe Flash Builder 4 Premium que é uma IDE não gratuita, especializada em codificação e também possui editor visual possibilitando arrastar componentes, visualizar e editar seus atributos. Para design é recomendado o uso da ferramenta Adobe Flash Catalyst, que possui integração com outras ferramentas de design como Adobe Illustrator, Photoshop e Fireworks.

No Silverlight a IDE utilizada é o Microsoft Visual Studio 2010, que é a mesma utilizada pela plataforma *Microsoft .Net*. A ferramenta possui uma versão gratuita chamada de Visual Web Developer 2010, para construção de aplicativos em Silverlight e ASP nas linguagens C# e VB.Net. Ela é especializada em codificação mas possui editor visual.

Para design das aplicações em Silverlight é recomendado o uso do Expression Blend, que é uma ferramenta não gratuita especializada em design de aplicações Silverlight e WPF. Essa ferramenta faz parte do Expression Studio, que é um conjunto de ferramentas de design da Microsoft. Ela também possui um editor de código básico e possui integração com as versões não gratuitas do Visual Studio 2010.

A principal IDE utilizada pelo JavaFX é o Netbeans que é a também utilizada pela plataforma Java. Ela é totalmente gratuita e fornece editor visual.

A Tabela 6 mostra a relação das ferramentas que podem ser utilizadas para construção de aplicativos ricos em suas respectivas plataformas. Os preços foram coletados diretamente do site dos fabricantes, sendo preços sugeridos de uma unidade. Lembrando que é necessário apenas uma versão do Flash Builder e uma versão do Visual Studio 2010.

Tabela 6. Ferramentas de Desenvolvimento

Nome	Valor	Plataforma	Opcional
Flash Builder 4.5 Standard Edition	$289,00	Adobe Flex	Não
Flash Builder 4.5 Premium Edition	$805,00	Adobe Flex	Não
Flash Catalyst CS5.5	$486,00	Adobe Flex	Sim
Visual Web Developer 2010	Gratuita	Silverlight	Não
Visual Studio 2010 Professional	$799,00	Silverlight	Não
Expression Studio 4 Ultimate	$599,00	Silverlight	Sim
Netbeans	Gratuita	JavaFX	Não

6.4 LINGUAGENS SUPORTADAS

Aprender uma linguagem totalmente nova pode ser um empecilho para os desenvolvedores já habituados com outra linguagem. Esse aspecto pode aumentar o custo e o tempo de desenvolvimento de um aplicativo.

No Adobe Flex utiliza-se a linguagem ActionScript 3, que é a linguagem utilizada em projetos Flash. ActionScript 3 é uma linguagem orientada a objetos, baseada em ECMAScript, que é a mesma base do JavaScript (ADOBE SYSTEMS INCORPORATED,

2011, tradução nossa). Para definição dos componentes visuais é utilizada a linguagem de marcação MXML, que é uma linguagem similar ao XML. Com MXML também é possível declarar objetos não visuais como o objeto *PresentationModel* (ADOBE SYSTEMS INCORPORATED, 2010c, tradução nossa). Também é utilizada para definir o estilo e formatação dos elementos a linguagem CSS, a mesma utilizada em páginas HTML (POUCLET; SULLIVAN, 2011, tradução nossa).

No Silverlight é utilizado a linguagem de marcação XAML, que é baseada em XML, para definição dos componentes visuais e objetos não visuais. Com ela também é possível definir o estilo e formatação da aplicação.

Para definir a lógica e comportamento do aplicativo o Silverlight suporta as linguagens C# e VB.Net e qualquer linguagem baseada na CLR, também suporta linguagens dinâmicas como IronPython, IronRuby, Managed JScript e qualquer outra linguagem baseada na DLR. No entanto a IDE Visual Studio oferece suporte nativo apenas para C# e VB.Net (MICROSOFT CORPORATION, 2010d, tradução nossa).

Um fruto dessa flexibilidade é o Delphi Prism (também conhecido como D#), que fornece uma linguagem com a sintaxe baseada em Delphi e faz uso de todas as tecnologias da plataforma *Microsoft .Net*, inclusive o Silverlight, sendo uma solução para um aprendizado rápido de desenvolvedores familiarizados ao Delphi (EMBARCADERO TECHNOLOGIES, 2011, tradução nossa).

O JavaFX apesar de ser parte da plataforma Java possui uma linguagem própria chamada JavaFX Script, que é uma linguagem orientada a objetos, no entanto suporta um estilo declarativo e funcional. Com JavaFX Script é possível separar os componentes visuais da lógica de negócios com facilidade. Para definição do estilo e formatação dos componentes visuais é utilizado a linguagem CSS. Também é possível utilizar classes Java com JavaFX.

A Tabela 7 apresenta um resumo das linguagens suportadas:

Tabela 7. Linguagens suportadas

Categoria	Adobe Flex	Silverlight	JavaFX
Lógica	ActionScript	C#/VB.Net/IronPython/IronRuby/M.JScript	JavaFX Script/Java
Definição	MXML	XAML	JavaFX Script
Estilo	CSS	XAML	CSS

6.5 CONTROLES VISUAIS NA BIBLIOTECA INICIAL

No HTML todos os controles visuais são pré-definidos, e apesar da junção de HTML, CSS e JavaScript conseguir trazer mais riqueza visual, ainda possui limitações pois tudo tem base nos controles HTML. Em contrapartida nas plataformas RIA é possível criar os próprios controles visuais a partir de um controle base.

No entanto cada plataforma possui uma biblioteca inicial de controles visuais, sendo que há uma proximidade dos controles que são comuns em aplicativos desktop. A biblioteca inicial evita que o desenvolvedor gaste tempo para construir componentes já existentes no desktop e evita também a compra de componentes de terceiros.

Como demonstra a Tabela 8, de modo geral todos possuem os controles básicos já conhecidos em aplicações desktop. O JavaFX possui poucos controles além do básico sendo que a ausência de maior relevância é do *Datagrid* (também conhecido como tabela ou *table*) uma solução é utilizar componentes do *Swing* como o *JTable*, porém essa solução causa diferenças visuais entre os componentes e gera dependência que pode limitar o número de plataformas onde o aplicativo será executado (MORRIS, 2009, tradução nossa).

O componente *Web Browser* do Silverlight e Adobe Flash só funciona fora do navegador. Muitos componentes do Silverlight fazem parte do *Silverlight Toolkit*, que é uma biblioteca da Microsoft de código livre e independente das versões do Silverlight, no entanto a versão mais atualizada é instalada automaticamente junto com o SDK.

Tabela 8. Controles Iniciais

Nome	Adobe Flex	Silverlight	JavaFX
Básicos			
TextBox	Sim	Sim	Sim
Button	Sim	Sim	Sim
CheckBox	Sim	Sim	Sim
ComboBox/SelectBox	Sim	Sim	Sim
Label	Sim	Sim	Sim
ListBox/ListView	Sim	Sim	Sim
RadioButton	Sim	Sim	Sim
Containers/Layout	Sim	Sim	Sim
Datagrid/Table	Sim	Sim	Não
ProgressBar	Sim	Sim	Sim
Slider	Sim	Sim	Sim
ToggleButton	Sim	Sim	Sim
Formas Geométricas	Sim	Sim	Sim
Avançados			
Color Picker	Sim	Não	Não
DateChooser/DatePicker	Sim	Sim	Não
TreeView	Sim	Sim	Em Teste
Gráficos	Sim	Sim	Em Teste
Window	Sim	Sim	Não
RIchTextBox	Sim	Sim	Não
WebBrowser/HTML	Sim	Sim	Não
Tab-Control/Navigator	Sim	Sim	Não
Outros			
LinkButton	Sim	Sim	Sim
Numeric UpDown/NumericStepper	Sim	Sim	Não
Accordion	Sim	Sim	Não
TimePicker	Não	Sim	Não
ContextMenu/PopupMenu	Sim	Sim	Em Teste
Expander	Não	Sim	Não
Rating	Não	Sim	Não
Multimídia			
Imagem	Sim	Sim	Sim
Som	Sim	Sim	Sim
Vídeo	Sim	Sim	Sim
Flash	Sim	Não	Não

6.6 ANIMAÇÕES, TRANSFORMAÇÕES E EFEITOS VISUAIS

O suporte à animações pode trazer mais riqueza à aplicação adicionando movimento e interatividade. As plataformas RIA tem suporte à animações com transformações geométricas, filtros visuais entre outros recursos por meio de frameworks de fácil usabilidade. Porém a capacidade e maneira de execução desses recursos variam de uma plataforma para outra.

As animações nas plataformas possuem a característica de interpolar as propriedade de um objeto como cor, opacidade, altura ou largura. Podem ser aplicados nos objetos transformações 2D como rotação, translação ou mudança de escala. As plataformas Adobe Flex e Silverlight possuem transformações 3D com mudanças de ângulos nos planos *X*, *Y* e *Z*. No entanto, com JavaFX é possível realizar o efeito 3D por meio do filtro *PerspectiveTransform*, que altera as perspectivas do objeto (JOSH MARINACCI, 2008, tradução nossa).

As plataformas Adobe Flex e Silverlight possuem funções (conhecidas como *easing* ou *easy function*) que alteram o comportamento de uma animação existente, definindo a taxa de aceleração e desaceleração das animações criando efeitos realísticos de quique, elástico entre outros, de maneira simples. A Tabela 9 apresenta um resumo dessas funcionalidades:

Tabela 9. Suporte à Animações e Transformações

Descrição	Adobe Flex	Silverlight	JavaFX
Transformações 2D	Sim	Sim	Sim
Efeitos 3D	Sim	Sim	Sim
Easy Function	Sim	Sim	Não

Outra característica das plataformas é a capacidade de aplicar filtros nos componentes visuais. Os filtros não alteram mais as propriedades de um objeto na tela (como tamanho, angulo ou cor), mas sim a imagem gerada do objeto, essa técnica nas plataformas Adobe Flex e Silverlight são conhecidas como *Pixel Shader*. Os efeitos de *Pixel Shader* utilizam algoritmos que alteram os pixels da imagem gerada (ADOBE SYSTEMS INCORPORATED, 2011b, tradução nossa).

Adobe Flex e Silverlight possuem a possibilidade de criar o próprio efeito escrevendo o *Pixel Shader* e anexando ao programa. O algoritmo do *Pixel Shader* na plataforma Silverlight é construído por meio da linguagem *High-Level Shader Language* (HLSL). Na plataforma Adobe Flex é por meio da linguagem *Pixel Bender kernel language*.

A plataforma JavaFX possui mais filtros além dos listados na Tabela 10, como ajuste de cor, sombra interna, sépia entre outros. Apesar de poder criar o próprio *Pixel Shader*, o Silverlight possui apenas dois filtros prontos.

Tabela 10. Filtros Básicos

Efeito	Adobe Flex	Silverlight	JavaFX
Sombra	Sim	Sim	Sim
Ofuscar	Sim	Sim	Sim
Brilho	Sim	Não	Sim
Reflexo	Não	Nao	Sim
Blend	Sim	Não	Sim

6.7 OUTROS RECURSOS

As plataformas RIA oferecem também outros recursos que podem auxiliar o desenvolvimento ou determinar a viabilidade do aplicativo.

Como apresentado na Tabela 11, todas as plataformas oferecem armazenamento local, essa funcionalidade possibilita uma melhor execução offline do aplicativo. As

plataformas Adobe Flex e Silverlight possuem API para acesso a câmera e microfone, possibilitando por exemplo o uso da realidade aumentada. Outro recurso das duas plataformas citadas é o suporte nativo a impressão, porém o Adobe Flex possui impressão vetorial e o Silverlight apenas *bitmap* que gera resultado com qualidade inferior.

Tabela 11. Outros Recursos Nativos

Descrição	Adobe Flex	Silverlight	JavaFX
Armazenamento local	Sim	Sim	Sim
Câmera e microfone	Sim	Sim	Não
Impressão Vetorial	Sim	Não	Não
Impressão *Bitmap*	Sim	Sim	Não

6.8 PROTÓTIPO DE APLICAÇÃO

O protótipo de aplicação consiste no desenvolvimento de uma agenda nas plataformas Adobe Flex, Silverlight e JavaFX. Esse protótipo tem como intuito avaliar as ferramentas de codificação e design; linguagens de programação; manipulação de dados na web; a capacidade de alterar o estilo visual dos componentes visuais, entre outros aspectos.

A fim de deixar as aplicações parecidas em cada plataforma, buscou-se não utilizar recursos ou componentes visuais que não estivessem presentes em todas as plataformas e também se utilizou o padrão PM (ou MVVM) para o código ter as mesmas características de implementação, além de ser o padrão recomendado pela Adobe e Microsoft.

As ferramentas de desenvolvimento utilizadas nesse protótipo foram:

a) Microsoft Visual Web Developer 2010 Express SP1, versão gratuita do Visual Studio 2010, para criação do aplicativo no servidor em ASP e do cliente em Silverlight 4;

b) Adobe Flash Builder 4 para criação do cliente em Adobe Flex 4.1;

c) e na construção do cliente em JavaFX 1.3 utilizou-se o Netbeans 6.9.1.

6.8.1 Concepção e Modelagem do Protótipo

A fim de avaliar as plataformas RIA na criação de aplicativos ricos para web criou-se um protótipo com requisitos comumente encontrados em um sistema real como cadastro, exclusão, edição e listagem de dados, mas de maneira sucinta. A partir dessa etapa constatou-se que a criação de uma agenda preencheria todas as características do cenário possibilitando os testes e avaliações.

6.8.1.1 Funcionamento do Sistema e Identificação de Requisitos

A partir da escolha do cenário do protótipo foram estabelecidos os seguintes requisitos:

a) o usuário devera fazer o login para ter acesso a suas informações;
b) caso o usuário não esteja cadastrado, poderá se cadastrar acessando a tela de cadastro por meio da tela de login;
c) na tela de cadastro o usuário poderá criar um nova conta a partir de seus dados podem não poderá escolher um nome de login que esteja em uso;
d) para facilitar o usuário poderá verificar a disponibilidade do nome de login ou voltar para o tela de login;
e) uma vez que o nome de login e senha estejam corretos o usuário poderá escolher entre listar suas anotações ou cadastrar uma nova anotação por meio de um menu no topo;

f) na tela de listagem o usuário poderá filtrar por mês e ano da anotação ou palavras que estejam no titulo ou mensagem da anotação;

g) também na tela de listagem após a escolha de uma anotação o usuário poderá excluir ou editar a anotação;

h) na tela de cadastro o usuário poderá limpar os dados por meio de um botão;

As Figuras 10 e 11 trazem um esboço da interface da Agenda:

