UNIVERSIDADE FUMEC
FACULDADE DE CIÊNCIAS EMPRESARIAIS

JACKSON DOS SANTOS LOPES
RODRIGO OLIVEIRA TAVARES

JAVAFX:
Uma abordagem ao desenvolvimento de aplicações RIA

BELO HORIZONTE

2010

JACKSON DOS SANTOS LOPES
RODRIGO OLIVEIRA TAVARES

JAVAFX:
Uma abordagem ao desenvolvimento de aplicações RIA

Monografia realizada na Universidade FUMEC, no curso de Ciência da Computação, apresentado à disciplina Trabalho de Conclusão de Curso.

Orientadores:
Professor Osvaldo Manoel Corrêa
Professor Ricardo Terra

BELO HORIZONTE

2010

JACKSON DOS SANTOS LOPES
RODRIGO OLIVEIRA TAVARES

JAVAFX:
Uma abordagem ao desenvolvimento de aplicações RIA

Monografia de conclusão de curso submetida à Universidade FUMEC como requisito parcial para obtenção do título de Bacharel em Ciência da Computação e aprovada pela seguinte banca examinadora:

_____
Professor Osvaldo Manoel Corrêa (Professor TCC)
Universidade FUMEC

_____
Professor Ricardo Terra (Orientador)
Universidade FUMEC

Data da aprovação:

BELO HORIZONTE

2010

Dedicamos esta monografia a pessoas especiais que é nossa base de sustentação: nossas famílias.

AGRADECIMENTOS

Aos orientadores Prof. Osvaldo Manoel Costa e Prof. Ricardo Terra Nunes Bueno Villela pelo acompanhamento, sugestões e discussões.

A nossas famílias pelo apoio e encorajamento, aos nossos amigos pela cumplicidade e amizade e a todos que direta e indiretamente contribuíram para a realização deste trabalho.

As coisas simples devem ser simples e as complexas, possíveis. (ALAN KAY)

RESUMO

Com as exigências de um mercado globalizado e altamente competitivo, surgiu a necessidade de tecnologias inovadoras que propusessem a unir os conceitos de *web* e *desktop* em um mesmo ambiente, tornando assim mais simples e menos oneroso a criação de aplicativos interativos e portáveis. Contudo, isso só foi possível após a criação do conceito de RIA (*Rich Internet Applications*), que permitiu que tecnologias como JavaFX surgissem e proporcionassem experiências mais aprazíveis para os usuários.

Com as aplicações RIA ganhando notoriedade e pela união de vários conceitos antes tratados como exclusivos, esta monografia visa apresentar a plataforma JavaFx no desenvolvimento de aplicações ricas para Internet. Por meio de análises e comparativos com tecnologias similares, é apresentada a essa tecnologia no desenvolvimento de sistemas.

PALAVRAS CHAVES: JavaFX; Java; RIA; interface gráfica

LISTA DE FIGURAS

LISTA DE SIGLAS

AJAX – *Asynchronous JavaScript And XML*

API – *Application Programming Interface*

AWT – *Abstract Window Toolkit*

CSS – *Cascading Style Sheets*

FLV – *Flash Video*

GPU – *Graphics Processing Unit*

HTML – *Hypertext Markup Language*

HTTP – *Hypertext Transfer Protocol*

IDE – *Integrated Development Environment*

JAR – *Java Archive*

JIT – *Just In time*

JNLP – *Java Network Launching Protocol*

JPEG – *Joint Photographic Experts Group*

JRE – *Java Runtime Environment*

JSON – *JavaScript Object Notation*

JVM – *Java Virtual Machine*

JWS – *Java Web Start*

PNG – *Portable Network Graphics*

RIA – *Rich Internet Application*

RSS – *Really Simple Syndication*

SOAP – *Simple Object Access Protocol*

XAML – *Extensible Application Markup Language*

XML – *Extensible Markup Language*

WMV – *Windows Media Video*

SUMÁRIO

INTRODUÇÃO

Em meados da década de 90 com a popularização da Internet, um novo paradigma de comunicação surgiu, afastando-se do princípio da unidirecionalidade para a comunicação ágil e bidirecional. As tecnologias iniciais utilizadas na Internet permitiram a agilidade na apresentação da informação, mas seu caráter estático impossibilitou o desenvolvimento de serviços com interação e usabilidade. Conforme Jensen (1998, p. 185–204), a interatividade é ”uma medida do potencial de habilidade de uma mídia permitir que o usuário exerça influência sobre o conteúdo ou a forma da comunicação mediada”.

Além disso, a confluência de acontecimentos do mercado tecnológico, de acordo com Peters (2010), evidenciou que as empresas necessitavam de um novo modelo de negócios fundamentado na Internet capaz de atrair o usuário e prover melhores formas de interatividade e usabilidade. Esse modelo deve suprir as atuais limitações dos padrões tecnológicos, onde o desenvolvedor é obrigado a trabalhar em um contexto limitado e sem grandes recursos, e onde a falta de integração e comunicação entre equipes de desenvolvimento e *design* tem um forte impacto na qualidade e na produtividade do software desenvolvido.

Outros problemas eminentes no mercado de software são os altos custos da mão de obra especializada e o tempo de desenvolvimento, que se tornam cada dia mais oneroso. Com a variedade de sistemas operacionais existentes e suas peculiaridades, torna-se extremamente difícil a elaboração de sistemas que utilizem os mesmos recursos, tais como, *Application Programming Interface (API)*, bibliotecas, interfaces, entre outros. Além disso, o fato de não usar um padrão impossibilita o uso das melhores práticas recomendadas para cada ambiente operacional.

Do mesmo modo, outro grande problema contemporâneo é que nem todos navegadores de Internet seguem a rigor as padronizações da *World Wide Web Consortium* (W3C), comunidade internacional fundada em 1994 que promove e regulamenta os padrões utilizados na Internet.

Conforme Elis (2005), ao invés disso, os responsáveis pelo desenvolvimento dos navegadores de Internet, tendem a desenvolver seus próprios recursos, com o intuito de se diferenciar da concorrência e assim obter maior

notoriedade no mercado. Com isso, torna-se muito complexo desenvolver aplicações que tenham as mesmas funcionalidades e aparência em diversos navegadores, bem como em diferentes sistemas operacionais.

A rápida ascensão da Internet possibilitou um processo de reformulação e modernização no que se refere ao desenvolvimento de novas tecnologias. Impulsionados por usuários cada vez mais exigentes, cientistas da computação, estudiosos do assunto e grandes corporações sentiram que existia a necessidade de criar novos recursos mais interativos, a fim de estabelecer uma melhor relação entre homem e máquina. De acordo com a Associação Brasileira de Normas Técnicas (ABNT), o conceito de usabilidade é descrito como “medida na qual um produto pode ser usado por usuários específicos para alcançar objetivos específicos com eficácia, eficiência e satisfação em um contexto específico de uso.” (ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS, 2002, p.3).

Com o amplo mercado a ser explorado, tecnologias foram desenvolvidas a fim de permitir que o conteúdo outrora estático, fosse aperfeiçoado pela concepção de dinamismo, ou seja, poderia ser gerado conteúdo sob-demanda ou mesmo personalizado aos usuários. Durante muitos anos, a apresentação do conteúdo dinâmico se tornou quase um padrão no desenvolvimento de *sites* para *web* e atraiu grandes empresas. No entanto, a contínua evolução das tecnologias *web* propiciou a criação de novos recursos e paradigmas de desenvolvimento, levando ao usuário a junção de tecnologias *web* com *desktop*. Tal artifício foi denominado *Rich Internet Application (RIA)* traduzido por Aplicações ricas para Internet. Vistos por muitos como a revolução na *web*, grandes empresas tem dedicado estudo e orçamento no desenvolvimento e proliferação dessa nova tendência, que leva a níveis nunca antes visto de interatividade e usabilidade.

Seguindo essa tendência, pode-se mencionar o JavaFx, que conforme a Sun (2010), foi lançado no final de 2008 e tem como base de sustentação a plataforma Java. Utilizando-se do conceito de máquina virtual, o JavaFX herda de Java princípios de portabilidade e interoperabilidade entre os sistemas operacionais, beneficiando-se assim da maturidade, desempenho e ubiquidade da plataforma Java. Tais características são importantes no atual cenário de desenvolvimento, pois permite que aplicações sejam escritas em uma linguagem comum e executem em vários ambientes operacionais, bem como, em diversos navegadores de Internet, reduzindo assim o tempo e custos gastos em seu desenvolvimento.

Com as aplicações RIA ganhando notoriedade e pela união de vários conceitos antes tratados como exclusivos, esta monografia visa apresentar a plataforma JavaFx no desenvolvimento de aplicações ricas para Internet. Será apresentado os conceitos e características técnicas da plataforma JavaFX e demonstrado os resultados obtidos durante o desenvolvimento de um exemplo prático utilizando a tecnologia. Além disso, o trabalho apresenta as plataformas Adobe AIR e Silverlight e realiza um comparativo das tecnologias com o JavaFX. Para sustentar teoricamente essa monografia, serão citados os autores James Weaver, Jonathan Giles, Terrence Bar, Doris Chen e as empresas Sun Microsystems, Microsoft e Adobe.