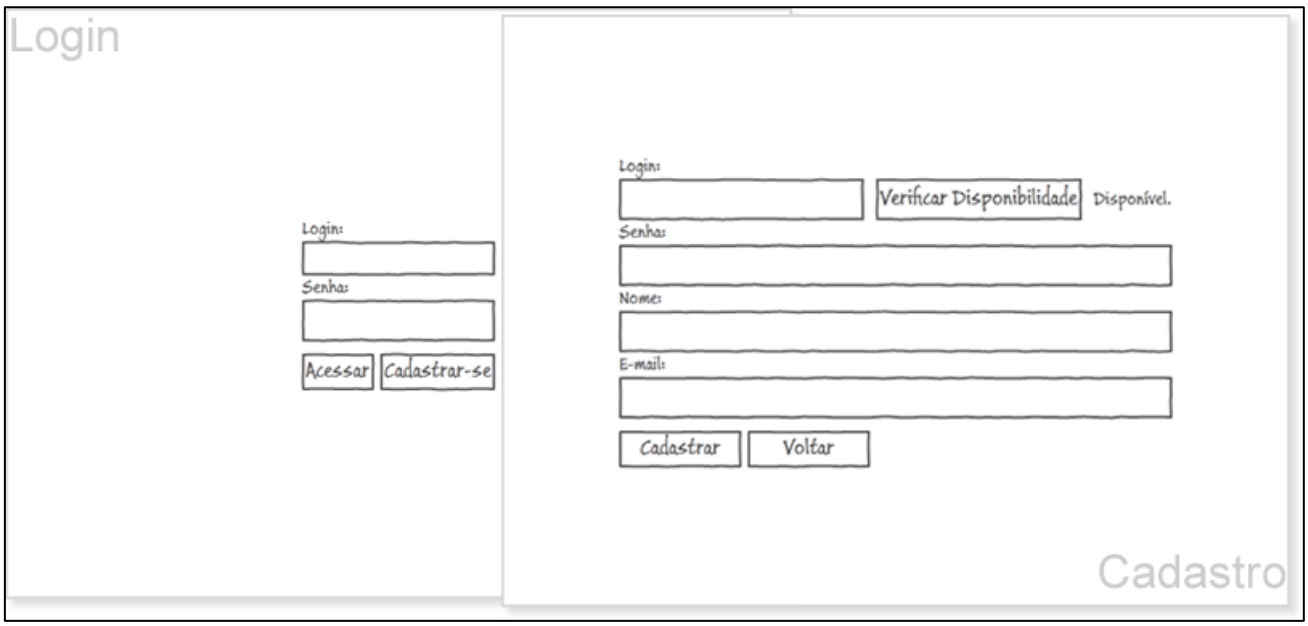

Figura 10. Esboço da tela de login e cadastro do usuário

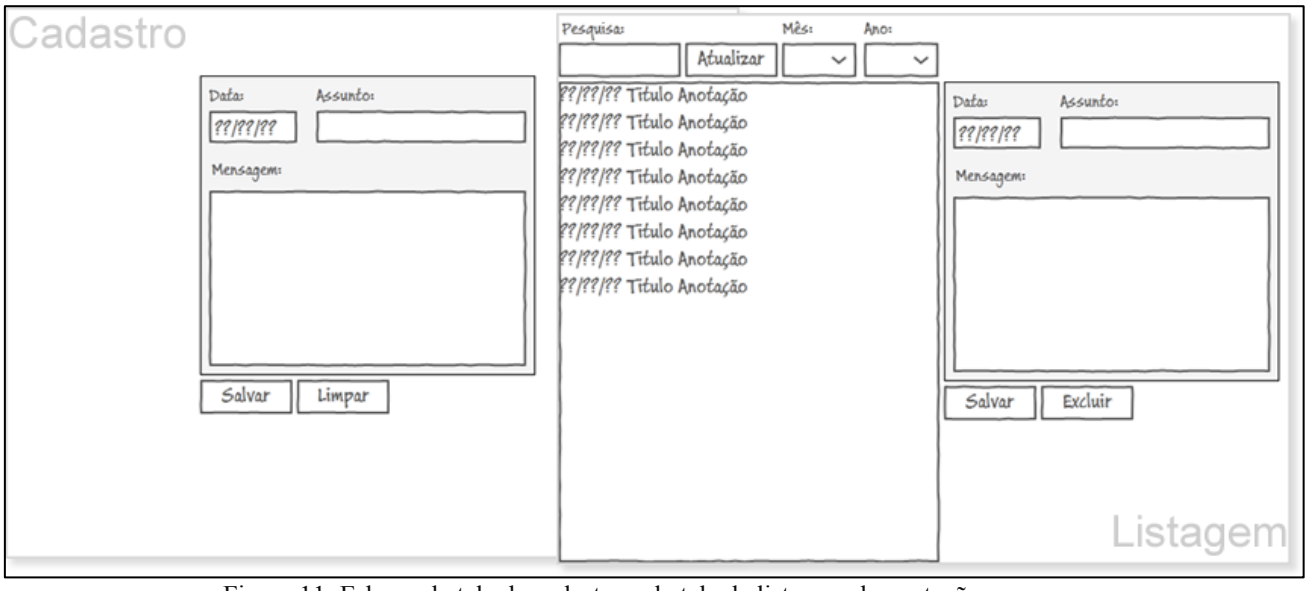

Figura 11. Esboço da tela de cadastro e da tela de listagem de anotações

6.8.1.2 Análise e Levantamento de Requisitos

A Tabela 12 apresenta os requisitos funcionais do protótipo desenvolvido.

Tabela 12. Listagem de Requisitos Funcionais

Requisito	Descrição	Oculto	Evidente
F01: Cadastro de usuário	Um novo usuário poderá cadastrar-se no sistema através de um formulário na aplicação preenchendo seus dados.		X
F02: Autenticação do usuário	O usuário deverá se autenticar para ter acesso ao sistema.		X
F03: Exibir listagem por mês e ano	O sistema deverá exibir as anotações dentro de um determinado mês e ano.		X
F04: Filtrar listagem com palavras	O sistema deverá permitir que o usuário coloque palavras para pesquisar anotações que contenham tais palavras no titulo ou mensagem, refinando a pesquisa.		X
F05:Exclusão da anotação	O usuário poderá excluir uma anotação a partir da listagem.		X
F06: Edição da anotação	A partir da escolha na listagem de anotações o usuário poderá editá-la e salvar as alterações;		X
F07: Cadastro de anotação	O usuário poderá cadastrar uma nova anotação no sistema.		X

A Tabela 13 apresenta a listagem dos requisitos não funcionais:

Tabela 13. Listagem de Requisitos Não Funcionais

Requisito	Restrição	Categoria	Desejável	Permanente
NF01: *Plugins*	Para realizar a execução do sistema é necessário instalar um *plugin*. O *plugin* varia de acordo com a versão do sistema referente a plataforma em que foi criada.	Software		X
NF02: Servidor remoto	Todas as aplicações versões dos aplicativos terão conexão com o mesmo servidor.	Software/arquitetura		X

6.8.1.3 Casos de Uso Levantados

A Tabela 14 apresenta de forma resumida os casos de uso encontrados:

Tabela 14. Listagem de Casos de Uso em Forma Resumida

Nome	Descrição
C01	Efetuar login/autenticação.
C02	Cadastrar novo usuário no sistema.
C03	Navegar pela aplicação
C04	Exibir listagem de anotações de acordo com a pesquisa.
C05	Editar anotação.
C06	Excluir anotação.
C07	Cadastrar nova anotação.
C08	Limpar formulário de cadastro de anotação.

6.8.1.4 Diagrama de Casos de Uso e de Classes

A Figura 12 apresenta os casos de uso da aplicação:

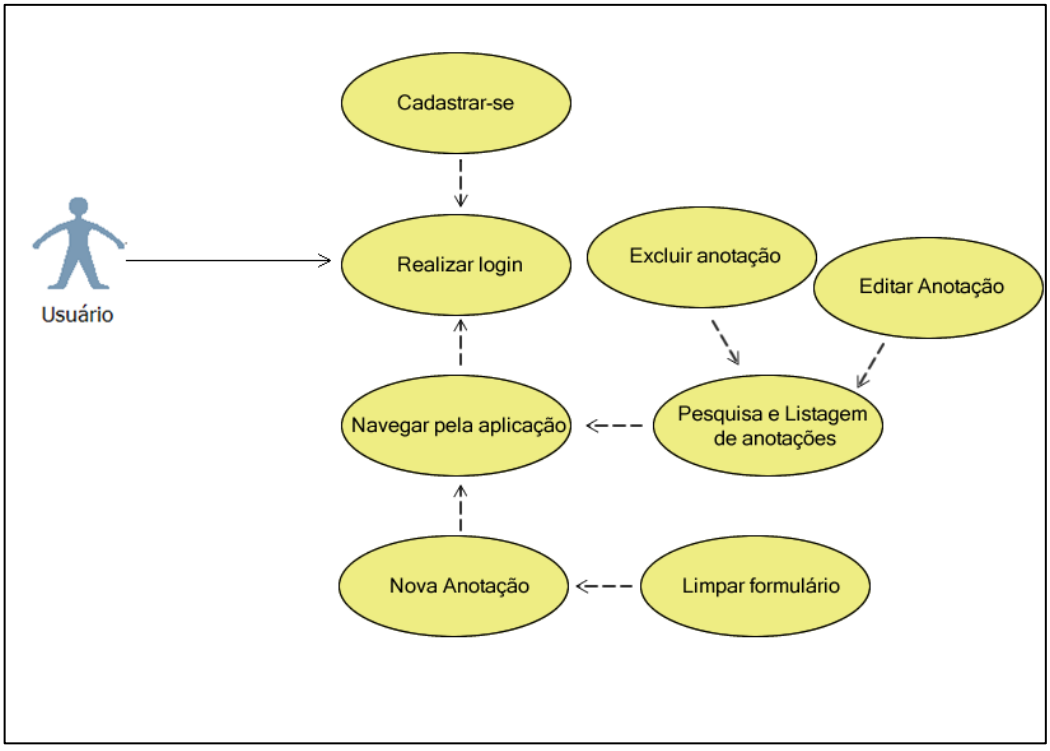

Figura 12. Diagrama de casos de uso

A Figura 13 apresenta o diagrama das classes principais do protótipo:

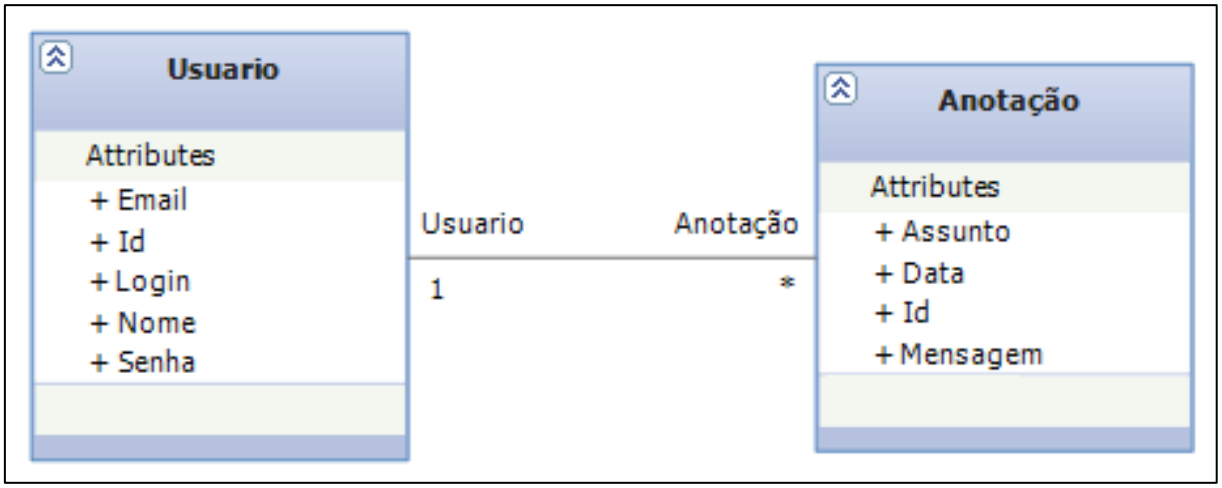

Figura 13. Diagrama das classes principais do protótipo.

6.8.2 A Construção do Protótipo

Após o levantamento das informações, modelagem, escolha das ferramentas e tecnologias passou-se para a fase de desenvolvimento da aplicação. As aplicações possuem a seguinte arquitetura:

a) o cliente desenvolvido na plataforma RIA;

b) o servidor web remoto que é encarregado de receber as requisições das aplicações e enviar um retorno.

6.8.2.1 Servidor Web

Construiu-se o servidor de aplicações web por meio da plataforma ASP, com o uso do framework ASP MVC 3, ferramenta Visual Studio 2010 e a linguagem C#. Foi escolhido o framework ASP MVC 3 por ele suportar todos os protocolos necessários para

comunicação com os clientes criado em Adobe Flex, Silverlight e JavaFX. O aplicativo no servidor tem sua arquitetura composta em duas partes apresentado na Figura 14:

a) o domínio que possui as entidades, camada de persistência e validações;

b) a interface web possui *web services* que expõem métodos de listagem e a manipulação dos dados do domínio em diferentes protocolos.

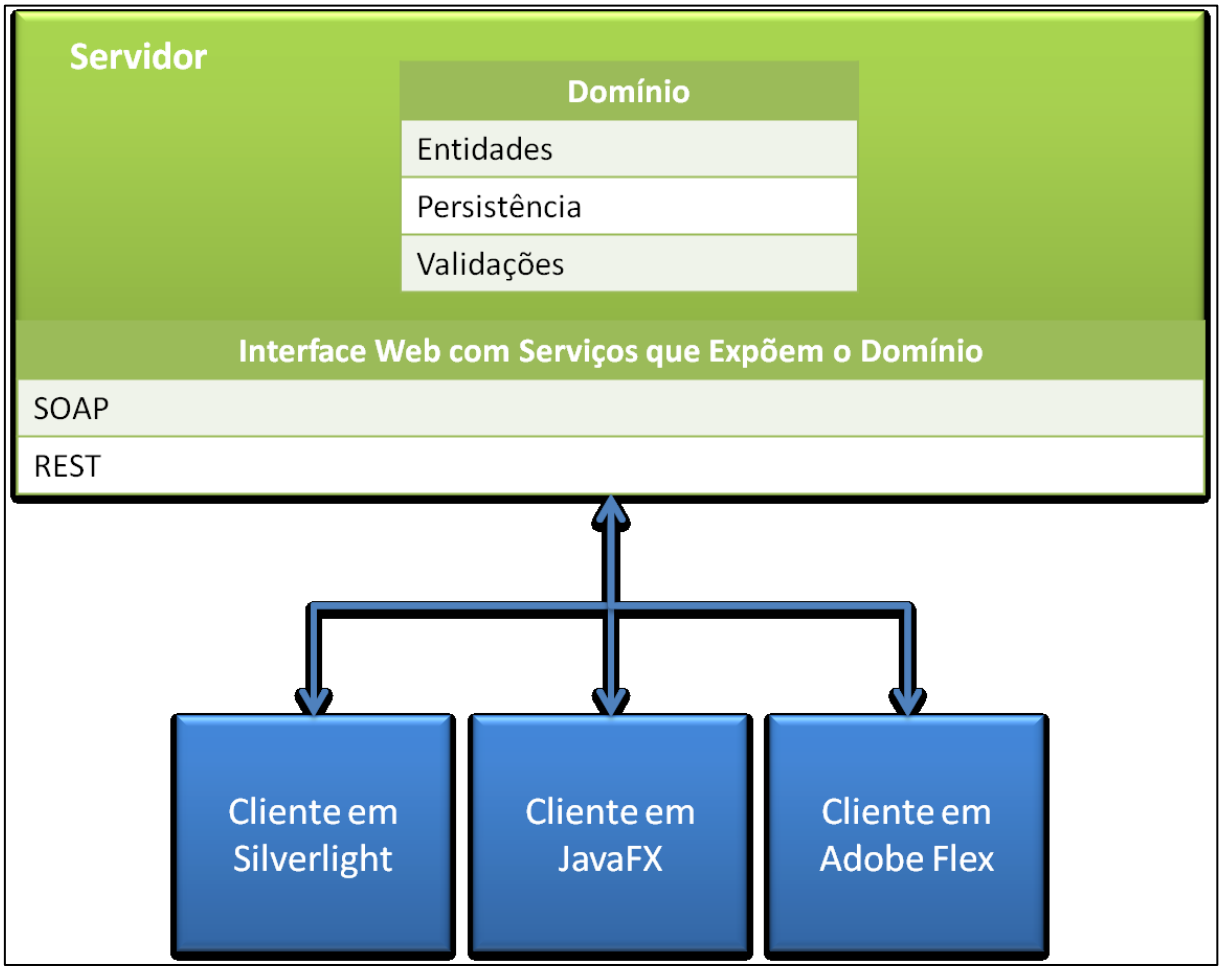

Figura 14. Arquitetura do servidor dos aplicativos

A implementação dos *web services* foram graduais a medida que as ferramentas e plataformas foram exploradas. Por fim foram desenvolvidos serviços diferentes para cada uma das plataformas visando tirar proveito das ferramentas e também suprir as deficiências de cada plataforma. Desse modo criou-se um serviço em SOAP para o Adobe Flex, outro em

SOAP para Silverlight porém com comportamento alterado para o mesmo apresentar mensagem de erro e um serviço RESTful para atender as necessidades da plataforma JavaFX.

Os métodos expostos aos clientes estão listados na Tabela 14, no entanto os nomes dos métodos, entradas e retorno podem variar dependendo do *web service*.

Tabela 15. Métodos do Servidor

Nome	Dados de Entrada	Retorno
AddAnotacao	UsuarioId, Anotacao	Nada ou erro
AddUsuario	UsuarioId	Nada ou erro
ExcluirAnotacao	UsuarioId, Anotacao	Nada ou erro
Login	Login, Senha	Usuário ou erro
LoginExiste	Login	Verdadeiro ou falso ou erro
PesquisaAnotacao	UsuarioId, Frase, Mes, Ano	Coleção de Anotação
UpdateAnotacao	UsuarioId, Anotacao	Nada ou erro

Utilizou-se o sistema gerenciador de banco de dados (SGBD) Microsoft SQL Server 2008 R2 para armazenamento dos dados.

6.8.2.2 Criação do Projeto

Visual Studio na criação do projeto do Silverlight inicialmente proporciona a escolha de linguagem, podendo optar por C# ou VB.Net, e modelo de projeto. Escolheu-se a linguagem C# e o modelo *Silverlight Application* que é um modelo limpo com apenas uma página principal, como apresentado na Figura 15. Logo após essa etapa, é possível optar por versões antigas do Silverlight e em criar um projeto web em ASP para servir de servidor da aplicação possuindo uma integração natural entre as plataformas do *.Net Framework*. Porém optou-se por não criar, pois o servidor está separado do projeto.

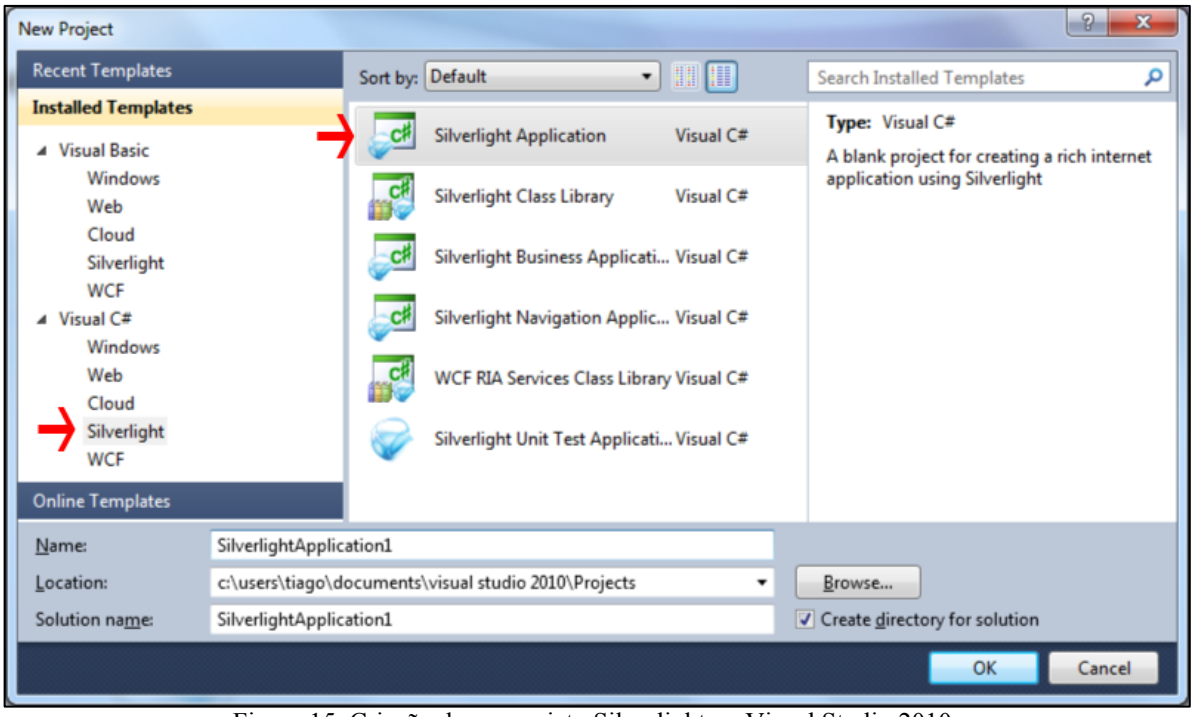

Figura 15. Criação de um projeto Silverlight no Visual Studio 2010.

A ferramenta Flash Builder na criação de um projeto Adobe Flex, proporciona mais escolhas referente aos tipos de servidor, possuindo compatibilidade com *ASP.NET*, ColdFusion, J2EE e PHP, como apresentado na Figura 16. Também é necessário escolher se o aplicativo irá executar dentro do navegador através do Flash Player ou desktop com Adobe AIR. Inicialmente escolheu-se a opção do tipo de servidor Nenhum/Outro (*None/Other*) e também optou-se pela execução no navegador com Flash Player.

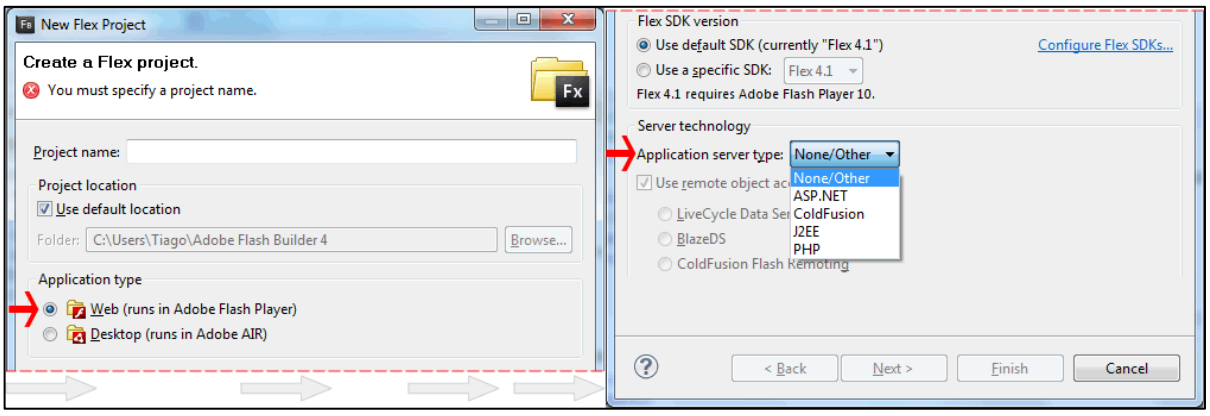

Figura 16. Criação de um projeto Adobe Flex no Flash Builder 4.

Na criação do projeto JavaFX no Netbeans é apenas necessário escolher o modelo de projeto, sendo que escolheu-se o modelo *Aplicativo JavaFX Script*, como apresentado na Figura 17. Por fim todos possuem opções básicas de nomear o projeto e definir o local.

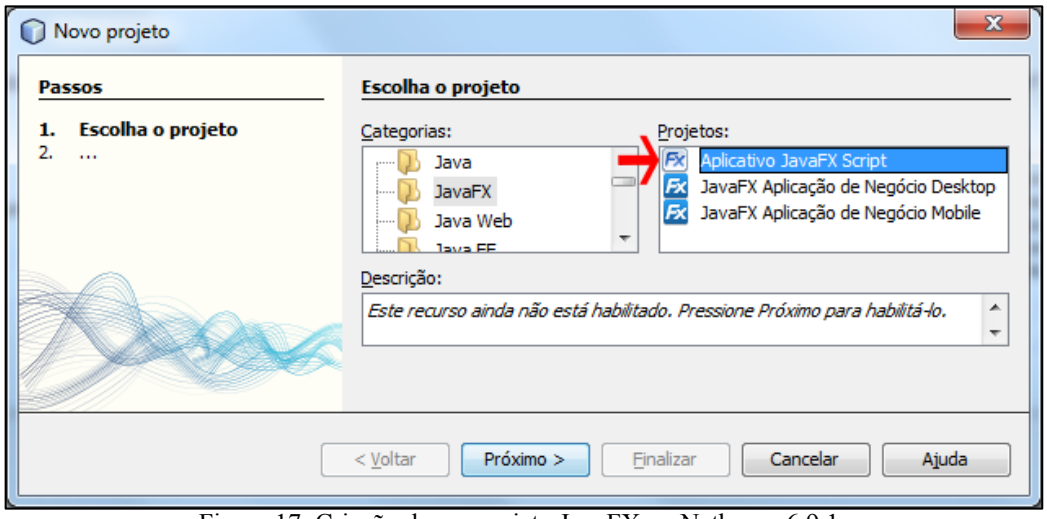

Figura 17. Criação de um projeto JavaFX no Netbeans 6.9.1.

6.8.2.3 Editor Visual

Todas as ferramentas possuem editor visual básicos possibilitando arrastar componentes e alterar suas propriedades.

No entanto se utilizar o Netbeans dessa forma o código gerado não poderá ser alterado dificultando uma alteração rápida e o uso do recurso de *binding* oferecida pela linguagem JavaFX Script. Outro ponto constatado é que em livros ou tutorias oferecidos pela Oracle não são utilizados o editor visual do Netbeans. Porem para facilitar a IDE disponibiliza a paleta de componentes dentro do código, como demonstrado na Figura 18. Outro recurso é a pré-visualização da tela, no entanto esse recurso só funciona com a tela principal do programa, tonando-se inútil em um programa com mais janelas ou na construção de um novo componente visual.

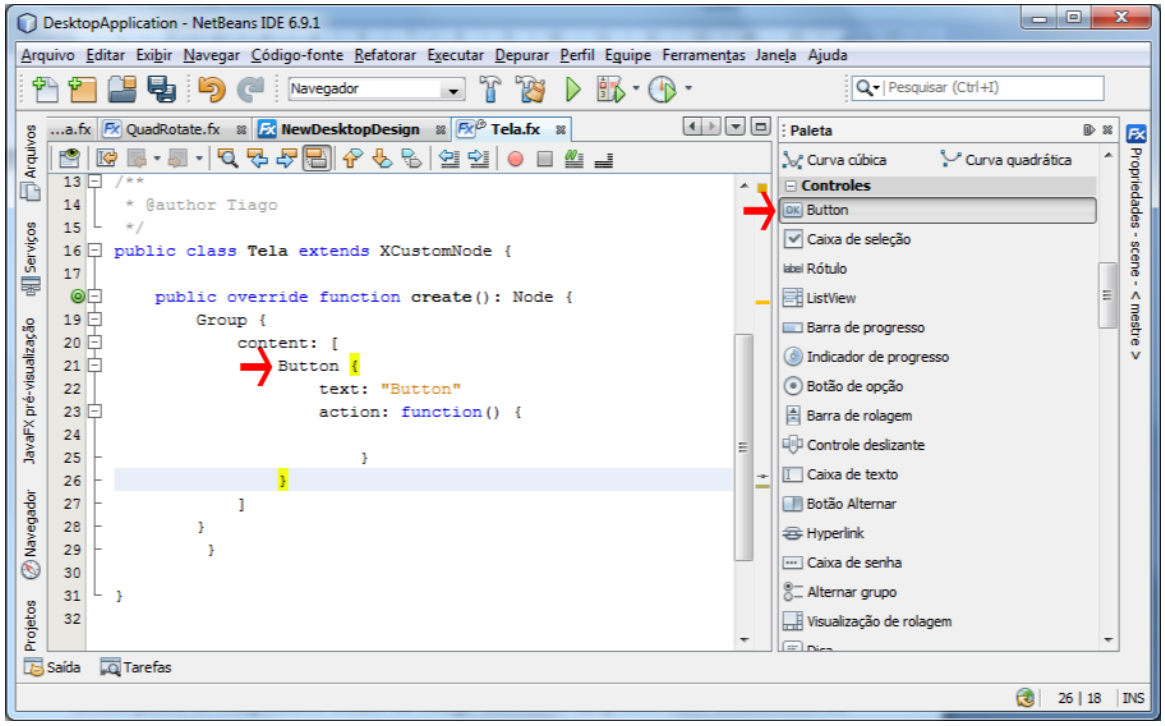

Figura 18. Paleta de controles via código no Netbeans 6.9.1.

Os controles visuais criados pelo desenvolvedor ficam disponíveis também na paleta do Visual Studio e Adobe Flex, dando suporte a visualização tal como os componentes nativos. O código gerado pelas duas ferramentas são bastante limpo graças as linguagens de marcação baseadas em XML.

Outra abordagem é a criação de estilos, *skins* ou *templates* para customização visual da aplicação.

O Flash Builder possui editor visual para a criação de um novo *skin*. Com um botão, por exemplo, é possível navegar entre seus estados e alterar o estilo. Porém o editor possui limitações como não suportar cores gradiente, por isso é recomendado outras ferramentas de designer da Adobe. Na criação de arquivos de estilo o Flash Builder ajuda com o código CSS.

Com o Visual Studio não é possível criar novos estilos ou *templates,* no entanto é possível visualizar os estilos e *templates* existentes. Para customização visual é recomendado

o uso da ferramenta Expression Blend que possui integração direta com o Visual Studio. E na criação de arquivos de estilo o Visual Studio ajuda com o código XAML.

O JavaFX conta apenas com a criação de estilos por meio do CSS e o Netbeans além de não ajuda com o código, sugere as propriedades de elementos HTML ao invés dos componentes em JavaFX.

Apesar das diferenças de implementação e ferramentas os protótipos se aproximam visualmente.

6.8.2.4 Suporte à *Binding* e Padrão PresentationModel

O padrão PM ou MVVM consiste na separação da visão (interface do usuário) por meio de um objeto que abstrai a visão, definindo seu comportamento e estado, esse objeto mediador é chamado de ViewModel ou PM. A visão conhece e observa o ViewModel, e o ViewModel não conhece a visão, sendo que qualquer mudança nos dados do ViewModel é refletida na visão.

Um exemplo simples é a tela de login do protótipo que possui dois componentes de entrada de texto, para login e senha, e outro componente botão para dar acesso ao sistema. Porém como o objeto mediador abstrai a visão, ele é quem possui os dados do tipo texto login e senha e o método de acesso como demonstrado na Figura 19. O desenvolvedor apenas vincula os componentes com os respectivos atributos e métodos do objeto mediador, esse mecanismo de vinculo é chamado de *binding*.

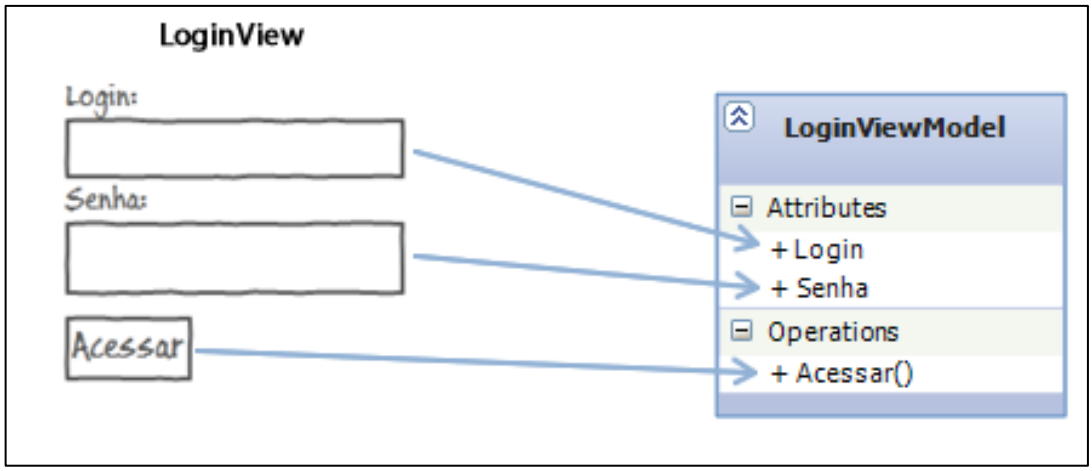

Figura 19. Ilustração do padrão PM/MVVM

Cada plataforma possui uma implementação diferente. No Adobe Flex o *binding* é realizado por meio de uma sintaxe sucinta. A sintaxe do *binding* é simples "{atributo}" ou "@{atributo}", o símbolo "@" significa que qualquer alteração realizada no componente visual irá refletir no atributo.

A Figura 20 apresenta um trecho de código MXML com *binding* que demonstra a declaração de um objeto *LoginViewModel* na visão, os componentes visuais *TextInput* vinculados aos atributos e o evento *click* do *Button* com um script para invocar o método *fazLogin*.

```
<fx:Declarations>
      <viewModel:LoginViewModel id="vm"/>
</fx:Declarations>
<s:Label text="Login:" />
<s:TextInput text="@{vm.login}" />
<s:Label text="Senha:" />
<s:TextInput text="@{vm.senha}"  displayAsPassword="true" />
<s:Button label="Acessar" click="vm.fazLogin()" />
```

Figura 20. Exemplo de *binding* no Adobe Flex com MXML

No Silverlight o *binding* não é apenas uma sintaxe com palavra chave. O *binding* normalmente é realizado por um objeto do tipo *Binding*, no entanto pode ser qualquer objeto que descenda da classe *BindingBase*. O *binding* suporta conversão, validação e formatação

dos dados. A sintaxe mais comum para *binding* é "{Binding atributo}" ou "{Binding Path=atributo, Mode=modo}", sendo que o *Mode* define se o controle visual pode ou não atualizar o objeto.

A Figura 21 apresenta um trecho de código em XAML equivalente ao da Figura 20. No trecho pode-se observar: a declaração do objeto *ViewModel* como um recurso; o mesmo preenchendo um atributo *DataContext* do container *Grid*; os componentes visuais *Textbox* e *Button* vinculados aos atributos do ViewModel.

```
<UserControl.Resources>
       <ViewModel:LoginViewModel x:Key="vm"/>
</UserControl.Resources>
<Grid DataContext="{StaticResource vm}">
       <TextBlock Text="Login:"/>
       <TextBox Text="{Binding Path=Login, Mode=TwoWay}"/>
       <TextBlock Text="Senha:" />
       <PasswordBox Password="{Binding Path=Senha, Mode=TwoWay}"/>
       <Button Command="{Binding Path=AcessarCommand}" Content="Acessar" />
</Grid>
```

Figura 21. Exemplo de *binding* no Silverlight em XAML

A linguagem JavaFX Script possibilita o uso de *binding* através da palavra chave *bind*. Para refletir as mudanças da interface gráfica no atributo são utilizadas as palavras *with inverse* como demonstrado na Figura 22.

```
def viewModel: LoginViewModel = LoginViewModel {};
def formulario = VBox {
                content: [
                    Label { text: "Login:" },
                    TextBox {
                        text: bind viewModel.login with inverse
                    }
                    Label { text: "Senha:" },
                    PasswordBox {
                        text: bind viewModel.senha with inverse
                    }
                    Button {
                        text: "Acessar",
                        onMouseClicked: function(ev) {
                            viewModel.fazLogin();
                        }
                    }
                ]
            };
```

Figura 22. Exemplo de *binding* no JavaFX em JavaFX Script

Os atributos do objeto que a visão faz *binding,* nos exemplo o *ViewModel,* precisa reportar para a mesma qualquer mudança que ocorrer em seus atributos. Cada plataforma tem seu próprio mecanismo.