No primeiro capítulo desse trabalho é apresentado a história e conceitos básicos da plataforma JavaFX.

No segundo capítulo é exposto os componentes que são a base arquitetural do JavaFX e as características que a compõe. Além disso, são abordados os detalhes específicos do JavaFX para os ambientes *desktop* e *web*. Na última seção do capítulo é apresentado um exemplo prático que foi desenvolvido utilizando a tecnologia e os resultados obtidos.

No terceiro e último capítulo são apresentadas as características das tecnologias Adobe AIR e Silverlight e o comparativo dessas plataformas RIA com o JavaFX.

1. INTRODUÇÃO A PLATAFORMA JAVAFX

Com as exigências de um mercado globalizado e altamente competitivo, surgiu a necessidade de tecnologias inovadoras que propusessem a unir os conceitos *web* e *desktop* em um mesmo ambiente, tornando assim mais simples e menos oneroso a criação de aplicativos interativos e portáveis. Contudo, isso só foi possível após a criação do conceito RIA, que permitiu que tecnologias como JavaFX surgissem e proporcionassem experiências mais aprazíveis para os usuários. Com isso, aplicações RIA agregam melhor usabilidade, interatividade e interfaces mais intuitivas, produzindo assim um software mais rico em recursos e mais satisfatório para o usuário.

A plataforma JavaFX une características interessantes como a aproximação entre desenvolvedores e *designers*, conforme descrito pela Sun Microsystems (2007):

> Os desenvolvedores e designers podem usar o JavaFX para imaginar, criar e expressar uma experiência real que salta do navegador e adentra nossas vidas cotidianas. O JavaFX combina, de forma poderosa, uma linguagem de scripts (JavaFX Script), uma plataforma-cliente avançada e um conjunto de ferramentas que permitem um fluxo de trabalho desenvolvedor-designer produtivo e cooperativo.

Isso conduz a tecnologia a novas perspectivas, pois reduz drasticamente o ciclo de produção entre *designers* e desenvolvedores além de possibilitar o desenvolvimento ágil de aplicações por meio de trabalho mútuo.

Seguindo o tema exposto dessa monografia, o JavaFX traz novas possibilidades no desenvolvimento de aplicações RIA, que conforme Allaire (2002, p.2) são “modelos que combinam o poder de mídia rica da área de trabalho tradicional com a implantação e conteúdo de natureza rica das aplicações web”. Segundo Coward (2010, p. 14), “a beleza e o apelo do JavaFx estão na forma simples de ele poder expressar as características móveis de aplicações bonitas, envolventes, fluídas e atraentes”.

Essa interessante plataforma, que une variados conceitos, é a promessa onipresente da tecnologia Java para aplicações RIA, que cria no mercado um novo conceito de interatividade e usabilidade, levando os usuários a experiências mais ricas no uso de interfaces.

Assim, este capítulo tem o intuito de apresentar um breve histórico sobre a plataforma JavaFX, que é visto na subseção 1.1, e as tendências atuais no desenvolvimento de aplicações, que são vistas na subseção 1.2.

1.1. HISTÓRIA

Conforme Rehem e Telemaco (2010), por muitos anos a Sun Microsystems tentou estabelecer o Java como plataforma de desenvolvimento de aplicações RIA. Porém, sua tecnologia “Java Applets” não perdurou e surgiram tecnologias mais sólidas para o desenvolvimento das aplicações. Com isso, a empresa focou em melhorar a plataforma Java e fez poucos investimentos na área de aplicações com execução no lado do cliente. Porém, essa situação alterou-se com a concepção do conceito RIA e uma promissora tecnologia (JavaFX) que havia sido desenvolvida por uma empresa de médio porte.

A tecnologia JavaFX originou-se em uma empresa chamada SeeBeyond, que foi adquirida pela Sun Microsystems. Conforme Weaver (2010), na SeeBeyond havia a necessidade de criação de interfaces gráficas mais elaboradas com o objetivo de entreter os usuários, mas que deveria ser de fácil e rápida implementação. A ideia original partiu de um integrante da empresa chamado Chris Oliver, que iniciou a prototipação de uma linguagem de programação denominada F3 (*Form Follows Function*), que, após a aquisição pela Sun, foi renomeada para JavaFX Script. A linguagem é declarativa e dinâmica, com uma sintaxe agradável e produtiva, sendo ideal para desenvolvedores acostumados com a linguagem Java. Conforme Weaver (2010), a versão inicial da linguagem era puramente interpretada e considerada hoje um protótipo da atual linguagem JavaFX Script, que é compilada.

De forma a engrenar a plataforma Java no mercado de aplicações RIA, iniciou-se um exaustivo trabalho na Sun para agregar o legado tecnológico do Java na versão inicial da plataforma JavaFX. Esse trabalho, conforme Weaver (2010), refere-se à união do JavaFX às tecnologias Java. Inicialmente o JavaFX passou a utilizar o JRE (*Java Runtime Environment*) como ambiente em tempo de execução e a utilizar o JWS (*Java Web Start*) como mecanismo de inicio na execução das aplicações.

Conforme Weaver (2010), no final de 2007, a versão compilada da linguagem JavaFX Script atingiu um ponto de maturidade e foi anunciada em diversos *sites* especializados em tecnologia da Internet. Após isso, houve muito progresso no desenvolvimento da plataforma e no ano de 2008, conforme Weaver (2010), o JavaFX já possuía integração com o ambiente de desenvolvimento integrado NetBeans e muitas partes da interface gráfica do usuário haviam sido re-escritas para permitir uma melhor integração com o *Swing* (interface gráfica do Java).

Um fato histórico acerca do JavaFX, conforme Weaver (2010), é que após o árduo trabalho de se criar uma versão compilada da linguagem JavaFX Script, o líder do projeto do compilador, postou uma mensagem em um *site* da Internet com a imagem da primeira aplicação JavaFX, e de acordo com Ball (2007):

> Uma analogia com um elefante me veio quando eu fui recentemente pressionado sobre quando, exatamente, a equipe do compilador de JavaFX entregaria nossa primeira liberação de aplicação. E eu respondi que não posso lhe dar uma data precisa. Isso é como empurrar um elefante através de uma porta; até que uma massa crítica passe do limiar, no entanto, o resto acontece rapidamente, e de uma maneira que pode ser mais acuradamente predita.

Mesmo sendo uma aplicação simples, a FIGURA 1 ilustra o trabalho difícil que a equipe do compilador teve, e representa o início de sucessivas liberações de versões contendo várias melhorias e funcionalidades. A FIGURA 1 ilustra a histórica aplicação, que apesar de simples, demonstra que o compilador já havia alcançado o objetivo que Tom Ball se referia.

FIGURA 1 – Primeira aplicação da versão compilada do JavaFX.
Fonte: <http://learnjavafx.typepad.com/photos/uncategorized/2007/12/05/elephantthroughthedoor.png>. Acesso em: 9 jun. 2010.

Ainda conforme Weaver (2010), o anúncio da liberação da versão 1.0 do JavaFX, que ocorreu em dezembro de 2008, representou uma base estável de código que conseguiu atrair muita atenção de vários desenvolvedores e gerentes de empresas de informática. Isso motivou a equipe da Sun Microsystems, que continuou a aprimorar e acrescentar funcionalidades a plataforma, de forma que em pouco tempo já era possível anunciar a versão 1.2. No entanto, o anúncio dessa versão é talvez o mais significativo de todos, pois conforme a Sun (2010) é considerada a versão em que todas as posteriores APIs do JavaFX serão compatíveis.

Em abril de 2010, a Sun Microsystems anunciou a versão 1.3 do JavaFX e que, conforme a Sun (2010), contém significativas melhorias em relação a versões anteriores, como novos controles de interfaces, melhorias de desempenho no ambiente em tempo de execução, menor consumo de memória, novos efeitos gráficos, fontes de texto nativas, suporte para o desenvolvimento de aplicações para TV, entre outras.

1.2. TENDÊNCIAS ATUAIS

A competitividade do mercado atual exige cada vez mais das grandes corporações a criação de soluções com melhor usabilidade e interatividade com o usuário, servindo assim de referência para o surgimento de novos paradigmas no desenvolvimento de software. Além disso, o dinamismo do cenário tecnológico possibilitou mudanças radicais na forma de desenvolvimento das aplicações e estabeleceu conceitos que estão em plena expansão. Entre esses, está o conceito de aplicações ricas para Internet, que desvincula do tradicional método de desenvolvimento de software para plataformas específicas e possibilita que as aplicações possam ser executadas de forma consistente em diversos ambientes operacionais, além de agregar a pluralidade do conteúdo da Internet.