Para realizar esse processo no Adobe Flex basta apenas marcar os atributos ou a classe com a *metadata Bindable*, como apresentado na Figura 23.Deste modo a classe já irá reportar as mudanças à visão.

```
[Bindable]
public class LoginViewModel
{
	public var login:String = "";
	public var senha:String = "";

	public function fazLogin():void {
		//conecta com o web service
	}
}
```

Figura 23. Classe LoginViewModel em ActionScript 3

No Silverlight é necessário realizar o processo de forma manual, que consiste na implementação da interface *INotifyPropertyChanged* e do disparo do evento no *set* de cada atributo.

Para facilitar esse processo e evitar a repetição de código utilizou-se um pequeno framework de código aberto, chamado *MVVM Light Toolkit. O* framework contém uma classe chamada *ViewModelBase,* que implementa a interface *NotifyPropertyChanged* e possui um método que facilita o disparo do evento reduzindo a quantidade de código. No entanto ainda é necessário a implementação em cada atributo como demonstrado na Figura 24.

```
public class LoginViewModel : ViewModelBase
{
    public RelayCommand AcessarCommand { get; private set; }

    private String login;
    private String senha;

    public String Login
    {
        get { return login; }
        set
        {
            login = value;
            RaisePropertyChanged("Login");
        }
    }
    public String Senha
    {
        get { return senha; }
        set
        {
            senha = value;
            RaisePropertyChanged("Senha");
        }
    }

    public LoginViewModel()
    {
        AcessarCommand = new RelayCommand(Acessar);
    }

    private void Acessar()
    {
        //acessa o web service
    }
}
```

Figura 24. Classe LoginViewModel em C#

O JavaFX Script realiza esse processo de forma transparente para o desenvolvedor, reportando qualquer mudança automaticamente, como ilustrado na Figura 25.

```
public class LoginViewModel {
    public var login: String;
    public var senha: String;

    public function fazLogin() {
        //acessa o web service
    }
```

Figura 25. Classe LoginViewModel em JavaFX Script.

6.8.2.5 Acesso ao Web Service

O acesso à *web services* são construídos a partir de protocolos simples normalmente utilizando como base o HTTP.

O Flash Builder possui uma ferramenta para auxiliar o consumo de serviços podendo escolher entre alguns tipos de protocolos e plataformas, como apresentado na Figura 26. Para o protótipo escolheu-se o *WSDL Web Service* que é o padrão implementado no servidor do protótipo. A ferramenta se encarrega de ler a *metadata* por meio do protocolo WSDL criando classes e métodos para acessar o serviço no protocolo SOAP, diminuindo o tempo de desenvolvimento.

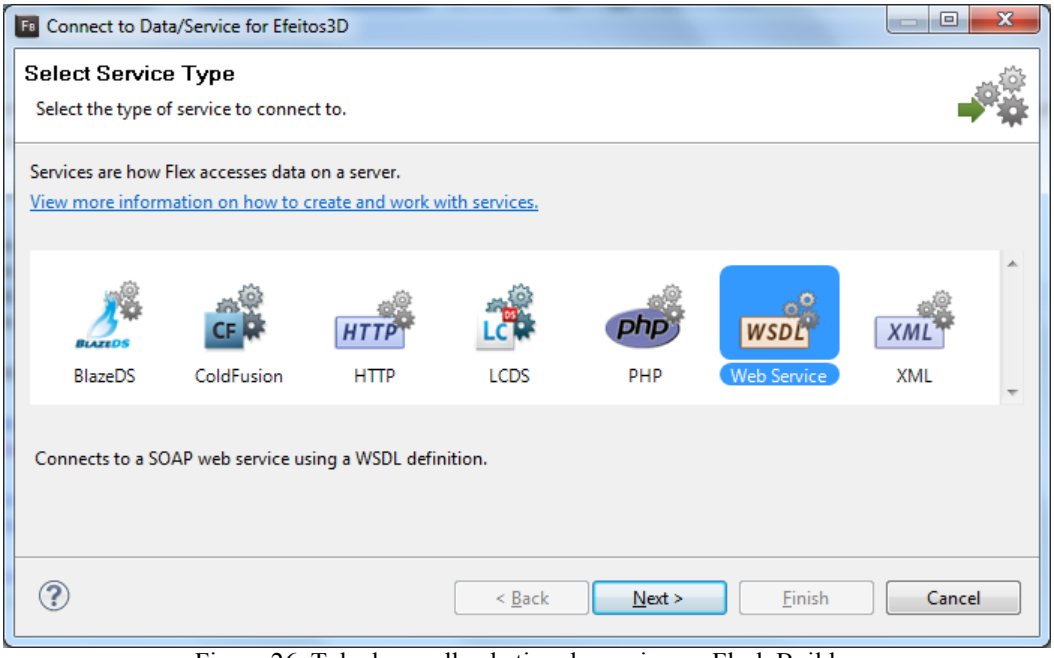

Figura 26. Tela de escolha de tipo de serviço no Flash Builder

O Flash Builder permite a escolha do nome dos pacotes onde permanecerá os códigos gerados da classe do serviço e as classes de tipos de dados. Como demonstrado na Figura 27 o servidor do protótipo expõem a classe *Anotacao* e a classe *Usuario* que foram geradas no pacote *ValueObjects* e a classe do *web service* foi gerada no pacote *services.agendabasicservice*.

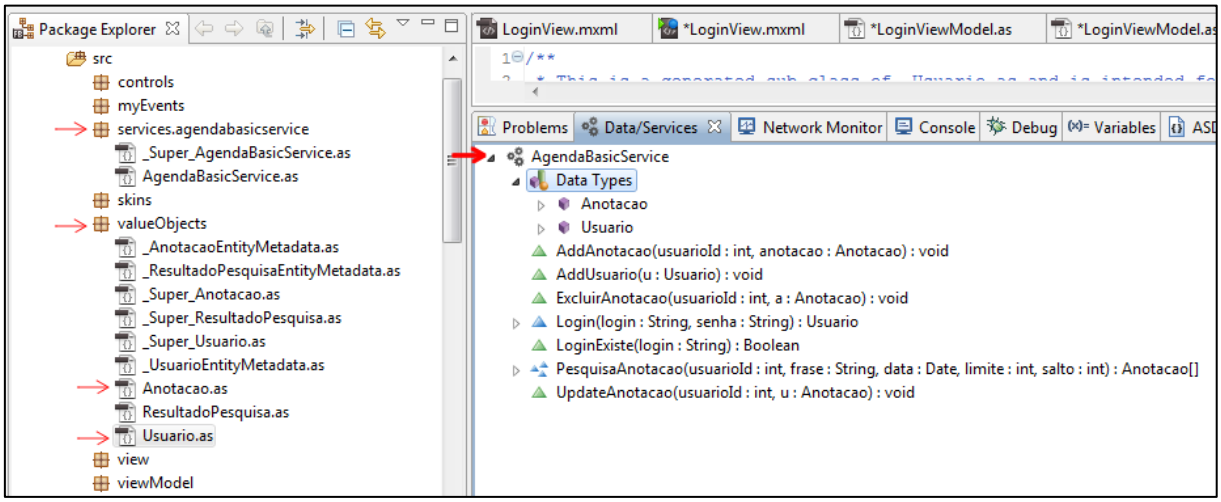

Figura 27. Classes geradas pelo Flash Builder

E ainda o Flash Builder facilita o uso do *web service*, bastando apenas escolher um botão ou um componente de exibição de dados, e escolher o método do *web service* para ele gerar automaticamente o código de uma chamada. Porém não foi utilizado esse recurso pois o Flash Builder gera o código todo na visão, indo contra a arquitetura padrão do protótipo que é a mesma recomendada pela Adobe.

A Figura 28 apresenta um trecho de código que demonstra o uso do objeto *AgendaBasicService* que encapsulada os métodos de acesso do *web service*, sua chamada e seus eventos, tornando simples o seu uso.

```
private var ws : AgendaBasicService; // web service
private var callResponder : CallResponder; //gerenciador da requisição


public function LoginViewModel()
{
	//criando instancias e associando a eventos
	callResponder = new CallResponder();
	callResponder.addEventListener(ResultEvent.RESULT, Login_Completed);
	ws = new AgendaBasicService();
	ws.addEventListener(FaultEvent.FAULT, Login_OnFault);
	ws.showBusyCursor = true;
}

public function acessar():void {
	//faz chamada do método Login
	callResponder.token = ws.Login(login, senha);
}

//dispara quando completado com a resposta pronta
private function Login_Completed(event:ResultEvent):void
{
	var result : Usuario = Usuario(event.result);
	//código omitido
}
//dispara em caso de erro
private function Login_OnFault(event:FaultEvent) : void
{
	Alert.show(event.fault.faultString);
}
```

Figura 28. Chamada de web service em ActionScript 3

O Visual Studio também possui ferramenta para consumir serviços em SOAP a partir do protocolo WSDL, como apresentado na Figura 29. Alem disso o Visual Studio possui uma solução chamada *WCF RIA Services* que integra o Silverlight com *ASP.NET*, facilitando o desenvolvimento de aplicativos em várias camadas. A principal vantagem dessa abordagem é a unificação da lógica de negócios entre servidor e cliente por meio de cópia automática de código.

No entanto utilizou-se o SOAP, pois o servidor é o mesmo para todos os clientes. Na tela de mapeamento de *web service* é possível escolher o nome do pacote (*namespace*) onde o código gerado permanecerá e nas opções avançadas pode-se escolher o tipo de coleção.

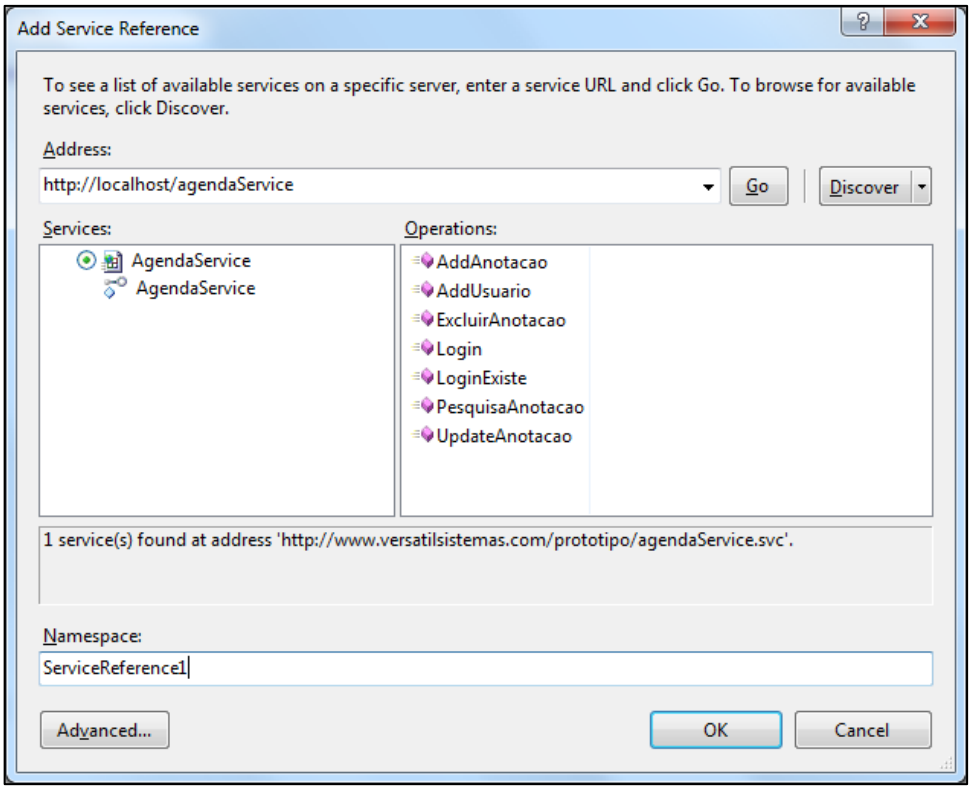

Figura 29. Tela de mapeamento de web service no Visual Studio

O código gerado fica oculto dentro da pasta *Service References*. Para utilizar o serviço basta instanciar a classe do *web service* que possui encapsulado os métodos de acesso e eventos. Como demonstrado na Figura 30 após a realização da chamada do método é disparado um evento que contem o resultado, erro entre outras informações.

```
private AgendaServiceClient ws; //web service

public LoginViewModel()
{
    //criando instancia e associando evento do ws
    ws = new AgendaServiceClient();
    ws.LoginCompleted += new EventHandler<LoginCompletedEventArgs>(ws_LoginCompleted);
}

private void Acessar()
{
   ws.LoginAsync(Login, Senha); //fazendo chamado de método
}

//dispara quando completado com resposta ou erro
private void ws_LoginCompleted(object sender, LoginCompletedEventArgs e)
{
   //codigo omitido
}
```

Figura 30 Chamada de web service em C#

Com o Netbeans não é possível mapear um serviço de forma automática gerando as classes em JavaFX Script. O JavaFX possui apenas uma classe para realizar uma requisição HTTP, por esse motivo implementou-se também um serviço RESTful que é baseado em simples requisições HTTP. Com o uso da classe *HttpRequest* é possível realizar as requisições, mas de forma repetitiva e braçal.

Para analise do resultado em XML ou JSON, o JavaFX possui a classe *PullParser*. A Figura 31 apresenta um trecho de código onde é utilizado a classe *HttpRequest* em conjunto com o *PullParser* para analisar o resultado em JSON e instanciar um objeto do tipo *Usuario*.

```
    public function fazLogin() {
        var getRequest: HttpRequest = HttpRequest {
                    location: "http://endereco/Login?login={login}&senha={senha}",
                    method: HttpRequest.GET
                    onInput: parseResponse
                    onException: function(ex: java.lang.Exception) {
                        //dispara em caso de erro
                    }
                    onDone: function() {
                        //dispara quando termina, porém precisa-se analisar o
código de retorno
                    }
                }
        getRequest.start();
    }

    //dispara quando chega dados
    function parseResponse(is: InputStream) {
        try {
            var parser = PullParser {
                        input: is
                        onEvent: parse
                        documentType: PullParser.JSON;
                    }
            parser.parse();
        } finally {
            is.close();
        }
    }

    var usuario: Usuario; //objeto que será construído
    //dispara a cada elemento do JSON
    function parse(event: Event) {
        //código omitido - analise campo a campo para reconstrução do objeto
    }
```

Figura 31. Requisição em JavaFX Script utilizando *HttpRequest*

Para reduzir a repetição de código e facilitar o uso de requisições criou-se as classes *HttpWrapper* e *ParserBase* que desacoplam e embrulham as classes *HttpRequest* e *Pullparser*. Como apresentado na Figura 32 a classe *HttpWrapper* abstrai da requisição os eventos de erros, analise e criação de objetos. Ao termino da requisição é disparado o evento com o resultado e erro.

```
public function fazLogin() {
    var request = HttpWrapper {
        endereco: "http://endereco/Login?login={login}&senha={senha}"
        metodo: HttpRequest.GET
        onCompleted: requestCompleted
        parametros: parametros
        parser: UsuarioParser {}
    }
    request.start();
}

 //dispara o evento no final com o erro e dados
public function requestCompleted(e: ResultEvent) {
     //codigo omitido
}
```

Figura 32. Requisição em JavaFX Script utilizando *HttpWrapper*

6.8.2.6 Visão Geral dos Protótipos

Os protótipos foram desenvolvidos tendo como base a mesma estrutura de arquivos. A Figura 33 apresenta os pacotes de código de cada plataforma.

Os pacotes *Controls* armazenam os controles criados, todos possuem um controle chamado *AnotacaoControl* que é o formulário da anotação. No projeto do JavaFX contem o *AnotacaoListItem,* que é o componente que representa a anotação na listagem, e o *BusyIndicator* que mostra que o programa está ocupado.

Os pacotes *View* possuem as telas e formulários. E os pacotes *ViewModel* armazenam os objetos que abstraem a visão. O restante são classes auxiliares, código gerado automaticamente ou arquivos de estilo.

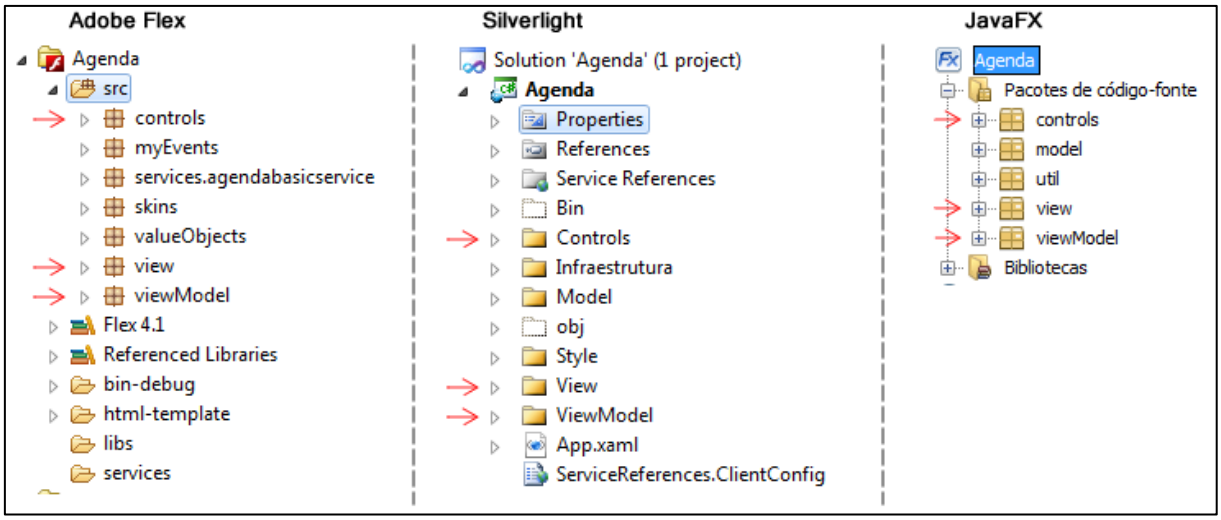

Figura 33. Estrutura dos arquivos dos protótipos

A Figura 34 apresenta a tela de login de cada protótipo desenvolvido:

Figura 34. Tela de login dos protótipos

A Figura 35 apresenta a tela de cadastro de usuário de cada protótipo desenvolvido:

Figura 35. Tela de cadastro de usuário dos protótipos

A Figura 36 apresenta as telas de listagem de anotações de cada protótipo:

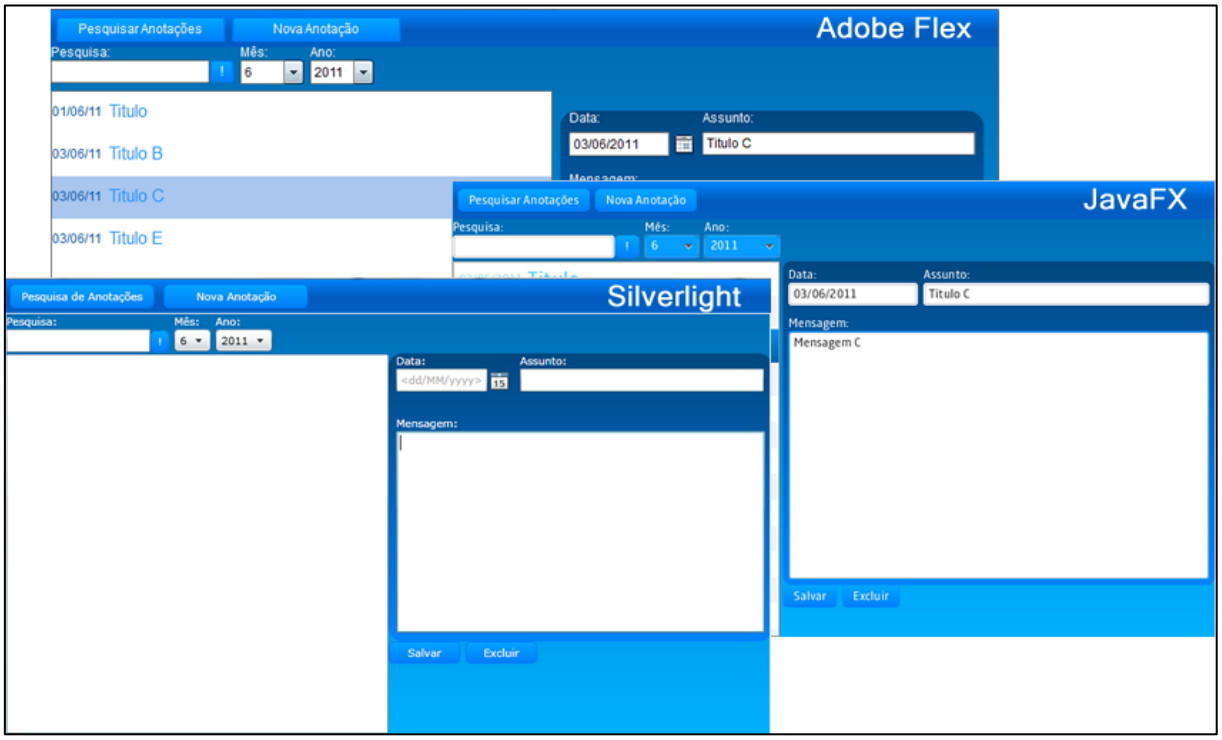

Figura 36. Tela de listagem de anotações dos protótipos

Foi possível alcançar um aspecto visual parecido em cada plataforma com a customização visual dos componentes, refletindo o esboço da etapa de modelagem.

6.9 CONSUMO DE PROCESSAMENTO

As plataformas proporcionam à aplicação um visual rico por meio das animações, efeitos entre outros recursos. Sendo que tudo é feito por meio de um cliente leve que proporciona certa acessibilidade da aplicação. No entanto uma aplicação pode ser limitada a execução em computadores atuais dependendo do consumo de processamento do aplicativo. A fim de avaliar o consumo de processamento desenvolveram-se os seguintes protótipos:

a) protótipo com animações de translação de dez objetos, rotação de dez objetos e mudança de escala de dez objetos;
b) protótipo com animações de translação de dez objetos, rotação de dez objetos, mudança de escala de dez objetos todos com efeito de sombra aplicado.

Os protótipos foram executados nos seguintes computadores:

a) notebook com processador modelo Intel I5 450M, freqüência 2.4GHZ, dual core com quatro threads, 4GB de memória RAM, VGA dedicada e sistema operacional Windows 7 64bits;
b) notebook com processador modelo Intel Celeron M410, freqüência 1.46GHZ, um core e sistema operacional Windows XP SP2 32bits.

Nos computadores foram instalados os seguintes *plugins*: *Flash Player 10.3.181.14* para execução do protótipo em Adobe Flex*; JRE 1.6u25* para execução do protótipo em JavaFX; e *Silverlight 4.0.60310.0* para execução do protótipo em Silverlight.

Para medir o uso do processador utilizou-se um aplicativo chamado *Process Monitor* de autoria desconhecida. Os protótipos foram executados dentro do navegador Firefox 4.0.1. Para os protótipos desenvolvidos em Adobe Flex e Silverlight o navegador abriu outro processo chamado *plugin-container.exe* e para o JavaFX foi aberto um processo *Java.exe*.

Buscou-se no desenvolvimento dos protótipos a similaridade de código, sendo que as três plataformas possuem implementação equivalente. Foram criados novos componentes encapsulando objetos retângulos e cada componente possui animação própria. O protótipo com efeito de sombra possui os mesmos componentes, no entanto é declarado apenas um efeito de sombra e aplicado em todos os componentes da tela. A Figura 37 apresenta uma das versões do protótipo em execução.

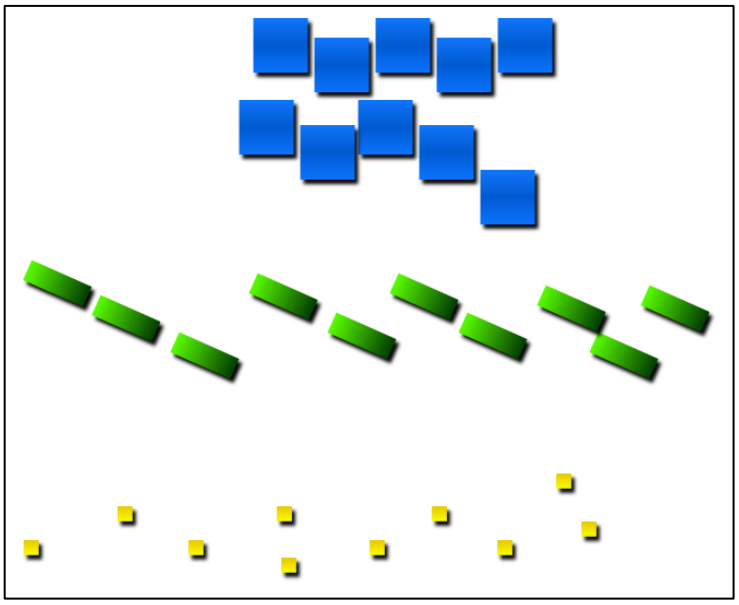

Figura 37. Protótipo de animações com efeito de sombra

O notebook Celeron M410 possui configurações modestas sendo que ele é mais obsoleto. Como apresentado na Figura 38, na execução do protótipo sem efeitos de sombra o Silverlight ganhou destaque consumindo pouco processamento. O JavaFX consumiu quase o dobro de processamento em relação ao Silverlight, porem o pior desempenho foi do Adobe Flex que consumiu mais de três vezes o processamento em ralação ao Silverlight.

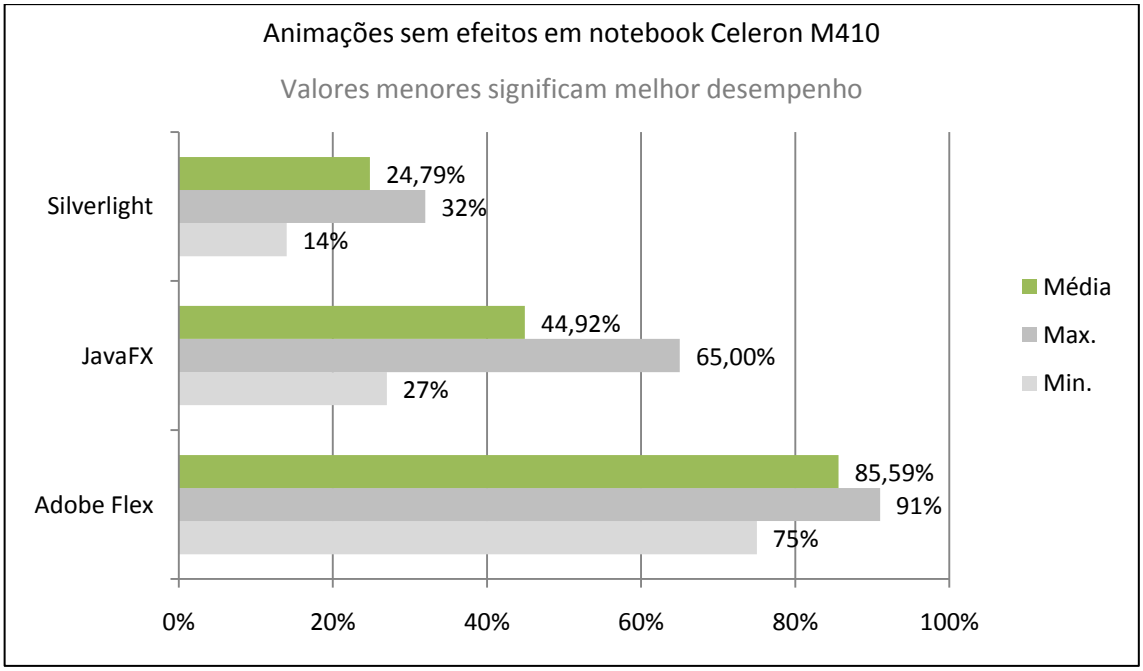

Figura 38. Desempenho do protótipo sem efeitos no notebook Celeron M410

Na execução do protótipo com efeitos visuais no mesmo cenário, todas as plataformas consumiram processamento excessivo, como demonstrado na Figura 39.

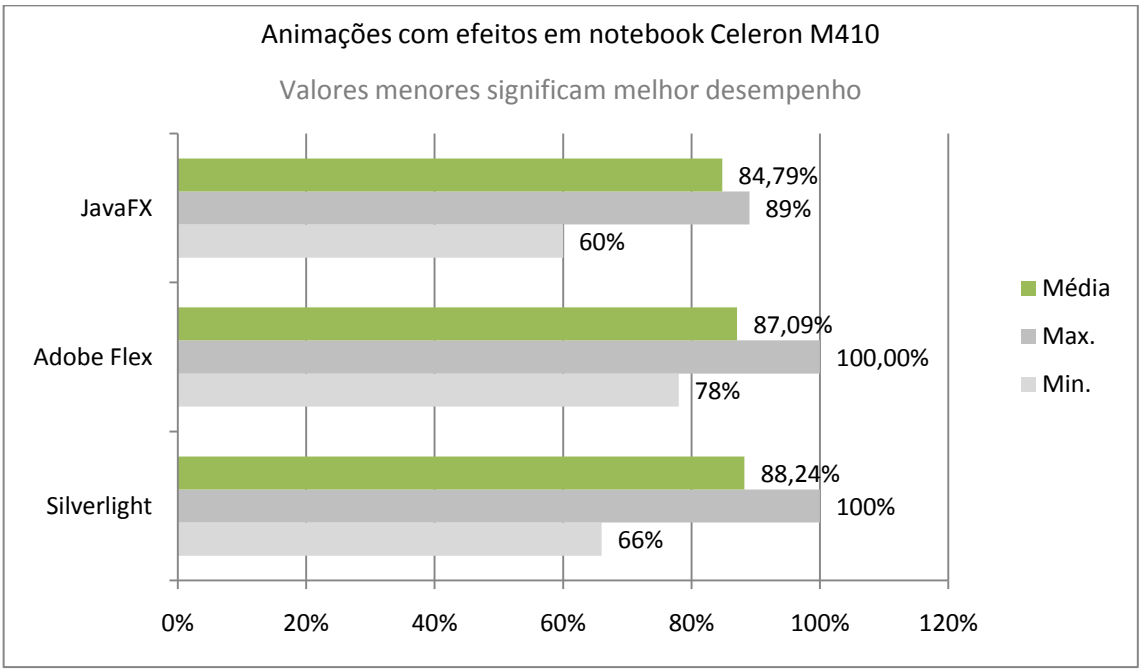

Figura 39. Desempenho do protótipo com efeitos no notebook Celeron M410

Apesar do baixo consumo geral das plataformas na execução do protótipo sem efeitos de sombra no notebook Core I5 450M, o resultado entre as plataformas foi similar, sendo que o Silverlight continuou consumindo menos processamento, como apresentado na Figura 40.

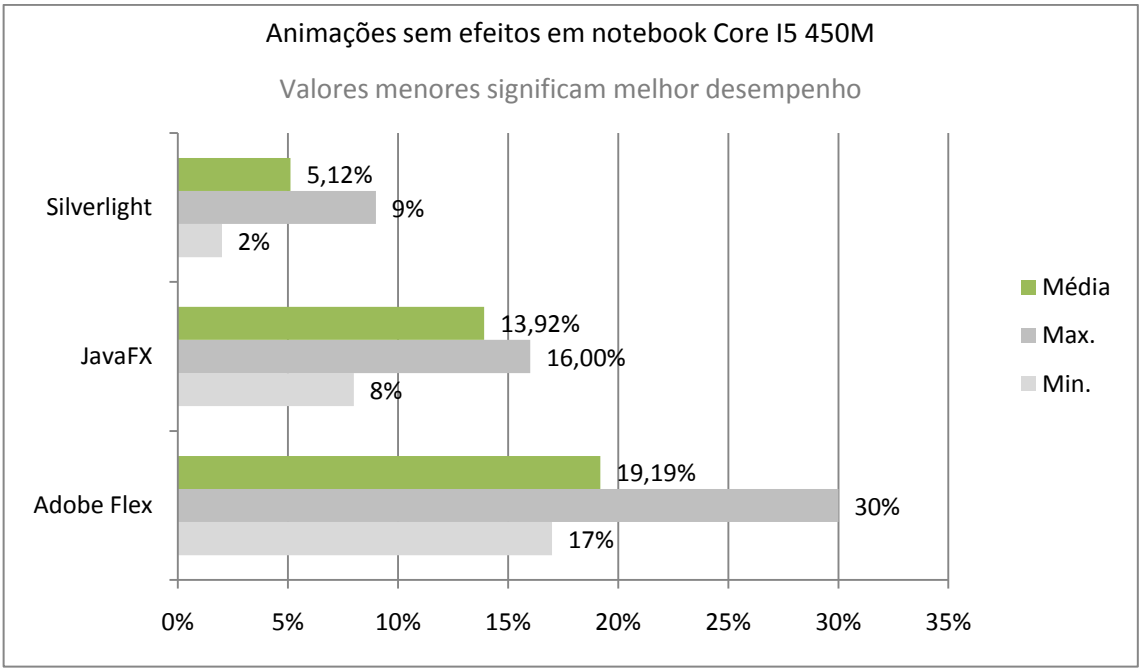

Figura 40. Desempenho do protótipo sem efeitos no notebook Core I5 M450

A execução do protótipo com efeitos visuais no mesmo cenário, também gerou resultados similares onde todas as plataformas tinham consumos parecidos. No entanto o melhor desempenho foi do Adobe Flex, como apresentado na Figura 41.