A Internet tornou-se um importante marco na concepção de novas tecnologias, porque permite que as informações possam ser obtidas e alteradas independentes do meio de acesso. No entanto, a junção entre sistemas de software voltados ao *desktop* e a *web* só foi possível com a criação de plataformas RIA

sólidas e com ênfase no desenvolvimento ágil. Isso possibilitou que aplicações exclusivamente *web* tivessem comportamentos semelhantes a aplicações *desktop*, sem que para isso fosse necessária a utilização de recursos extras, tais como, bibliotecas e protocolos de comunicação etc.

A abrangência das aplicações ricas para Internet visa permitir que os atuais aplicativos possam fornecer melhor usabilidade, fluidez e dinamismo em suas interfaces, permitindo ao desenvolvedor focar em funcionalidades que agregam valor ao *software* desenvolvido. Além do mais, aplicações RIA normalmente podem oferecer versões similares de uma mesma aplicação de forma transparente para diversos sistemas operacionais, trabalho que seria muito árduo se fosse utilizado componentes nativos desses ambientes. Outra característica importante em aplicações RIA é o suporte a comunicação remota, em que normalmente as plataformas RIA abstraem a complexidade e oferecem mecanismos transparentes para tratar as conexões, envio e recebimento de dados.

As atuais plataformas de desenvolvimento de aplicações RIA, como o JavaFX, fornecem todo suporte necessário para que o desenvolvedor possa agregar funcionalidades complexas nas aplicações e usufruir da tecnologia de gerenciamento e controle fornecido pela plataforma, como mecanismos de segurança, abstração no armazenamento de dados, interfaces coerentes nos diversos ambientes operacionais suportados, abstração em protocolos de comunicação e avançados recursos de mídia.

O atual dinamismo do mercado de tecnologia impõe que as aplicações que desejam se sobressair, devem oferecer interfaces e funcionalidades integradas de forma harmoniosa, necessidade essa que é o princípio básico do conceito de aplicações RIA. Além do mais, conforme Alves (2010), grandes empresas têm investido recursos para que essa tendência se torne um padrão no desenvolvimento de aplicações, seja fornecendo ferramentas de suporte para criação dos aplicativos, especificando e padronizando novos meios de comunicação, permitindo interoperabilidade entre os sistemas operacionais ou fornecendo plataformas completas para o desenvolvimento integrado e ágil de aplicações RIA.

2. JAVAFX

O Capítulo 2 deste trabalho monográfico tem o intuito de apresentar as características internas da plataforma JavaFX. Na seção 2.1, é visto um detalhamento dos componentes que são a base da arquitetura do JavaFX e as características que compõem a plataforma, incluindo uma exemplificação da linguagem JavaFX Script. As seções 2.2 e 2.3 abordam as peculiaridades de JavaFX para ambientes *desktop* e *web*, respectivamente, assim como o modelo de desenvolvimento e a seção 2.4 apresenta uma aplicação desenvolvida em JavaFX.

2.1. ARQUITETURA

A plataforma JavaFX, conforme a Sun Microsystems (2010), é “uma avançada e significativa plataforma-cliente para criação e geração de experiências avançadas de Internet em todos dispositivos utilizados.”. Essa afirmação faz referência de que a plataforma é um ambiente em tempo de execução que suporta uma variedade de dispositivos, como televisões, celulares e computadores.

A arquitetura de JavaFX foi elaborada de forma a abstrair a complexidade no tratamento de vários dispositivos diferentes e propiciar uma forma simples e intuitiva no desenvolvimento de aplicações, além de reúso de tecnologias já existentes. Uma vez que sua plataforma baseia-se na Máquina Virtual Java, o JavaFX herda os princípios de portabilidade e interoperabilidade entre sistemas operacionais, beneficiando-se assim da maturidade, desempenho e ubiquidade da plataforma Java. Tais características são importantes no atual cenário de desenvolvimento, pois permite que aplicações sejam escritas em uma linguagem portável e largamente utilizada em fábricas de *software*.

Conforme pode ser observado na FIGURA 2, a plataforma JavaFX é modularizada em camadas bem definidas:

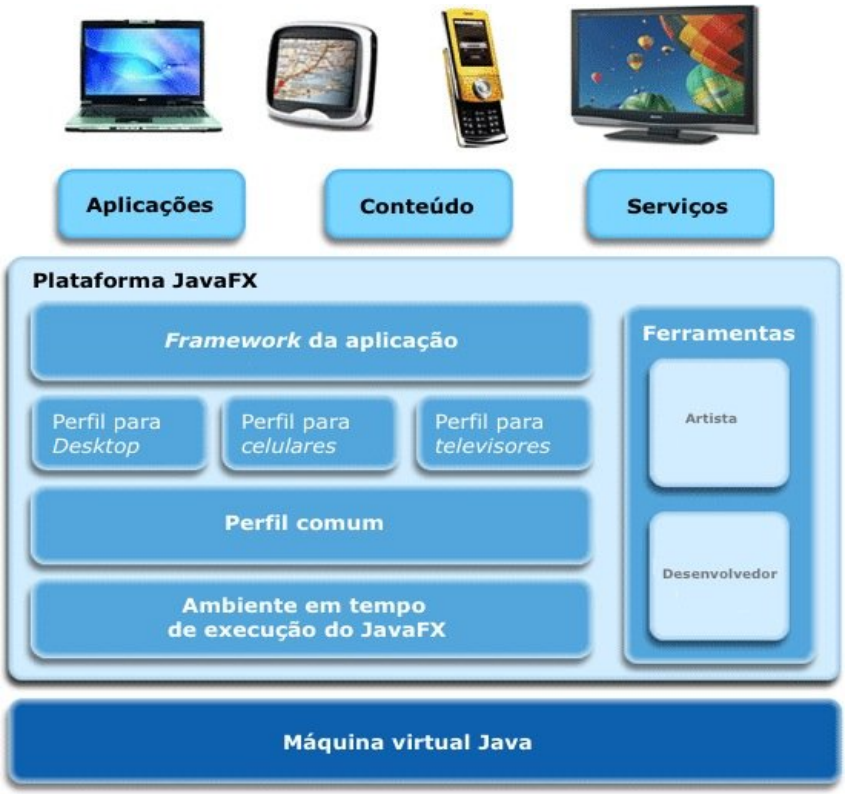

FIGURA 2 – Arquitetura da plataforma JavaFX.
Fonte: Adaptado de <http://java.sun.com/developer/technicalArticles/javame/javafxmobile-javame/figure1.gif>.
Acesso em: 13 abr. 2010.

Conforme a Sun Microsystems (2010), nessa arquitetura, podemos destacar os seguintes componentes:

a) **Máquina Virtual Java:** Encontra-se na base da arquitetura da plataforma JavaFX, com a responsabilidade de interpretar e executar o *bytecode* produzido pelo ambiente em tempo de execução do JavaFX. Através de seu engenhoso método de geração de código nativo, técnica esta conhecida por JIT (*Just In Time*), que conforme Terra *et. al*[1] (2010) é uma tecnologia presente na plataforma Java que durante a execução da aplicação, efetua a tradução do *bytecode* Java para código nativo do sistema operacional em que está sendo executado, permitindo assim um resultado altamente otimizado para determinadas operações. Isso permite

[1] TERRA, Ricardo. et al. Análise comparativa do código gerado por compiladores Java e C++., 2010.

uma grande melhora no desempenho de execução das aplicações JavaFX;

b) **Ambiente em tempo de execução do JavaFX:** O ambiente em tempo de execução do JavaFX, através de seu compilador que estende as características do compilador Java, gera *bytecode* compatível com a JVM, possibilitando assim a plataforma usufruir de uma máquina virtual estável. Além disso, é totalmente integrado ao JRE, o que permite utilizar toda a API de Java nas aplicações JavaFX;

c) **Perfil comum:** São APIs comuns a todos modelos de desenvolvimento, com destaque aos gráficos 2D, animações, texto, áudio e vídeo. Por meio dessa característica, a plataforma permite uma grande reutilização do código escrito;

d) **Perfis para *desktop*, celulares e televisores:** É um conjunto de APIs projetadas para funcionalidades específicas para cada um dos modelos de desenvolvimento: *desktop*, celulares e televisões;

e) ***Framework* da aplicação:** São as funcionalidades utilizadas para o desenvolvimento de aplicações JavaFX, o que inclui a linguagem JavaFX Script, que é utilizada para a escrita das aplicações;

f) **Ferramentas de apoio ao desenvolvimento:** JavaFX permite que tarefas sejam segmentadas em áreas especializadas tais como *design* e desenvolvimento. Para simplificar essa divisão, *plugins* foram desenvolvidos para os principais programas gráficos utilizados pelos *designers* - Adobe Illustrator e Adobe Photoshop - e que tem a capacidade de gerar códigos na linguagem JavaFX Script, preservando assim a estrutura do trabalho. Além disso, possui suporte aos principais ambientes de desenvolvimento utilizado pelos programadores, como NetBeans e Eclipse.

A base arquitetural possui algumas características bem definidas que visam permitir um reúso das tecnologias e padrões já existentes e que serão apresentados na subseção seguinte.