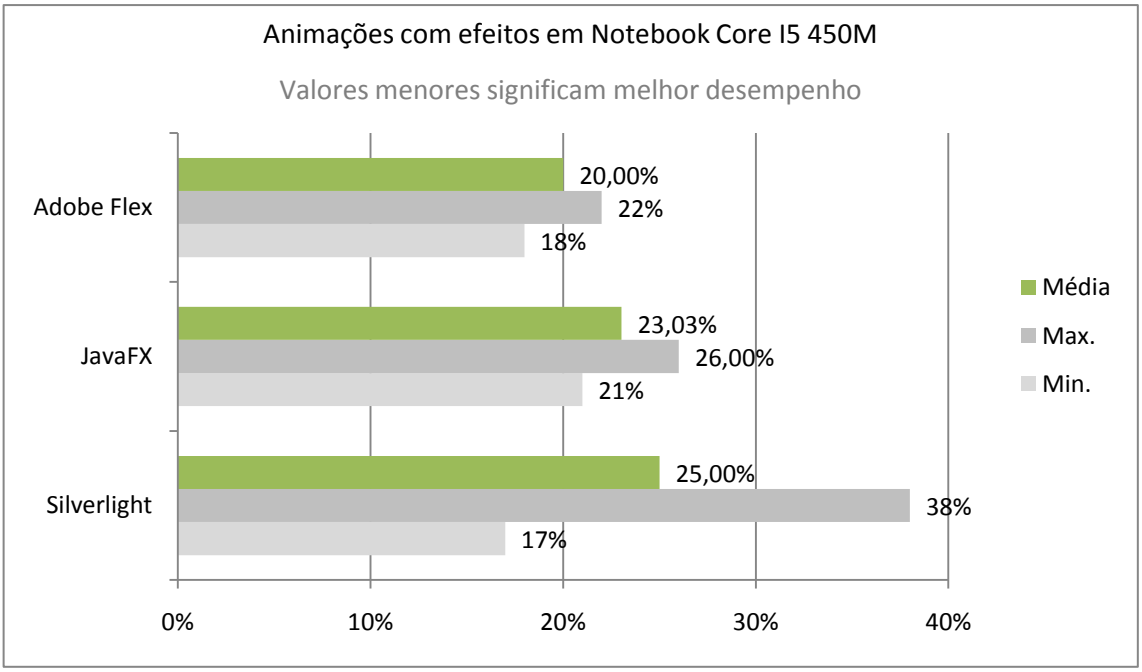

Figura 41. Desempenho do protótipo com efeitos no notebook Core I5 M450

CONCLUSÃO

Neste trabalho, foi realizada a comparação das plataformas Adobe Flex 4.1, Silverlight 4 e JavaFX 1.3, a partir da pesquisa e analise de suas características. Para analise foram desenvolvidos protótipos que experimentaram o uso: das linguagens ActionScript 3, C#, JavaFX Script, MXML, XAML, e CSS; das ferramentas de desenvolvimento Adobe Flash 4, Visual Studio 2010 e Netbeans 6.91; acesso a dados por e meio do SOAP e REST; entre outras características que fazem parte do desenvolvimento de uma aplicação real. Também foi possível avaliar o consumo de processamento por meio dos cenários analisados.

Adobe Flex e Silverlight mostraram-se plataformas maduras, com funcionalidades e ferramentas robustas que tornam fácil e viável o desenvolvimento de aplicativos ricos na internet. O JavaFX mostrou-se promissor, porém não possui ferramentas que facilitam o desenvolvimento e apesar das funcionalidades presentes na linguagem JavaFX Script, a mesma pode afastar os desenvolvedores habituados com o Java.

As plataformas mostraram-se como uma opção poderosa na criação de aplicativos para web. O desenvolvimento similar ao desktop e os recursos oferecidos suprem muitas limitações do DHTML. No entanto as plataformas mostraram-se menos acessíveis devido a necessidade de *plugins*, destacando-se o Silverlight que oferece suporte apenas para Mac e Windows.

Na fase de desenvolvimento dos protótipos teve o desafio de aprender as ferramentas, bibliotecas e características de cada plataforma. O desenvolvimento e aprendizado do JavaFX tornou-se difícil devido a falta de informação disponível no site, muitas vezes com tutoriais superficiais, funcionalidades não documentadas e artigos desatualizados.

Os objetivos específicos e o geral foram alcançados com sucesso e os resultados documentados podem nortear os desenvolvedores na adoção de uma das plataformas para o desenvolvimento web.

Como sugestão para trabalhos futuros e continuidade nessa pesquisa sugere-se:

a) realizar uma comparação entre outras tecnologias RIA como GWT e HTML5;
b) aplicar as plataformas RIA estudadas em novos estudos de casos e com base nos resultados determinar em que situação cada tecnologia é melhor;
c) analisar as especificidades da implementação do HTML5 pelos diversos navegadores.
d) analisar métodos de segurança dos *web services*;

BIBLIOGRAFIA

ADOBE SYSTEMS INCORPORATED. **Adobe: Adobe Flex**. Ano 2010a. Disponível em: <http://www.adobe.com/br/products/flex/>. Acesso em: 26 maio 2010.

ADOBE SYSTEMS INCORPORATED. **About MXML.** 2010c. Disponível em: <http://help.adobe.com/en_US/flex/using/WS2db454920e96a9e51e63e3d11c0bf5f39f-7fff.html>. Acesso em: 26 maio 2011.

ADOBE SYSTEMS INCORPORATED. **ActionScript Technology Center.** 2011. Disponível em: <http://www.adobe.com/devnet/actionscript.html>. Acesso em: 26 maio 2011.

ADOBE SYSTEMS INCORPORATED. **Introducing the Adobe Flash Platform.** 2010b. Disponível em: <http://www.adobe.com/devnet/flashplatform/articles/flashplatform_overview.html>. Acesso em: 10 nov. 2010.

ADOBE SYSTEMS INCORPORATED. **Using Adobe Flex 4.5.** 2011b. Disponível em: <http://help.adobe.com/en_US/flex/using/flex_4.5_help.pdf>. Acesso em: 30 maio 2011.

ALLAIRE, Jeremy. **Macromedia Flash MX - A next-generation rich client.** San Francisco: Macromedia, 2002. Disponível em: <http://www.c2isoft.in/white-papers/richclient.pdf>. Acesso em: 20 maio 2010.

ANDERSON, P. What is Web 2.0? **Ideas, technologies and implications for education**. JISC Technology and Standards Watch. [S.l.], p. 64. 2007.

BASHAN, Brian; SIERRA, Kathy; BATES, Bert. **Use a Cabeça: JSP & Servlets.** Rio de Janeiro: Alta Books, 2005

BOS, Bert et al. **World Wide Web Consortium (W3C):** Cascading Style Sheets Level 2 Revision 1. Disponível em: <http://www.w3.org/TR/CSS2/cover.html>. Acesso em: 02 out. 2010.

CASARIO, M. Flex Solutions: **Essential Techniques for Flex 2 and 3 Developers**. [S.l.]: friends of ED, 2007.

CLEEREN, G.; DOCKX, K. **Silverlight 4 Data and Services Cookbook**. [S.l.]: Packt Publishing, 2010

CORMODE, Graham; KRISHNAMURTHY, Balachander. **Key Differences between Web1.0 and Web2.0.** New Jersey: AT&T Labs–research, 2008

CORPORATION, Oracle. **General Questions.** 2010c. Disponível em: <http://javafx.com/faq/>. Acesso em: 25 maio 2011.

COX, Ken. **ASP.NET 3.5 For Dummies.** Indiana: Wiley Publishing, 2008.

CROCKFORD, Douglas. **JavaScript:** The Good Parts. Sebastopol: Yahoo Press, 2008.

DRIVER, Mark; VALDES, Ray; PHIFER, Gene. **Management Update:** Rich Internet Applications Are the Next Evolution of the Web. Stamford: Gartner, 2005.

EMBARCADERO TECHNOLOGIES. **Delphi Prism XE.** 2011. Disponível em: <http://www.embarcadero.com/products/delphi-prism>. Acesso em: 26 maio 2011.

FAIN, Y.; RASPUTNIS, V.; TARTAKOVSKY, A. **Enterprise Development with Flex**. [S.l.]: O'Reilly Media, 2010.

FLANAGAN, David. **JavaScript:** The Definitive Guide. 4. ed. Sebastopol: O'Reilly & Associates, 2001.

FRATERNALI, P.; ROSSI, G.; SÁNCHEZ-FIGUEROA, F. **Rich Internet Applications**. IEEE Computer, 2010. Disponível em: <http://www.computer.org/portal/web/csdl/abs/html/mags/ic/2010/03/mic2010030009.htm>. Acesso em: 18 Maio 2010.

GOOGLE. **The Chromium Projects**: Chromium OS: Store, edit, and share documents online. 2010a. Disponível em: < http://www.chromium.org/chromium-os>. Acesso em: 21 maio 2010.

GOOGLE. **HTML5ROCKS:** Slides. 2010b. Disponível em: <http://slides.html5rocks.com/#slide1>. Acesso em: 06 nov. 2010.

GORALSKI, G.; LEON, L. **Foundation Flex for Designers**. [S.l.]: friends of ED, 2008.

HALL, Gary Mclean. **Pro WPF and Silverlight MVVM:** Effective Application Development with Model-View-ViewModel. United States Of America: Apress, 2010.

HICKSON, Ian; HYATT, David (Ed.). **World Wide Web Consortium (W3C)**: HTML5. 2010. Disponível em: <http://www.w3.org/TR/html5/>. Acesso em: 20 maio 2010.

JOSH MARINACCI. Oracle Corporation. **Using Perspective Transform for a 3-D Flip Transition.** 2008. Disponível em: <http://javafx.com/samples/PhotoFlip/index.html>. Acesso em: 30 maio 2011.

KONCHADY, Sandeep; REDKO, Alla; BARANOV, Vyacheslav. **JavaFX Media: Platform, Format, and RTSP Support.** 2010. Disponível em: <http://download.oracle.com/javafx/1.3/tutorials/media-platforms-formats/>. Acesso em: 25 maio 2011.

LAIR, R. **Beginning Silverlight 3**. 1. ed. [S.l.]: Apress, 2009.

LECRENSKI, N. **Silverlight 4**: Problem - Design - Solution. [S.l.]: Wrox, 2010.

MACDONALD, M. **Pro Silverlight 3 in C#.** 2. ed. [S.l.]: Apress, 2009.

MICHAIL, A. **Essential Silverlight 3**. 1. ed. [S.l.]: Addison-Wesley Professional, 2009.

MICROSOFT CORPORATION. **.NET Framework Conceptual Overview**. 2010a. Disponível em: <http://msdn.microsoft.com/pt-br/library/zw4w595w.aspx>. Acesso em: 10 nov. 2010.

MICROSOFT CORPORATION. **Silverlight Overview**. 2010d. Disponível em: < http://msdn.microsoft.com/en-us/library/bb404700(VS.95).aspx >. Acesso em: 26 maio 2010.

MICROSOFT CORPORATION. **Silverlight Roadmap**. 2010c. Disponível em: < http://msdn.microsoft.com/en-us/library/bb404700(VS.95).aspx >. Acesso em: 26 maio 2010.

MICROSOFT CORPORATION. **Silverlight Architecture.** 2010b. Disponível em: <http://msdn.microsoft.com/en-us/library/bb404713(v=VS.95).aspx>. Acesso em: 04 nov. 2010.

MICROSOFT. Microsoft: **Office Web Apps**: Store, edit, and share documents online. Disponível em: <http://www.microsoft.com/office/2010/en/office-web-apps/default.aspx>. Acesso em: 21 maio 2010.

MORRIS, Simon. **JavaFX in Action.** Greenwich: Manning Publications, 2009.

NODA, T.; HELWIG, S. **Rich Internet Applications**. University of Wisconsin E-Business Consortium. Madison, p. 10. 2005. (ISSN).

NOVELL. **Moonlight.** 2009. Disponível em: <http://www.go-mono.com/moonlight/>. Acesso em: 25 maio 2011.

ORACLE CORPORATION. **JavaFX**: JavaFX | Rich Internet Applications Development | RIAs Java FX. 2010a. Disponível em: <http://javafx.com/>. Acesso em: 26 maio 2010.

ORACLE CORPORATION. **Develop Expressive Content with the JavaFX Platform.** 2010b. Disponível em: <http://www.javafx.com/about/overview/>. Acesso em: 06 nov. 2010.

O'REILLY, Tim. **What is Web 2.0:** Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <http://oreilly.com/web2/archive/what-is-web-20.html>. Acesso em: 23 set. 2010.

O'ROURKE, C. **A Look at Rich Internet Applications**. Oracle Magazine, 2004.

PISA, F. D. Beginning Java and Flex. [S.l.]: Apress, 2009.

POUCLET, Romain; SULLIVAN, Ed. **How to use the new CSS syntax in Flex 4.** 2010. Disponível em: <http://cookbooks.adobe.com/post_How_to_use_the_new_CSS_syntax_in_Flex_4-15726.html>. Acesso em: 26 maio 2011.

PRECIADO, J. C.; LINAJE, M.; SÁNCHEZ, F. **Necessity of methodologies to model Rich Internet Applications**. Proceedings of the 2005 Seventh IEEE International Symposium on Web Site Evolution, 2005.

RAGGETT, Dave; HORS, Arnaud Le; JACOBS, Ian. **World Wide Web Consortium (W3C):** HTML 4.01 Specification. Disponível em: <http://www.w3.org/TR/html401/>. Acesso em: 01 out. 2010.

SANDERSON, Steven. **Pro ASP.NET MVC 2 Framework.** 2. ed. United States Of America: Apress, 2010.

SCHMITZ, Daniel Pace. **ADOBE FLEX BUILDER 3.0:** CONCEITOS E EXEMPLOS. Rio de Janeiro: Brasport, 2008.

SMITH, Josh. **WPF Apps With The Model-View-ViewModel Design Pattern.** 2009. Disponível em: <http://msdn.microsoft.com/en-us/magazine/dd419663.aspx>. Acesso em: 22 maio 2011.

SRINIVASAN, Latha; TREADWELL, Jem. **An Overview of Service-oriented Architecture, Web Services and Grid Computing.** California: Hewlett-packard Development Company, 2005.

TAIVALSAARI, Antero. Mashware: **The Future of Web Applications**. Tampere: Sun Microsystems Laboratories, 2009.

THE TRUE Story of the Internet. Apresentação John Heileman. USA: Discovery Channel, 2008. 2 DVD.

W3C (Org.). **Simple Object Access Protocol (SOAP) 1.1.** 2000. Disponível em: <http://www.w3.org/TR/2000/NOTE-SOAP-20000508/>. Acesso em: 14 nov. 2010.
W3C (Org.). **Web Services Description Language (WSDL) 1.1.** 2001. Disponível em: < http://www.w3.org/TR/wsdl>. Acesso em: 14 nov. 2010.

WEAVER, J. L. et al. **Pro JavaFX Platform**. [S.l.]: Apress, 2009.

WILLIAMS, Paul. **Presentation Patterns - Presentation Model.** 2007. Disponível em: <http://blogs.adobe.com/paulw/archives/2007/10/presentation_pa_3.html>. Acesso em: 22 maio 2011.

APÊNDICE A – ESTUDO DE CASO

A Versátil Sistemas e Automação é uma empresa que atua no ramo de motéis, cerâmicas e comercio em geral. A empresa que está situada em Criciúma - Santa Catarina está no mercado desde 1998 e possui grande penetração na região sul.

Em 2007 iniciou-se a migração da plataforma Visual Basic 6 para *Microsoft .NET*, um dos novos projetos da empresa foi um módulo de contas a pagar e receber chamado de Under Control.

As duas primeiras versões do Under Control foi construído por meio da API gráfica *Windows Forms*, que é parte do *Microsoft .NET*. Essa API permite a criação de formulários tradicionais para desktop. O banco de dados era instalado em cada maquina.

Para executar o aplicativo era necessária a instalação do *.Net Framework 3.5 SP1* e o do PostgreSQL 8.3, o tamanho das dependências eram em torno de 100mb. Sendo que ainda havia necessidade de suporte técnico para configuração do SGBD. Por ter muitos clientes em outras cidades e alguns em outros estados a implantação do sistema tornou-se difícil.

Diante dos problemas de implantação estudou-se a migração para web, fazendo uso de um cliente mais leve. Nessa nova abordagem o servidor centralizaria os dados, não necessitando de instalação do SGBD para cada computador. Porém o sistema apesar de pequeno possuía funcionalidades e interatividade das quais dificultaria o desenvolvimento do aplicativo em DHTML usando *ASP.NET*, além de que os desenvolvedores estavam habituados em ambiente desktop.