2.1.1. Características

A plataforma JavaFX foi projetada para reutilizar grandes partes da estável tecnologia Java. Devido a essa integração e, principalmente em relação à compatibilidade total com o JRE, é possível utilizar a variedade das classes Java nas aplicações JavaFX.

JavaFX foi elaborado com o conceito de perfil comum, que conforme a Sun (2010), são componentes reutilizáveis em todos dispositivos suportados, o que permite uma redução do tamanho do núcleo da plataforma e consequentemente um reúso de APIs internas. Conforme Chen (2008), nesse perfil, podemos destacar as seguintes características:

a) **Apresentação gráfica:** A apresentação gráfica representa a interface que será exibida pela aplicação. Dentre seus elementos destacam-se:

1. Formas geométricas;
2. Linhas;
3. Arcos;
4. Transparência;
5. Preenchimento de cores;
6. Textura;
7. Suporte a exibição em tela cheia;
8. Transformações tais como: girar, redimensionar e inclinar.

b) **Apresentação textual:** Refere-se ao modo como os textos da interface serão apresentados na aplicação. Dentre seus elementos destacam-se:

1. Renderização de fontes;
2. Transformações tais como: girar, redimensionar e inclinar.

c) **Animação:** Permite a criação de forma fácil de conteúdo animado para as aplicações. Dentre seus elementos destacam-se:

1. Animações tais como: rotação e aproximação;
2. Transições animadas entre componentes.

d) **Áudio e vídeo:** Compõe a camada de apresentação incorporada de vídeo e áudio. Dentre seus elementos destacam-se:

1. Suporte a áudio e vídeo digitais;

2. Reprodução, pausa, pesquisa, volume e controle de velocidade;
3. Fluxo de mídia sobre HTTP com utilização de *buffer.*

Para permitir que o desenvolvedor não se preocupe com peculiaridades de cada dispositivo suportado, foi desenvolvida uma linguagem de programação que é utilizada para a construção das aplicações JavaFX. A linguagem JavaFX Script, é de acordo com a Sun Microsystem (2010) ”uma linguagem de programação de script de alto desempenho que permite criar e disponibilizar a próxima geração de aplicações avançadas de Internet para desktop, celulares e televisores.”. A FIGURA 3 ilustra as camadas e classes que compõem a estrutura da linguagem:

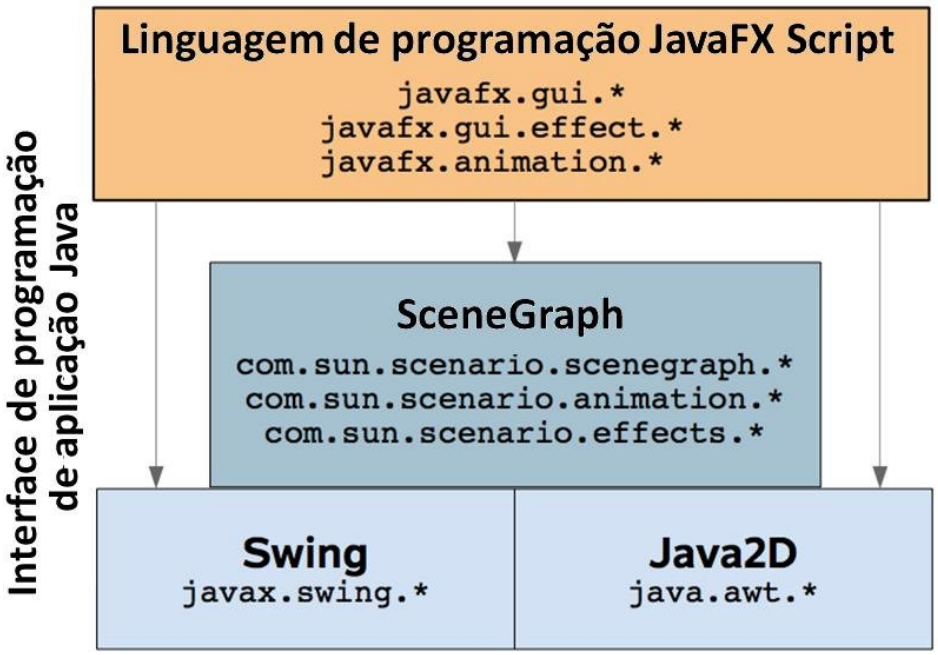

FIGURA 3 – Classes da linguagem JavaFX Script.
Fonte: Adaptado de <http://www.javapassion.com/javafx/javafx_overview.pdf>.
Acesso em: 13 fev. 2010.

Conforme Weaver (2010), dentre as diversas características da linguagem JavaFX, destacam-se:

a) É orientada a objetos;

b) É uma linguagem com estilo funcional, onde as funções podem ser atribuídas a variáveis;

c) É uma linguagem de expressão, ou seja, todo código JavaFX consiste de expressões;

d) É uma linguagem declarativa, o que é conveniente para o desenvolvimento das interfaces para o usuário;

e) Possui a característica de ser uma linguagem que segue o raciocínio lógico, permitindo assim a descrição mais fácil das interfaces para o usuário;

f) Permite, de forma simples a separação de modelos e visualizações das interfaces;

g) É uma linguagem fortemente tipada, com capacidades de inferência de tipo básico, ou seja, o compilador identifica os tipos e reporta quaisquer erros;

h) O código JavaFX Script é compilado para classes Java;

i) Possui quatro tipos de dados: tipos primitivos, de vetores, de função e de classe.

A FIGURA 4 ilustra uma exemplificação de código da linguagem JavaFX Script:

```
package javafxapl;

import javafx.stage.Stage;
import javafx.scene.Scene;
import javafx.scene.image.Image;
import javafx.scene.image.ImageView;
import javafx.scene.text.Text;
import javafx.scene.text.Font;
import javafx.scene.text.FontWeight;

// instancia da imagem de fundo
var img = Image { url: "{__DIR__}javafx.png" };

Stage {
    width: 250
    height: 232
    title: "Aplicação JavaFX"
    scene: Scene {
        content: [
            ImageView { // seta imagem de fundo

                image: img
            }
            Text {
                content: "JavaFX!"
                x: 75
                y: 150
                font: Font.font("Monotype Corsiva", FontWeight.BOLD, 25)
            }
        ]
    }
}

```

FIGURA 4 – Código JavaFX Script.
Fonte: Do autor.

A FIGURA 5 ilustra a interface criada após execução da aplicação de base no código da FIGURA 4:

FIGURA 5 – Aplicativo JavaFX em ambiente *desktop*.
Fonte: Do autor.

O trecho de código ilustrado na FIGURA 4 representa:

a) **Linhas 1-9:** Cria uma estrutura de diretório com nome “javafxapl”, indica as classes que são dependências da aplicação e que necessitam fazer parte do escopo para execução normal da aplicação;
b) **Linha 11:** Assim como na linguagem Java, nessa linha pode-se observar um comentário de final de linha. Convém mencionar que o comentário do trecho de código (/* */) também é permitido;
c) **Linha 12:** É criada uma instância da classe *Image* que representa a imagem de fundo da aplicação. A classe *Image* representa itens gráficos na aplicação e é utilizada para carregar imagens de diretórios locais ou remotos;
d) **Linha 14:** É declarado um objeto do tipo *Stage* que é basicamente a interface de usuário das aplicações JavaFX, completamente independente do dispositivo;

e) **Linhas 15-17:** São variáveis de instância que representam as características de largura, altura e título da aplicação, respectivamente;

f) **Linha 18:** Essa linha representa uma cena na aplicação, definida pela classe *Scene.* Essa classe contém os elementos gráficos da aplicação e que serão apresentados na *Stage*, que representa uma janela na aplicação;

g) **Linhas 20-23:** Esse trecho do código indica a imagem de fundo que será apresentada na aplicação, sendo que a classe *ImageView* é utilizada para desenhar as imagens que são representadas pela classe *Image*. Além disso, o caractere "[" indica o início de uma sequência lógica de ações;

h) **Linhas 24-30:** Essa sequência, definida pela classe *Text,* exibe na janela da aplicação a palavra JavaFX com uma fonte *Monotype Corsiva* em negrito e tamanho de 25 *pixels*. Além disso, o caractere "]" indica o fim de uma sequência lógica que foi iniciada pelo caractere "[" na linha 15.

A linguagem JavaFX Script foi criada para permitir que desenvolvedores possam atuar de forma mais próxima com outras áreas do processo de produção da aplicação e, por estes fatos, é uma linguagem que consegue unir conceitos legados a técnicas atuais de desenvolvimento como o uso da plataforma Java.

2.2. AMBIENTE DESKTOP

As tradicionais aplicações nativas para *desktop* são normalmente construídas utilizando-se as APIs de cada sistema operacional, isso consequentemente possibilita considerável ganho no desempenho da aplicação, contudo aumenta a complexidade de desenvolvimento. Assim, o plano de desenvolvimento da plataforma JavaFX incorpora a execução de suas aplicações em vários ambientes operacionais através do uso da JVM, como citado anteriormente.

A subseção seguinte apresenta as características específicas da plataforma JavaFX para ambiente *desktop*, citando a forma de gerenciamento de execução das aplicações e as classes que representam o perfil sobredito.

2.2.1. Características

Para abstrair a forma de execução das aplicações JavaFX em ambientes *desktop*, foi utilizado o JWS, componente do JRE. Conforme a Sun (2010), ele permite que automaticamente seja feito o *download* da última versão da aplicação e garante que a versão mais recente seja executada, abstraindo o desenvolvimento de toda complexidade na criação e distribuição da aplicação para vários sistemas operacionais.

De acordo com a Sun (2010), para a aplicação utilizar o JWS, ela deve ser assinada digitalmente e, caso não se tenha um emissor confiável do certificado, ela será assinada como *self signed jar*, que significa que ela foi assinada digitalmente, mas não possui um autorizador para o certificado. Com isso, ao executar uma aplicação auto-assinada que utilize recursos externos, como a Internet, torna-se um incômodo, pois será exibido um diálogo solicitando autorização por parte do usuário para a liberação do recurso.

As interfaces do JavaFX utilizam alguns componentes do *Swing*, interface gráfica padrão do Java, porém de acordo com Giles (2010) “não é a intenção do JavaFX criar uma simples camada sobre os componentes Swing/AWT, pois estes não são portáveis para celulares, dispositivos e televisões, o qual faz parte dos planos do JavaFX.”. A FIGURA 6 ilustra as classes e camadas que compõem a arquitetura JavaFX para ambientes *desktop*, além de demonstrar claramente a próxima relação com as classes Java.

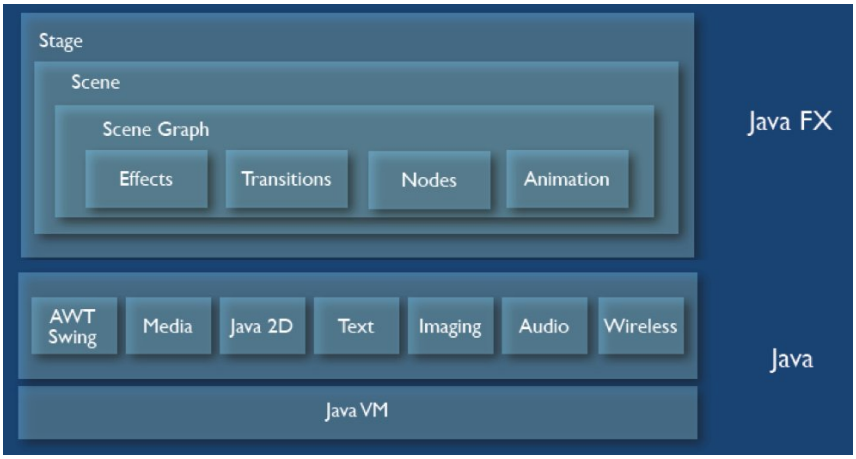

FIGURA 6 – Componentes JavaFX específicos para *desktop*
Fonte: Adaptado de
<http://www.devnexus.com/static/2009/presentations/Keynote_JavaFX_Doris.pdf>.
Acesso em: 16 abr. 2010.

Ainda conforme Giles (2010), nessa arquitetura, destacam-se os seguintes componentes:

a) ***Stage*:** A classe *Stage* está no topo da hierarquia de interface do usuário para qualquer aplicação JavaFX. Ela representa o ponto principal de inicio da aplicação;

b) ***Scene*:** Essa classe representa o segundo nível na hierarquia das classes nas aplicações JavaFX. Ela contém todos os elementos da interface do usuário criada na aplicação, sendo tais elementos denominados “nós gráficos”. As subclasses que herdam diretamente de *Scene*, representam as características de apresentação da aplicação, em que na FIGURA 6 são representadas por:

1. ***Effects*:** São efeitos de animação, tais como reflexão e brilho que foram criados para utilização de forma trivial por parte do desenvolvedor da aplicação;
2. ***Transitions*:** São efeitos de transição de componentes da aplicação que são baseadas em tempo de exibição e permitem transições suaves;
3. ***Nodes*:** Em uma aplicação JavaFX, os nós são ditos como os elementos gráficos da aplicação. A classe *Node* contém eventos

relacionados ao *mouse*, cursores, translação na tela por coordenadas;

4. ***Animation*:** Essa classe contém as funcionalidades que permitem uma aplicação JavaFX prover conteúdo de animação, tais como transições por tempo, rotacionamento.

As classes adicionais do perfil de desenvolvimento para *desktop* possibilitam criar aplicações que possam usufruir melhor das funcionalidades específicas do ambiente operacional já que ela não é executada em um ambiente restrito como um navegador *web*.

2.3. AMBIENTE WEB

Durante a década de 90 a tecnologia de Java Applets demonstrava todo potencial no desenvolvimento de aplicativos que poderiam ser executados diretamente nos navegadores *web*. Devido à complexidade, falta de produtividade e escassez de boas ferramentas, a utilização dessa tecnologia se tornou menos frequente. O avanço no desenvolvimento dos navegadores *web* e a concepção do conceito de aplicações ricas para Internet permitiram novos investimentos nessa área e conforme a Sun (2010), por volta de dezembro de 2008, foi demonstrado a primeira versão de JavaFX, que tem como objetivo revigorar a plataforma Java no desenvolvimento de aplicativos com execução do lado do cliente.

A subseção 2.3.1 apresenta as características específicas da plataforma JavaFX para ambiente *web* e sua forma de desenvolvimento.

2.3.1. Características

A proposta de revigorar os Applets com a utilização do JavaFX exigiu melhorias na base arquitetural da plataforma. Conforme Bar (2009), as principais alterações foram:

a) Atualização 10 do Java 6 permitiu:

1. Modularização, o que permite que os componentes sejam carregados por demanda;
2. Eficiente *download* e instalação dos módulos;
3. Novo componente para aplicativos Java, proporcionando maior agilidade em seu início.

b) Novo *plugin* para navegadores *web*:

1. Reescrito do zero, seguindo uma nova arquitetura;
2. Os Applets agora executam em processos separados, independente do navegador, com isso tem-se maior controle e estabilidade.

c) Reescrita do *LiveConnect JavaScript Bridge*:

1. Possibilitou melhor interação via JavaScript entre o navegador *web* e o Applet. O *LiveConnect* permite que um Applet comunique transparentemente com um código JavaScript.

O JavaFX foi elaborado para permitir que uma aplicação seja executada em diferentes ambientes com o mínimo de modificação. Uma aplicação JavaFX desenvolvida para ser executada dentro do navegador *web*, necessita da criação de um arquivo JNLP (*Java Network Launching Protocol*) e a inserção de uma chamada a aplicação no código HTML (*HyperText Markup Language*). A FIGURA 7 demonstra um código JavaFX Applet.

```
1   <HTML>
2     <HEAD>
3       <TITLE>Meu Aplicativo</TITLE>
4       </HEAD>
5      <BODY>
6      <SCRIPT SRC="http://dl.javafx.com/1.2/dtfx.js">
7          <SCRIPT>
8              javafx(
9              {
10                 archive: "MeuAplicativo.jar",
11                 draggable: true,
12                 width: 350,
13                 height: 200,
14                 code: "meuAplicativo.MeuAplicativo",
15                 name: "MeuAplicativo"
16             }
17         );
18         </SCRIPT>
19     </BODY>
20  </HTML>
21
```

FIGURA 7 – Código JavaScript para iniciar execução do JavaFX Applet.
Fonte: Do autor.

O trecho de código ilustrado na FIGURA 7 representa:

a) **Linhas 1-5:** O trecho de código define o início da linguagem de marcação HTML que possibilitará a inserção direta do código JavaScript;
b) **Linha 6:** O arquivo JavaScript “dtfx.js” contém definições para verificar automaticamente atualizações e geração da tag “<APPLET>”;
c) **Linhas 7-17:** O código exposto define a configuração da aplicação no navegador *web*. Nesse caso, será exibido uma aplicação que possui 200 pixels de altura e 350 pixels de largura, e o início da aplicação é configurado para o método principal da classe "MeuAplicativo".

O princípio proposto pela plataforma JavaFX é que a aplicação possa ser executada de forma transparente em todos ambientes. No entanto torna-se evidente que a forma de distribuição da aplicação deve ser adaptada ao perfil em que será executada. Conforme apresentado nessa subseção, a aplicação JavaFX para *web* necessita de uma adaptação na forma de distribuição, que é resolvido através da utilização do JWS.

2.4. A APLICAÇÃO JXPLAYER

A plataforma JavaFX fornece toda infra-estrutura tecnológica para o desenvolvimento de aplicações RIA complexas por meio de componentes integrados e simples de usar. Como forma de verificar a produtividade no desenvolvimento de aplicações JavaFX, foi elaborado um exemplo funcional, denominado JXPlayer, que tem o intuito de apresentar algumas características da plataforma. As subseções 2.4.1, 2.4.2, 2.4.3 e 2.4.4 abordam, respectivamente, as funcionalidades da aplicação, software utilizados, interface e efeitos gráficos, e o suporte à mídia.