Diante desse cenário optou-se pela utilização do Silverlight. O desafio inicial foi lidar com uma nova linguagem de marcação, o XAML, e conhecer as peculiaridades da API. No entanto, de modo geral a migração ocorreu de forma natural, visto que o desenvolvimento

foi similar ao desktop, inclusive utilizando a mesma linguagem, ferramenta e grande parte das bibliotecas.

Com ele foi possível recriar o sistema de maneira fiel, não apenas mantendo funcionalidades antigas, mas também agregando novas funcionalidades. O sistema sofreu mudanças que melhoraram o leiaute das telas e fez uso dos recursos de animação e capacidade de customização dos componentes. Os clientes já acostumados com o antigo sistema não tiveram dificuldades para se adaptaram com a nova versão.

Com essa nova abordagem foi possível reduzir o tempo e a dificuldade de implantação. Às vezes o cliente tem dificuldade para instalar o *plugin*, mas normalmente é resolvido com orientação por telefone. Outro beneficio foi a expansão do aplicativo para o sistema operacional Mac OS, utilizado por alguns clientes.

APÊNDICE B – ARTIGO

Comparação Entre as Plataformas Adobe Flex, Silverlight e JavaFX para Criação de Aplicativos Ricos na Web

Tiago de Alcântara Esmeraldino[1], Fabrício Giordani[2]

[1]Acadêmico do curso de Ciência da Computação – Unidade Acadêmica de Ciências, Engenharias e Tecnologias - Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brasil

[2]Professor do curso de Ciência da Computação - Unidade Acadêmica de Ciências, Engenharias e Tecnologias - Universidade do Extremo Sul Catarinense (UNESC) – Criciúma, SC – Brasil

{tiago.esm, fgiordani}@gmail.com

Abstract. *This paper presents a comparison between the platforms Adobe Flex 4.1, Silverlight 4, and JavaFX 1.3 for creating web applications. The comparison was performed by the study of the general characteristics of each platform and by creating use cases. It was possible to assess the accessibility, development tools available, supported programming languages, visual controls. It was also possible to experience and evaluate the tools, programming languages, the ease of use web services, resource consumption and other aspects.*

Resumo. *Este artigo apresenta uma comparação entre as plataformas Adobe Flex 4.1, Silverlight 4 e JavaFX 1.3 na criação de aplicativos na internet. A comparação foi realizada por meio do estudo das características gerais de cada plataforma e pela criação de casos de uso, onde foi possível avaliar a acessibilidade, ferramentas disponíveis, linguagens suportadas, controles visuais. Também foi possível experimentar e avaliar as ferramentas, linguagens, a facilidade de consumir web services, o consumo de recursos entre outros aspectos.*

1. Introdução

Inicialmente, a internet foi criada para ser uma plataforma de acesso a textos dinâmicos ou estáticos construídos em HyperText Markup Language (HTML). Nesse modelo original, os usuários são condicionados a interagir com páginas HTML por meio de links e preenchendo dados em formulários. Essa arquitetura simples tornou-se um padrão universal, onde as páginas são visualizadas em qualquer dispositivo que tenha um sistema operacional que possua um navegador básico (LAIR, 2009, tradução nossa).

A web está cada vez mais sendo vista como uma plataforma, e isso reflete hoje em sistemas operacionais e dispositivos voltados à Web . O termo "Web 2.0" é construído com foco no usuário, colocando-o no centro dos aplicativos, como pode ser constatado em redes sociais, *wikis*, vendas virtuais, buscadores entre outros. Nessa nova era de aplicativos, existe a necessidade de um alto grau de interação e usabilidade. A arquitetura HTML tradicional não suporta muitos aspectos exigidos nessa nova era de aplicativos em áreas como apresentação, comunicação e lógica de negócios. Muitas tecnologias estão moldando a web tradicional e acrescentando funcionalidades para corrigir os aspectos em que ela falha. Dentre estas tecnologias, as plataformas RIA se destacam desempenhando um papel fundamental (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010, tradução nossa).

O termo RIA não se refere apenas a uma única tecnologia, mas sim a um conjunto heterogêneo de tecnologias que tem como intuito adicionar novos recursos e funcionalidades

para a web tradicional (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010, tradução nossa).

De acordo com O'Rourke (2004, tradução nossa), tecnologias RIA nos permitem implantar clientes ricos por meio da Internet com a simplicidade de aplicações web tradicionais. Com isso, as aplicações podem ser mais sofisticadas, interativas, melhorando a usabilidade.

2. A Web

A Web consiste em milhares de clientes que interagem por meio de navegadores; servidores executando aplicações; conectados por meio de redes com fio e wireless (BASHAN; SIERRA; BATES, 2005). Ela foi criada inicialmente para ser uma plataforma de acesso a textos dinâmicos ou estáticos construídos em HyperText Markup Language (HTML). Os web sites baseados apenas nessa tecnologia permitem que o usuário leia textos, veja imagens, baixe arquivos e interaja preenchendo formulários e clicando em links.

Devido aos poucos requisitos para navegar em uma página HTML esse modelo tornou-se universal, pois é necessário apenas de um dispositivo que possua um navegador básico, já que não há necessidade de baixar nenhum *plugin* para visualizar as páginas construídas em HTML (FRATERNALI; ROSSI; SÁNCHEZ-FIGUEROA, 2010, tradução nossa).

A web possui uma arquitetura simples, com clientes e servidores. Toda comunicação é feita por meio do protocolo HyperText Transfer Protocol (HTTP), sendo que o conteúdo das requisições e respostas são encapsuladas dentro dessas mensagens. Ele depende de outros protocolos que são responsáveis por enviar com integridade as mensagens para o destino, mesmo que elas estejam divididas em blocos (BASHAN; SIERRA; BATES, 2005).

A web vem sofrendo várias evoluções desde a década de 90. Nesse processo a mesma alterou vários paradigmas dos sistemas clássicos como: software como serviço, onde não há licenças ou vendas, apenas o uso do usuário; não há prazos para lançamento do sistema, apenas contínuos aperfeiçoamentos; sem conversões para diferentes plataformas (O'REILLY, 2005, tradução nossa).

O conteúdo da web sofreu evoluções no decorrem dos anos e é possível especificar três fases: na primeira fase as páginas eram simples documentos de textos com algumas imagens intercaladas, a navegação era baseada simplesmente em hiperlinks; gradualmente com a evolução gradual do DHTML tornou-se possível a criação de páginas web cada vez mais interativas com gráficos animados e *plugins* que permitiram um conteúdo mais dinâmico e rico para ser apresentado; e a ultima fase é caracterizada pelas aplicações ricas, que são construídas com tecnologias que tendem combinar a colaboração e interatividade, permitindo que os aplicativos web tenham uma interatividade elevada, remetendo às aplicações desktop (TAIVALSAARI, 2009, tradução nossa).

3 Rich Internet Application

As interfaces das aplicações desktop evoluíram conforme o tempo e tornaram-se mais ricas e interativas proporcionando uma melhor experiência para o usuário. No entanto, um desafio de uma aplicação desktop é a implantação, pois geralmente é preciso ter a base de código para cada plataforma. Além disso, cada atualização e manutenção devem ser realizadas localmente na máquina (LAIR, 2009, tradução nossa).

Essas desvantagens dos aplicativos desktop são facilmente resolvidas em aplicações web, pois a implantação é simples necessitando apenas de um dispositivo que tenha um navegador, além disso, atualizações e manutenções são realizadas apenas no servidor diminuindo a complexidade.

O termo Rich Internet Application (RIA) foi criado pela Macromedia quando ela deu inicio a uma nova abordagem para o Flash. O mesmo poderia fazer muito mais que apenas animações, fornecendo um cliente rico para execução de aplicações na internet. Nessa nova abordagem uniu-se o melhor dos dois mundos, como ilustrado na Figura 1. (O'REILLY, 2005, tradução nossa).

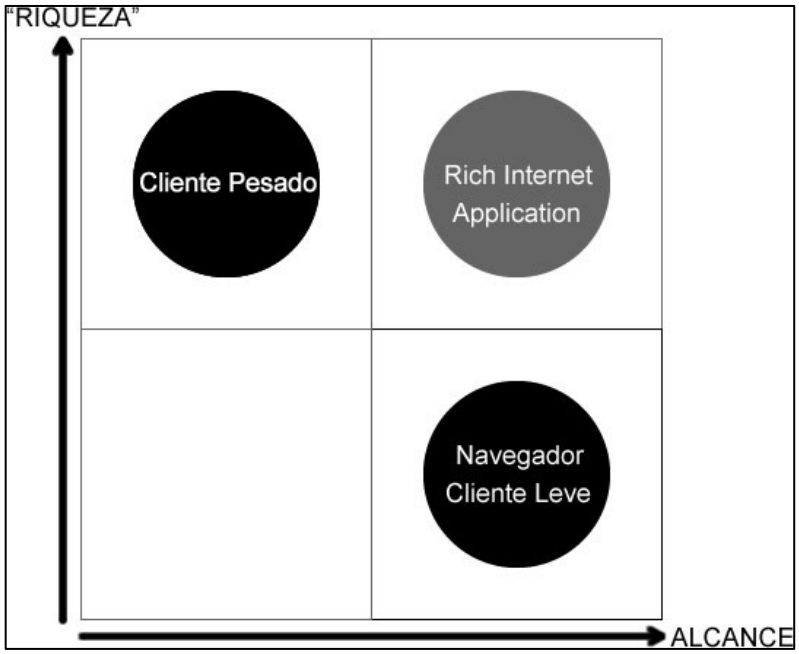

Figura 1. A abordagem das aplicações ricas
Fonte: Adaptado de DRIVER, M; VALDES, R; PHIFER, G (2005)

Existem algumas categorias de tecnologias RIA, sendo que Adobe Flash/Flex, Silverlight e JavaFX se encontram na mesma categoria, todas utilizam um cliente rico ou seja, são baseadas em *plugins* e com possibilidade de execução no desktop. Enquanto que as tecnologias baseadas no navegador acabam tendo um alcance maior, mas perdendo na riqueza da aplicação (TAIVALSAARI, 2009, tradução nossa).

A Macromedia em 2002 estabeleceu que um cliente rico (ou *plugin*) devesse: proporcionar eficiência e alto desempenho para execução do código, conteúdo e comunicação, tendo em vista o a experiência do usuário final, contornando as limitações de um aplicativo web tradicional como a falta de armazenamento no lado do cliente, a incapacidade de invocar e usar lógica de negócios remota com facilidade e até mesmo o visual do HTML; fornecer modelos de objetos poderosos e extensíveis para interatividade ligados com a interface do usuário, comunicação e serviços do sistema; permitir o desenvolvimento rápido de aplicações por meio de componentes e reutilização, facilitando a criação de bibliotecas de terceiros e a reutilização de componentes visuais para acelerar o desenvolvimento; proporcionar o uso da web e serviços de dados fornecidos pelos servidores de aplicação; possibilitar fácil publicação em múltiplas plataformas e dispositivos, possuindo um certo alcance que é uma característica da web.

4 Adobe Flex

O Adobe Flex é um framework integrante da plataforma Flash que facilita o desenvolvimento de aplicações ricas na internet, contendo diversos tipos de componentes visuais e uma poderosa linguagem de programação chamada ActionScript. Ele utiliza o *runtime* do Flash Player, sendo assim ele é baseado em suas funcionalidades que cuidam da lógica do lado do cliente e que também permite interagir com o conteúdo HTML e JavaScript do navegador. O

Flash Player possui uma alta taxa de adoção e é de natureza multi-plataforma que permite poupar muito tempo que seria gasto em testes e depuração em diferentes versões do navegador e sistemas operacionais diferentes (SCHMITZ, 2008).

Os aplicativos Flex são compostos de arquivos MXML e ActionScript. O MXML é uma linguagem de marcação que se baseia no padrão Extensible Markup Language (XML), ela é utilizada para definir a interface do usuário por meio de tags que correspondem a elementos visuais e aos aspectos não visuais da aplicação (como data-binding e recursos do lado servidor). Cada tag é baseada em classes criadas em ActionScript e tudo é compilado em um arquivo SWF. O ActionScript é uma linguagem de programação baseada no padrão ECMA Script 262, ela permite que o desenvolvedor programe o comportamento da aplicação e como o usuário pode interagir com os elementos da interface (CASARIO, 2007, tradução nossa).

5 Silverlight

O Microsoft Silverlight (ou apenas Silverlight) é uma implementação do .Net Framework para criação e entrega de aplicativos ricos na internet em múltiplos navegadores e sistemas operacionais. O Silverlight utiliza uma versão compacta do .NET CLR (runtime da Plataforma .Net) que permite escrever o código em linguagens suportadas pela plataforma .NET como C# ou Visual Basic, além de linguagens dinâmicas (MICROSOFT CORPORATION, 2010a, tradução nossa).

Para definição da interface é utilizada a linguagem Extensible Application Markup Language (XAML) que tem sua base em XML. Com XAML é possível definir elementos visuais como controles ou formas além de animações, transformações e eventos. O código em linguagem de programação gerenciado ou dinâmico define o comportamento do aplicativo (MICROSOFT CORPORATION, 2010b, tradução nossa).

6 JavaFX

O JavaFX é uma plataforma para criação e distribuição de aplicativos ricos na internet, no desktop, televisões e outros dispositivos. JavaFX utiliza o *Java Runtime Environment* (JRE), que é o mesmo *runtime* da plataforma Java. Sendo assim a aplicação é executada dentro do navegador pela *Java Virtual Machine* (JVM) (ORACLE CORPORATION, 2010, tradução nossa).

Para o desenvolvimento de aplicações, o JavaFX utiliza uma nova linguagem de programação chamada *JavaFX Script*. Essa linguagem foi construída do zero para modelagem e animação de aplicações multimídia. Ela é uma linguagem compilada e orientada a objetos, com uma sintaxe independente do Java tradicional, porém é capaz de trabalhar com classes Java (MORRIS, 2009, tradução nossa).

7 Comparação Entre as Plataformas

Para a realização da comparação foi estudado as características gerais de cada plataforma. Além do estudo foram criados cenários a fim de testar: o uso das ferramentas de codificação e design; linguagens de programação; facilidade de manipular dados na web; utilização de um padrão de arquitetura; a capacidade de alterar o visual dos elementos gráficos; animações e efeitos visuais. Foram escolhidas para realizar a avaliação as seguintes versões estáveis de cada plataforma: Adobe Flex 4.1, Silverlight 4 e JavaFX 1.3

7.1. Acessibilide das Plataformas

A web se tornou extremamente acessível por meio dos navegadores, bastando o mesmo exibir o resultado do HTML. Mas existem algumas barreiras para as plataforma RIA devido a

dependência dos plugins. A Tabela 1 apresenta alguns fatores que podem dificultar a acessibilidade da aplicação como: os navegadores suportados, sistemas operacionais, o tamanho e a dificuldade de instalação de seus *plugins*, e a taxa de penetração dos plugins.

Tabela 1. Acessibilide

	Adobe Flex	Silverlight	JavaFX
Navegadores Suportados Oficialmente			
Internet Explorer	Sim	Sim	Sim
Firefox	Sim	Sim	Sim
Chrome	Sim	Sim	Parcial
Safari	Sim	Sim	Sim
Opera	Sim	Não	Parcial
AOL	Sim	Não	Parcial
Sistemas Operacionais Suportados Oficialmente			
Windows	Sim	Sim	Sim
Mac OS X	Sim	Sim	Sim
Linux	Sim	Com Moonlight	Sim
Solaris	Sim	Não	Sim
Tamanho dos Plugins			
Nome/Versão	Flash Player 10.3	Silverlight 4	JR 1.6u24 x86
Tamanho	~3,22MB	~6MB	15,77MB
Base Instalada/Taxa de Penetração dos Plugins			
Stat OWL	~96.18%	~71.62%	~65,57%
RIA Stats	~95,86%	~61.42%	~78.13%
Média	~96,02%	~66,52%	~71.85%
Dificuldade de Instalação dos Plugins			
Passos do Usuário	3	6	6
Mostra o conteúdo no final da instalação	Sim	Não	Sim
Mensagem em inglês	Não	Não	Sim

O JavaFX suporta todos navegadores, porém a API JavaFX Media responsável pela execução de vídeo e áudio possuem implementações diferentes para cada sistema operacional, devido a essa característica é oferecido suporte oficial nos navegadores Internet Explorer, Firefox e Safari (KONCHADY; REDKO; BARANOV, 2011, tradução nossa).

O Silverlight tem foco nos sistemas operacionais Windows e Mac OS, no entanto existe o Moonlight que é uma implementação de código aberto para os sistemas operacionais baseados em Unix criado pela Novell com colaboração da Microsoft (NOVELL, 2009, tradução nossa).

Para ter uma base da taxa de adoção foram utilizados como fonte os sites Stat OWL e RIA Stats que fornecerem estatísticas de uso e de penetração das plataformas RIA. Apesar do JRE mostrar uma boa taxa de adoção apenas a versão JRE 1.6 oferece suporte ao JavaFX. Além do Flash Player ter uma alta taxa de adoção, o plugin já vem embutido no navegador Google Chrome.