2.4.1. Características da aplicação JXPlayer

A aplicação JXPlayer foi desenvolvida para demonstrar que funcionalidades complexas em outras linguagens, são construídas de forma simples em JavaFX. As funcionalidades da aplicação são:

- Suporte à áudio/vídeo local e remoto (*streaming*);
- Controles de reprodução de áudio/vídeo (avançar/retroceder reprodução, diminuir/aumentar volume, interromper/finalizar reprodução);
- Suporte a aplicação de efeitos em fotos;
- Execução de tarefas em segundo plano, por exemplo, escutar música enquanto manipulo uma foto;
- Efeitos de transição entre as telas;
- Efeito de ofuscamento de imagem durante reprodução de uma música;
- Barras de progressão de áudio e vídeo;
- Controles de interfaces com efeitos gráficos.

A FIGURA 8 ilustra a tela principal da aplicação.

FIGURA 8 – Aplicativo JXPlayer.
Fonte: Do autor.

2.4.2. Ferramentas de apoio ao desenvolvimento

A interface e funcionalidades da aplicação JXPlayer foram desenvolvidas no IDE NetBeans 2008 com o *plugin* JavaFX Composer 1.10.3. Conforme a Sun Microsystems (2010), JavaFX Composer é um *plugin* para o NetBeans que fornece uma infra-estrutura integrada e um ambiente visual no desenvolvimento das aplicações JavaFX.

A utilização de técnicas de *drag-and-drop*[2] providas pelo ambiente de desenvolvimento possibilitou uma redução no tempo dedicado a criação da interface gráfica e formatação dos componentes visuais em tela. Internamente, o *plugin* gera os códigos necessários para apresentar os componentes e restringe as alterações

[2] Conforme a Adobe (2010), *drag and drop* é a ação de “arrastar e soltar” determinado conteúdo em outro destino.

apenas para o modo visual no NetBeans. Com isso, as características dos componentes terão o comportamento e forma definidos pelas funcionalidades implementadas no *plugin*, o que muitas das vezes não é suficiente.

Apesar de auxiliar e automatizar a utilização de vários componentes no desenvolvimento do JXPlayer, o *plugin* apresentou instabilidade e comportamentos incoerentes. Por exemplo, uma simples modificação de aparência (alteração de uma Fonte) de um *menu* impossibilitou toda a aplicação de ser executada.

O *plugin* JavaFX Composer fornece uma forma visual e amigável no desenvolvimento de aplicações JavaFX, que agiliza a criação e integração dos componentes visuais do JavaFX na tela da aplicação. No entanto, sua instabilidade e rigidez no gerenciamento do código fonte, podem impedir que ele seja adotado como uma ferramenta principal na elaboração de aplicações JavaFX mais complexas.

2.4.3. Interface e efeitos gráficos

Conforme mencionado na subseção anterior, as ferramentas de apoio ao desenvolvimento NetBeans e JavaFX Composer ofereceram a infra-estrutura básica na criação das interfaces gráficas da aplicação. Como a plataforma JavaFX provê APIs específicas e bem estruturadas para efeitos e transições, a implementação desses recursos na aplicação foi simples e com poucas linhas de código. A simplicidade do uso foi devido à utilização da palavra chave *bind*, que conforme Weaver (2010) é um arranjo dinâmico que normalmente está associado a uma variável e reflete suas atualizações. Um exemplo da versatilidade do *bind* utilizado na aplicação é a redução dinâmica do brilho e contraste de uma imagem quando uma música está sendo reproduzida.

A criação de interfaces em JavaFX é feita de forma simples e declarativa, portanto não houve grandes obstáculos na elaboração da parte gráfica da aplicação. Porém, a utilização do JavaFX Composer restringiu a modelagem dos componentes gráficos e inviabilizou o desenvolvimento de um aplicativo mais rico em detalhes visuais.

2.4.4. Suporte a áudio e vídeo

Uma parte do escopo da aplicação JXPlayer é permitir que arquivos de mídia possam ser executados de forma transparente, seja de origem local ou remota (Internet). O JavaFX permitiu que essas funcionalidades fossem incorporadas a aplicação de forma trivial e rápida, por meio de sua API de mídia. Contudo, para o diálogo de seleção de arquivos locais, foi necessário utilizar um componente do *Swing*, interface gráfica nativa do Java, pois não existe a funcionalidade como recurso nativo do JavaFX. Isso demonstra que alguns componentes essenciais, por exemplo um componente de seleção de arquivos, ainda precisam ser incorporados ao JavaFX, além de demonstrar também a integração com a plataforma Java, pois foi possível utilizar classes nativas do Java na aplicação.

As classes do JavaFX abstraem toda complexidade na reprodução de arquivos de mídia e oferecem suporte a vários tipos de arquivos. Incorporar na aplicação a funcionalidade de reprodução de mídias locais e remotas foi trivial, porém a alternância da origem dos arquivos foi onerosa porque mesmo modificando no objeto da classe a localização da mídia, o arquivo permanecia o mesmo. A solução encontrada foi utilizar o recurso *bind* para gerar uma classe dinâmica associada ao componente de mídia do JavaFX.

As ferramentas de produtividade, NetBeans e JavaFX Composer, e a variedade de APIs públicas do JavaFX, auxiliaram o desenvolvimento da aplicação JXPlayer e possibilitaram incorporar funcionalidades que seriam muito complexas de serem agregadas caso fosse necessário desenvolvê-las por completo. No entanto, a restrição e falta de implementação de alguns componentes indicam que é necessário um prévio conhecimento da plataforma Java para elaborar soluções que não estejam nativamente incorporadas ao JavaFX.

3. COMPARATIVO COM TECNOLOGIAS SEMELHANTES

O Capítulo 3 tem o intuito de apresentar as tecnologias Adobe AIR e Silverlight e um breve comparativo com o JavaFX. Na seção 3.1 é feito uma introdução sobre o Adobe AIR e nas subseções 3.1.1 e 3.1.2 é visto as características e um comparativo com o JavaFX. Na seção 3.2 é apresentada a tecnologia Silverlight e nas subseções 3.2.1 e 3.2.2 são descritas as características e é realizado um comparativo com a plataforma JavaFX.

3.1. ADOBE AIR

Seguindo a tendência de desenvolvimento de aplicações ricas para Internet, a Adobe Systems demonstrou em 2007 a sua tecnologia que permite unificar características do ambiente *desktop* e *web* na mesma aplicação. Conforme CHAMBERS (2007, p. 6):

> Adobe AIR é um ambiente em tempo de execução multiplataforma desenvolvido pela Adobe que permite os desenvolvedores utilizarem tecnologias web para construir aplicações RIA voltadas ao ambiente desktop. Em essência, ele provê uma plataforma entre o desktop e o navegador, que combinam o alcance e facilidade de desenvolvimento do modelo web com a funcionalidade e a riqueza do modelo desktop.

Aproveitando-se de tecnologias amplamente utilizadas no modelo de desenvolvimento para *web*, como JavaScript, CSS (*Cascading Style Sheets*), HTML e além de permitir agregar características da sua tecnologia de animação para Internet (Flash), Adobe AIR representa uma forma simples e transparente para criação de aplicações híbridas com interfaces consistentes e similares para os diversos ambientes computacionais suportados.

A subseção 3.1.1 aborda as características da plataforma Adobe AIR e a subseção 3.1.2 apresenta um comparativo entre as plataformas JavaFX e Adobe AIR.

3.1.1. Características

A tecnologia Adobe AIR é um ambiente em tempo de execução expressivo para o desenvolvimento de aplicações ricas para Internet e tem como princípio a abstração da complexidade e integração com o *desktop*. No entanto, conforme Chambers (2007, p.7), “Adobe AIR não é uma tecnologia que tem o intuito de competir com as aplicações escritas nativamente para o ambiente em que são executadas”. Isso ocorre porque a Adobe decidiu adotar na plataforma tecnologias usuais do desenvolvimento para *web* e que notavelmente não provêem integração nativa com os componentes dos sistemas operacionais.

De forma a permitir essa integração, o Adobe AIR possui uma arquitetura com diversos componentes com tarefas específicas, que abstraem a forma de interação com o ambiente em que as aplicações são executadas. Essa característica permite a plataforma fornecer aplicativos com funcionalidades similares e consistentes em todos ambientes operacionais suportados. A FIGURA 9 ilustra as principais características da plataforma.

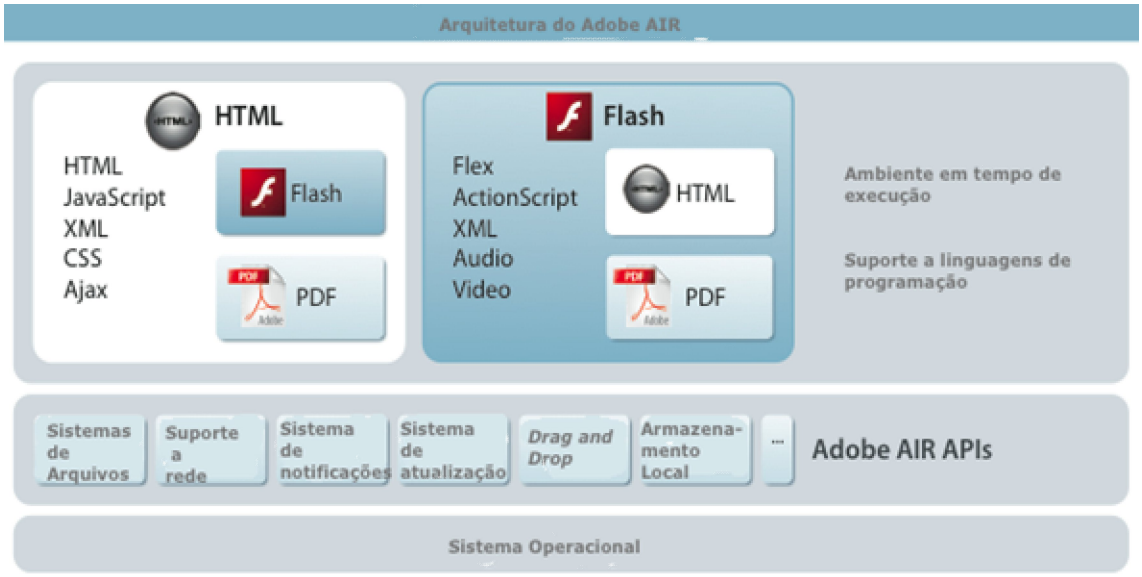

FIGURA 9 – Arquitetura da plataforma Adobe AIR.
Fonte: Adaptador de
<http://www.adobe.com/uk/products/air/pdfs/air_flex_datasheet.pdf>.
Acesso em: 13 mai. 2010.

De acordo com a Adobe (2008), as principais características da plataforma ilustradas na FIGURA 9 são:

- **Sistema de arquivos:** A plataforma consegue abstrair a interação com o sistema de arquivos nativo do sistema operacional de forma que não é necessária a reescrita do código para diferentes ambientes;
- **Suporte a rede:** Adobe AIR permite que a aplicação utilize diversos protocolos de rede padrões, por exemplo, o HTTP, que representa o principal protocolo utilizado na Internet;
- **Sistema de notificações:** Devido à integração do ambiente em tempo de execução do Adobe AIR com o *desktop,* a plataforma permite utilizar o sistema de notificações (avisos) do ambiente gráfico de forma similar para todos ambientes operacionais em que é executado;
- **Sistema de atualização:** A plataforma Adobe AIR provê um sistema próprio de atualização para as aplicações que facilita o método de atualizações das versões de uma aplicação;
- ***Drag and Drop:*** As APIs do Adobe AIR permitem que as aplicações incorporem facilmente a habilidade de transpor informações diretamente de meios externos por meio da utilização do recurso de “arrastar e soltar”;
- **Armazenamento local:** O Adobe AIR, através de sua API, fornece um método simples e integrado para armazenamento de informações em uma base de dados local criptografada.

A visão da plataforma Adobe AIR é oferecer simplicidade, integração e reutilização de tecnologias já estabelecidas. Com base nessa estratégia, ela possui suporte as tradicionais linguagens de programação e de marcação (HTML, CSS, AJAX, JavaScript) utilizadas no desenvolvimento de aplicações para Internet, além de oferecer a possibilidade de utilização da linguagem ActionScript, que é a base no desenvolvimento das aplicações Flash. Conforme Adobe (2010), existem três ferramentas (Adobe Flex Builder, Flash Professional e Adobe Dreamweaver) específicas para o desenvolvimento de aplicações AIR que oferecem um ambiente integrado e ágil para a implementação e testes. Cada ferramenta possui um

determinado público alvo e visa atrair desenvolvedores com habilidades específicas na criação de aplicações RIA voltadas ao *desktop*.

3.1.2. JavaFX e Adobe Air

As plataformas JavaFX e Adobe AIR apesar de terem propósitos semelhantes, tem origens e estruturas distintas. Conforme a Sun (2010), o JavaFX se beneficia da estrutura tecnológica estabelecida pelo Java, e, portanto sua maturidade. Além disso, utiliza o JRE como ambiente em tempo de execução, o que possibilita um alto grau de integração com a plataforma Java. Já o Adobe AIR, conforme a Adobe (2010), possui sua estrutura fundamentada na tecnologia Flash e utiliza um ambiente em tempo de execução próprio, diferente do Flash. De acordo com a Sun (2010), como o JRE está presente na maioria dos computadores pessoais, o JavaFX tem a vantagem de não precisar de quaisquer requisitos de instalação adicional, ação que, conforme a Adobe (2010), é necessária com o Adobe AIR.

Conforme a Sun (2010), as aplicações JavaFX são construídas utilizando a linguagem JavaFX Script, que foi elaborada para ser de fácil aprendizado, ter distinção lógica entre as interfaces do usuário e regras de negócio, ser dinâmica e possuir total integração com o Java. Essas características a fazem uma linguagem ideal para programadores Java que tenham o intuito de criar aplicações ricas para Internet e não queiram se preocupar com a complexidade de criação das interfaces gráficas. Já o Adobe AIR, conforme a Adobe (2010), possibilita que as aplicações sejam criadas em linguagens suportadas na *web*, como por exemplo, JavaScript, CSS e HTML. Além disso, suporta também a linguagem ActionScript, que é o principal recurso na criação das aplicações Flash. Isso permite que desenvolvedores possam adaptar facilmente suas aplicações Flash para a plataforma Adobe AIR, obtendo assim melhor aproveitamento dos recursos já existentes.

Na parte de mídia, conforme a Adobe (2010), o Adobe AIR possui suporte a vários formatos de arquivos (WMV, FLV, AVI etc), assim como o JavaFX. No entanto, conforme a Sun (2010), o JavaFX prioriza a reprodução de arquivos de mídia via *streaming*, enquanto o Adobe AIR não faz distinção no modo de

reprodução dos arquivos. As duas tecnologias também oferecem suporte aos formatos de imagens mais populares, como o JPEG e PNG, e possibilitam que efeitos complexos sejam utilizados de forma trivial em arquivos de mídia.

Ambas as plataformas oferecem a possibilidade de desenvolvedores e *designers* trabalharem em paralelo por meio da utilização de ferramentas de apoio distintas. O JavaFX provê suporte para os desenvolvedores no IDE NetBeans e possibilita através de um *plugin*, o desenvolvimento gráfico da aplicação nos aplicativos Adobe Photoshop e Adobe Illustrator. Conforme Adobe (2010), no desenvolvimento de aplicações AIR, é incluído suporte no IDE DreamWeaver e Adobe Flex Builder. Na parte de elaboração gráfica da aplicação, é possível também utilizar os aplicativos Photoshop e Adobe Illustrator. Como o Adobe AIR é um produto da Adobe, a integração da plataforma com as principais ferramentas gráficas do mercado (Photoshop e Illustrator) é mais consistente e integrada do que em JavaFX.

As soluções para criação de aplicações RIA, Adobe AIR e JavaFX, possibilitam que aplicativos interativos sejam desenvolvidos sem muito esforço. Como tecnologia emergente, o JavaFX ainda precisa estabelecer ferramentas e recursos mais integrados que facilitem o desenvolvimento de suas aplicações. Já o Adobe AIR é uma tecnologia mais madura no mercado de aplicações RIA e oferece melhores recursos de suporte (visual e desenvolvimento), o que proporciona uma melhor estrutura organizacional no desenvolvimento de aplicações RIA voltadas ao *desktop*.

3.2. MICROSOFT SILVERLIGHT

A tecnologia Silverlight foi desenvolvida pela Microsoft e representa sua iniciativa no mercado de aplicações RIA. É uma plataforma que tem sua base de sustentação em tecnologias já estabelecidas pela empresa e provê funcionalidades que foram inspiradas em seus concorrentes. No entanto, possui características particulares que a diferenciam, como o suporte a várias linguagens de desenvolvimento e vários componentes gráficos complexos prontos e perfeitamente integrados para utilização.

Conforme HORN (2009, p. 2):

> Silverlight é uma tecnologia desenvolvida pela Microsoft que habilita o desenvolvimento de aplicações ricas para Internet (RIAs) com foco para web. Silverlight é uma plataforma e um plugin, composto por um subset das funcionalidades incluídas no Windows Presentation Foundation (WPF) e no framework .NET. Quando elaborou e desenvolveu o Silverlight, a Microsoft observou os recursos oferecidos por todos os concorrentes e usou-os como linha de base para criação de um novo produto.

O Silverlight como plataforma refere-se à integração com o *framework* .NET[3] e a possibilidade de desenvolvimento nas diversas linguagens suportadas por esse. Já o *plugin* permite o uso do Silverlight em navegadores e ambientes operacionais distintos, sendo que isso é necessário para adoção dessa tecnologia no desenvolvimento de aplicações RIA.