Para mensurar a dificuldade de instalação foram realizados testes em uma maquina virtual com o Windows XP SP2 32bits com idioma em português do Brasil e com o navegador padrão Internet Explorer 6. O teste consistiu em acessar um conteúdo criado em cada plataforma. As páginas que hospedam o conteúdo são as páginas padrões gerado por cada ferramenta, sem qualquer customização de imagens e instruções de download do plugin.

7.2 Ferramentas Disponíveis

A Tabela 2 apresenta as ferramentas disponíveis para o desenvolvimento de cada plataforma e o custo unitário. O Adobe Flex e Silverlight possuem ferramentas especializadas em design que facilitam a criação de interfaces mais sofisticadas. E o JavaFX e Silverlight possuem ferramentas gratuitas para o desenvolvimento de aplicações.

Tabela 2. Ferramentas de Desenvolvimento

Nome	Valor	Plataforma	Design
Flash Builder 4.5 Standard Edition	$289,00	Adobe Flex	Não
Flash Builder 4.5 Premium Edition	$805,00	Adobe Flex	Não
Flash Catalyst CS5.5	$486,00	Adobe Flex	Sim
Visual Web Developer 2010	Gratuita	Silverlight	Não
Visual Studio 2010 Professional	$799,00	Silverlight	Não
Expression Studio 4 Ultimate	$599,00	Silverlight	Sim
Netbeans	Gratuita	JavaFX	Não

7.3 Linguagens Suportadas

Aprender uma linguagem totalmente nova pode ser um empecilho para os desenvolvedores já habituados com outra linguagem. Esse aspecto pode aumentar o custo e o tempo de desenvolvimento de um aplicativo.

Como apresentado na Tabela 3 o Silverlight se destaca por suportar as linguagens da plataforma *Microsoft .NET* para criação da lógica do aplicativo. O Adoble Flex e JavaFX utilizam a linguagem CSS para definição do estilo da aplicação, a mesma linguagem utilizada para formatação do conteúdo em HTML.

Tabela 3. Linguagens suportadas

Categoria	Adobe Flex	Silverlight	JavaFX
Lógica	ActionScript	C#/VB.Net/IronPython/IronRuby/M.JScript	JavaFX Script/Java
Definição	MXML	XAML	JavaFX Script
Estilo	CSS	XAML	CSS

7.4 Controles Visuais na Biblioteca Inicial

No HTML todos os controles visuais são pré-definidos, e apesar da junção de HTML, CSS e JavaScript conseguir trazer mais riqueza visual, ainda possui limitações. Em contrapartida nas plataformas RIA é possível criar os próprios controles visuais a partir de um controle base.

No entanto cada plataforma possui uma biblioteca inicial de controles visuais, sendo que há uma proximidade dos controles que são comuns em aplicativos desktop. A biblioteca inicial evita que o desenvolvedor gaste tempo para construir componentes já existentes no desktop e evita também a compra de componentes de terceiros.

De modo geral todos possuem os controles básicos já conhecidos em aplicações desktop. O JavaFX possui poucos controles além do básico sendo que a ausência de maior relevância é do *Datagrid* (também conhecido como tabela ou *table*) uma solução é utilizar componentes do *Swing* como o *JTable*, porém essa solução causa diferenças visuais entre os componentes e gera dependência que pode limitar o número de plataformas onde o aplicativo será executado (MORRIS, 2009, tradução nossa).

7.5 Protótipo de Aplicação

O protótipo de aplicação consiste no desenvolvimento de uma agenda nas plataformas Adobe Flex, Silverlight e JavaFX. Esse protótipo tem como intuito avaliar as ferramentas de

desenvolvimento; linguagens de programação; manipulação de dados na web; a capacidade de alterar o estilo visual dos componentes visuais, entre outros aspectos.

A fim de deixar as aplicações parecidas em cada plataforma, buscou-se não utilizar recursos ou componentes visuais que não estivessem presentes em todas as plataformas e também se utilizou o padrão PM para o código ter as mesmas características de implementação, além de ser o padrão recomendado pela Adobe e Microsoft. Também foi criado um único servidor em ASP.NET 3 com *web services* SOAP e RESTful.

As ferramentas de desenvolvimento utilizadas nesse protótipo foram: Microsoft Visual Web Developer 2010 Express SP1, versão gratuita do Visual Studio 2010, para criação do aplicativo no servidor em ASP e do cliente em Silverlight 4; Adobe Flash Builder 4 para criação do cliente em Adobe Flex 4.1; e na construção do cliente em JavaFX 1.3 utilizou-se o Netbeans 6.9.1.

A Tabela 4 apresenta resultados obtidos com a construção do protótipo da agenda.

Tabela 4. Aspectos avaliados com o protótipo

Categoria	Adobe Flex	Silverlight	JavaFX
Ferramenta	Flash Builder 4	Visual Web Developer 2010	Netbeans 6.9.1
Integração com Servidor na Criação do Projeto	ASP.NET/ColdFusion/J2EE/PHP	ASP.NET	Não
Geração de código automático pela WSDL	Sim	Sim	Não
Acesso aos Dados	Fácil	Fácil	Difícil
Criação de Telas	Fácil	Fácil	Difícil

O Visual Studio na criação do projeto do Silverlight inicialmente proporciona a escolha de linguagem, podendo optar por C# ou VB.Net, e modelo de projeto. Escolheu-se a linguagem C# e o modelo *Silverlight Application* que é um modelo limpo com apenas uma página principal. Logo após essa etapa, é possível optar por versões antigas do Silverlight e em criar um projeto web em ASP para servir de servidor da aplicação possuindo uma integração natural entre as plataformas da *Microsoft .Net.* Porém optou-se por não criar, pois o servidor está separado do projeto.

A ferramenta Flash Builder na criação de um projeto Adobe Flex, proporciona mais escolhas referente aos tipos de servidor, possuindo compatibilidade com *ASP.NET*, ColdFusion, J2EE e PHP. Também é necessário escolher se o aplicativo irá executar dentro do navegador através do Flash Player ou desktop com Adobe AIR. Inicialmente escolheu-se a opção do tipo de servidor Nenhum/Outro (*None/Other*) e também optou-se pela execução no navegador com Flash Player.

Na criação do projeto JavaFX no Netbeans é apenas necessário escolher o modelo de projeto, sendo que o escolheu-se o modelo *Aplicativo JavaFX Script*. Por fim todos possuem opções básicas de nomear o projeto e definir o local.

Todas as ferramentas possuem editor visual básicos possibilitando arrastar componentes e alterar suas propriedades.

No entanto se utilizar o Netbeans dessa forma o código gerado não poderá ser alterado dificultando uma alteração rápida e o uso do recurso de *binding* oferecida pela linguagem JavaFX Script. Outro ponto constatado é que em livros ou tutorias oferecidos pela Oracle não são utilizados o editor visual do Netbeans. Porem para facilitar a IDE disponibiliza a paleta de componentes dentro do código. Outro recurso é a pré-visualização da tela, no entanto esse recurso só funciona com a tela principal do programa, tonando-se inútil em um programa com mais janelas ou na construção de um novo componente visual.

A Figura 2 apresenta as telas de listagem dos protótipos construídos em cada plataforma.

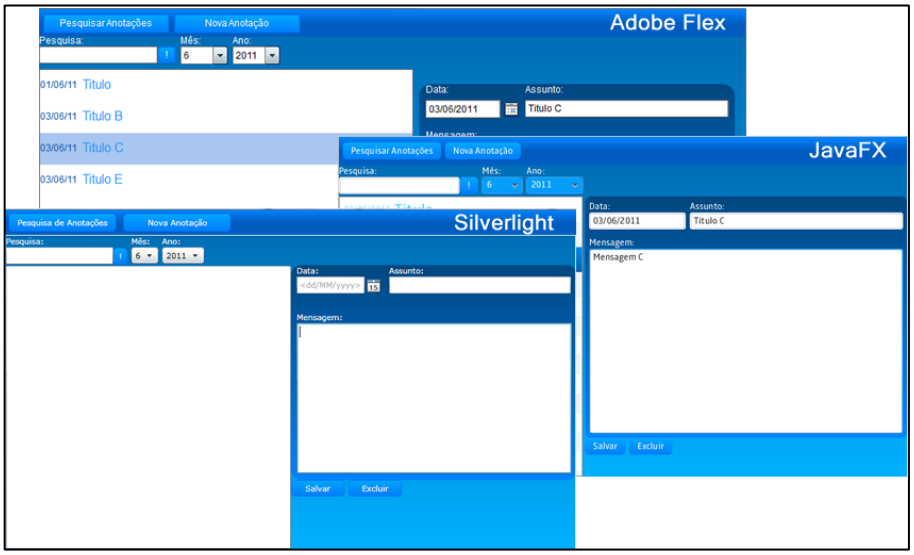

Figura 2. Telas de listagem de anotações dos protótipos.

7.6 Comparação do Desempenho

As plataformas proporcionam à aplicação um visual rico por meio das animações, efeitos entre outros recursos. Sendo que tudo é feito por meio de um cliente leve que proporciona certa acessibilidade da aplicação. No entanto uma aplicação pode ser limitada a execução em computadores atuais dependendo do consumo de processamento do aplicativo.

A fim de avaliar o consumo de processamento desenvolveram-se os seguintes protótipos: protótipo com animações de translação de dez objetos, rotação de dez objetos e mudança de escala de dez objetos; protótipo com animações de translação de dez objetos, rotação de dez objetos, mudança de escala de dez objetos todos com efeito de sombra aplicado.

Os protótipos foram executados nos seguintes computadores: notebook com processador modelo Intel I5 450M, freqüência 2.4GHZ, dual core com quatro threads, VGA dedicada e sistema operacional Windows 7 64bits; notebook com processador modelo Intel Celeron M410, freqüência 1.46GHZ, um core e sistema operacional Windows XP SP2 32bits.

A Figura 3 apresenta a média do consumo de processamentos dos protótipos construídos em Adobe Flex, Silverlight e JavaFX. O Silverlight ganhou destaque consumindo pouco processamento nos cenários onde foi executado o protótipo sem sombras. No entanto quando executado o protótipo com sombras o desempenho caiu drasticamente em todas as plataformas.

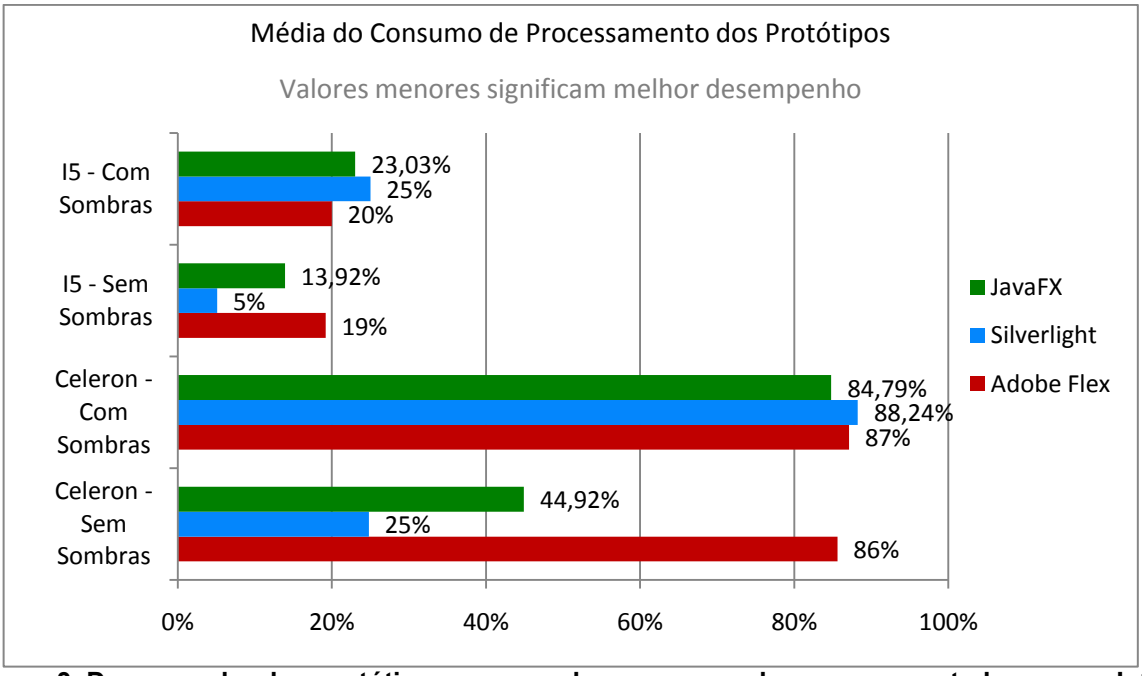

Figura 3. Desempenho dos protótipos com sombra e sem sombra nos computadores com Intel I5 e Intel Celeron.

8. Conclusão

Neste trabalho, foi realizada a comparação das plataformas Adobe Flex 4.1, Silverlight 4 e JavaFX 1.3, a partir da pesquisa e analise de suas características. Para analise foram desenvolvidos protótipos que experimentaram o uso: das linguagens ActionScript 3, C#, JavaFX Script, MXML, XAML, e CSS; das ferramentas de desenvolvimento Adobe Flash 4, Visual Studio 2010 e Netbeans 6.91; acesso a dados por e meio do SOAP e REST; entre outras características que fazem parte do desenvolvimento de uma aplicação real. Também foi possível avaliar o consumo de processamento por meio dos cenários analisados.

Adobe Flex e Silverlight mostraram-se plataformas maduras, com funcionalidades e ferramentas robustas que tornam fácil e viável o desenvolvimento de aplicativos ricos na internet. O JavaFX mostrou-se promissor, porém não possui ferramentas que facilitam o desenvolvimento e apesar das funcionalidades presentes na linguagem JavaFX Script, a mesma pode afastar os desenvolvedores habituados com o Java.

As plataformas mostraram-se como uma opção poderosa na criação de aplicativos para web. O desenvolvimento similar ao desktop e os recursos oferecidos suprem muitas limitações do DHTML.

Como sugestão para trabalhos futuros e continuidade nessa pesquisa sugere-se:

a) realizar uma comparação entre outras tecnologias RIA como GWT e HTML5;
b) aplicar as plataformas RIA estudadas em novos estudos de caso e com base nos resultados determinar em que situação cada tecnologia é melhor;
c) analisar as especificidades da implementação do HTML5 pelos diversos navegadores.
d) analisar métodos de segurança dos *web servies*;

Referências

BASHAN, Brian; SIERRA, Kathy; BATES, Bert. **Use a Cabeça: JSP & Servlets.** Rio de Janeiro: Alta Books, 2005

CASARIO, M. Flex Solutions: **Essential Techniques for Flex 2 and 3 Developers**. [S.l.]: friends of ED, 2007.

FRATERNALI, P.; ROSSI, G.; SÁNCHEZ-FIGUEROA, F. **Rich Internet Applications**. IEEE Computer, 2010. Disponível em: <http://www.computer.org/portal/web/csdl/abs/html/mags/ic/2010/03/mic2010030009.htm >. Acesso em: 18 Maio 2010.

KONCHADY, Sandeep; REDKO, Alla; BARANOV, Vyacheslav. **JavaFX Media: Platform, Format, and RTSP Support.** 2010. Disponível em: <http://download.oracle.com/javafx/1.3/tutorials/media-platforms-formats/>. Acesso em: 25 maio 2011.

LAIR, R. **Beginning Silverlight 3**. 1. ed. [S.l.]: Apress, 2009.

MICROSOFT CORPORATION. **Silverlight Overview**. 2010a. Disponível em: < http://msdn.microsoft.com/en-us/library/bb404700(VS.95).aspx >. Acesso em: 26 maio 2010.

MICROSOFT CORPORATION. **Silverlight Roadmap**. 2010b. Disponível em: < http://msdn.microsoft.com/en-us/library/bb404700(VS.95).aspx >. Acesso em: 26 maio 2010.

MORRIS, Simon. **JavaFX in Action.** Greenwich: Manning Publications, 2009.

NOVELL. **Moonlight.** 2009. Disponível em: <http://www.go-mono.com/moonlight/>. Acesso em: 25 maio 2011.

ORACLE CORPORATION. **Develop Expressive Content with the JavaFX Platform.** 2010b. Disponível em: <http://www.javafx.com/about/overview/>. Acesso em: 06 nov. 2010.

O'REILLY, Tim. **What is Web 2.0:** Design Patterns and Business Models for the Next Generation of Software. 2005. Disponível em: <http://oreilly.com/web2/archive/what-is-web-20.html>. Acesso em: 23 set. 2010.

O'ROURKE, C. **A Look at Rich Internet Applications**. Oracle Magazine, 2004.

SCHMITZ, Daniel Pace. **ADOBE FLEX BUILDER 3.0:** CONCEITOS E EXEMPLOS. Rio de Janeiro: Brasport, 2008.

TAIVALSAARI, Antero. Mashware: **The Future of Web Applications**. Tampere: Sun Microsystems Laboratories, 2009.