A subseção 3.2.1 aborda as características do Silverlight e a subseção 3.2.2 apresenta um comparativo entre as plataformas JavaFX e Silverlight.

3.2.1. Características

A tecnologia Silverlight foi fundamentada sob os conceitos do *framework* .NET, que conforme a Microsoft (2010), efetua uma tradução no código das linguagens de desenvolvimento para uma linguagem intermediária que pode ser compreendida e executada pelo *framework*. Isso permite utilizar várias linguagens de programação na criação de aplicações Silverlight e exclui a necessidade de uma nova linguagem de propósito específica. Essa característica permite ao Silverlight atingir um público mais amplo que os concorrentes e agregar maior valor a plataforma.

O Silverlight, assim como o JavaFX e Adobe AIR, permite que desenvolvedores e *designers* possam trabalhar em cooperação entre si por meio da utilização de ferramentas independentes que oferecem a infra-estrutura para criação de interfaces gráficas e desenvolvimento do código. Conforme a Microsoft (2010), o software Express Studio é destinado à criação das interfaces e tem como público

[3] Conforme a Microsoft (2010), .NET é uma plataforma de desenvolvimento de aplicações com funcionalidades similares ao Java.

alvo os *designers*, já o Visual Studio é um produto para desenvolvedores que oferece uma variedade de funcionalidades que auxiliam na organização e melhoria na qualidade do código. A FIGURA 10 ilustra a arquitetura e as principais características da plataforma.

FIGURA 10 – Arquitetura da plataforma Silverlight.
Fonte: Adaptado de
<http://msdn.microsoft.com/pt-br/magazine/cc163404.aspx>.
Acesso em: 08 jun. 2010.

De acordo com a Microsoft (2007), as principais funcionalidades da plataforma ilustradas na FIGURA 10 são:

- **Ambiente em tempo de execução:** O Silverlight utiliza o mesmo mecanismo de controle de execução do *framework* .NET, sendo esse responsável pelo controle e gerenciamento durante a execução da aplicação;

- **JavaScript:** A API de JavaScript suportada pelo Silverlight é utilizada para permitir que a plataforma consiga interpretar e responder a eventos de ações durante a execução do aplicativo;
- **XAML (*Extensible Application Markup Language)*:** O Silverlight, de acordo com Durães (2007), utiliza a linguagem XAML, que é baseada no XML (*Extensible Markup Language*), para criação das interfaces gráficas das aplicações. Com essa linguagem, é possível separar claramente o *design* da aplicação da camada referente ao código da aplicação;
- ***Codecs*[4] de mídia:** Os *codecs* de mídia permitem a plataforma Silverlight fornecer acesso a recursos avançados de mídia para as aplicações, sendo que há suporte a vídeos em alta definição e áudio digital;
- **Núcleo de apresentação:** O núcleo da apresentação representa o cerne (início) da aplicação e é gerenciado pelo ambiente em tempo de execução;
- ***Plugin* do navegador:** Uma aplicação Silverlight pode ser executada em diferentes navegadores *web* por meio da utilização de um *plugin* escrito especificamente para a versão do navegador e sistema operacional em que será executado.

Ainda conforme a Microsoft (2010), algumas características que estão presentes no Silverlight:

- ***Multi-threading*:** Essa característica permite ao Silverlight ter melhor desempenho em computadores que possuem mais de um processador, de forma que a execução das instruções é paralela e, portanto, mais eficiente e melhor aproveitada;
- **Controles gráficos:** O Silverlight fornece uma suíte de controles gráficos pré-fabricados, como calendários, campos para texto, variadas formas de botões, entre outros. Isso permite que a aplicação seja rapidamente construída com o uso desses componentes;

[4] De acordo com a Microsoft (2010), *codecs* são codificadores/decodificadores de áudio e vídeo digitais.

- **Modelo:** Através dessa característica, uma aplicação Silverlight pode ter a aparência gráfica customizada pelo desenvolvedor, que utiliza artifícios como o CSS para personalização dos componentes;
- **Armazenamento de dados:** O Silverlight fornece um banco de dados local, seguro e com recursos de criptografia. Isso permite ao desenvolvedor utilizá-lo de forma similar em todos ambientes suportados;
- **Acesso a *Web Services*:** Conforme Pamplona (2009), um *web service* é uma solução para integrar diferentes sistemas de informação normalmente utilizando a linguagem XML. O Silverlight possibilita que as aplicações forneçam e obtenham dados de *Web Services* de forma transparente.
- **Localização:** Esse recurso permite que uma aplicação Silverlight ofereça suporte aos dados em vários idiomas, não necessitando de uma versão para cada região ou idioma escolhido;
- **Aceleração por GPU (*Graphics Processing Unit*):** É uma interessante característica que permite que seja feito o processamento de vídeos e imagens diretamente na placa de vídeo, fazendo com que a execução seja mais rápida e livrando o processador principal para outras tarefas;
- **Perspectiva 3D:** A tecnologia Silverlight oferece nativamente recursos para tornar imagens e vídeos da aplicação em uma perspectiva de três dimensões, necessitando apenas a alteração das coordenadas referente aos objetos;
- **Impressão, microfone e *webcam***: A plataforma Silverlight provê capacidade para as aplicações interagirem transparentes com *hardware* de impressoras, microfones e *webcam.* Suportes a esses recursos fazem uma aplicação mais rica em detalhes para o usuário final.

Como o Silverlight destina-se apenas ao desenvolvimento de aplicações RIA voltadas para *web,* esse se torna um concorrente direto a tecnologia Adobe Flash, que de acordo com a Adobe (2010), está instalado em cerca de 98% de todos computadores pessoais do mundo conectados a Internet. Para tentar reverter à desvantagem, a Microsoft disponibiliza o *plugin* do Silverlight para a maioria dos

navegadores *web*, porém restringe seu acesso aos ambientes operacionais Windows e Mac OS X, conforme ilustrado na FIGURA 10. No entanto, para suprir essa aparente falha, surgiram alternativas livres que hoje possibilitam que aplicações Silverlight possam ser executadas em ambientes operacionais não suportados oficialmente. Uma dessas alternativas é Moonlight, que conforme a Novell (2010):

> É uma implementação livre do Silverlight, primariamente para ambientes baseados no Linux e Unix. Em setembro de 2007, Microsoft e a Novell anunciaram uma colaboração técnica que inclui acesso a suíte de testes da Microsoft para o Silverlight, além da distribuição de um pacote de mídia para usuários Linux, que contém o licenciamento de codecs proprietários para áudio e vídeo.

Apesar de ser um produto com qualidade comprovada, o Moonlight ainda carece de implementação de todos os recursos do Silverlight e requer vários pré-requisitos durante sua instalação. Além do mais, como é baseado em uma tecnologia proprietária, usuários de sistemas Linux acreditam em problemas de licenciamento, o que obviamente desestimula seu uso.

Para criação de interfaces gráficas, o Silverlight disponibiliza a linguagem XAML, que conforme a Microsoft (2007):

> Uma linguagem baseada em XML, que pode ser usado para definir recursos gráficos, interfaces do usuário, comportamentos, animações e muito mais. Foi introduzido pela Microsoft como a linguagem de marcação usada no Windows Presentation Foundation, uma tecnologia orientada a desktops que faz parte do .NET Framework 3.0, e projetado para ajudar a fazer a ligação entre o trabalho de designers e desenvolvedores na criação de aplicativos.

Essa linguagem é extensivamente utilizada por *designers* para a modelagem de interfaces e recursos gráficos da aplicação, sendo também referência para os programas de apoio ao desenvolvimento para representação das interfaces do usuário.

Conforme Horn (2009), no suporte a mídia, uma aplicação Silverlight é capaz de reproduzir vídeos em alta definição, fornecer acesso via *streaming* a vídeos, suporte a exibição de imagens de formatos populares (JPEG e PNG, por exemplo) e suporte a áudio digital. Além disso, o processamento de vídeos e imagens através da placa de vídeo permite um melhor desempenho da aplicação e, portanto, fornece uma solução de mídia eficiente e produtiva.

De acordo com a Microsoft (2010), o Silverlight oferece acesso aos principais protocolos utilizados no mercado, como SOAP (*Simple Object Access*

Protocol), integração nativa com as linguagens XML, JSON (*JavaScript Object Notation*) e suporte a formatos de dados como RSS (*Really Simple Syndication*) e ATOM[5]. Dessa forma, a tecnologia agiliza o desenvolvimento das aplicações, porque não é necessário criar soluções extras para agregar o conteúdo da Internet na aplicação que está sendo desenvolvida, além de oferecer todo aparato tecnológico para ser utilizado de forma segura e fácil.

Uma plataforma para criação de aplicações RIA deve oferecer interfaces bem elaboradas e efeitos gráficos avançados. Conforme a Microsoft (2010), o Silverlight fornece um conjunto de efeitos visuais (rotação, inclinação, *zoom* de texto e imagens, entre outras) que podem ser utilizados de forma simplória pelas aplicações. Isso possibilita que uma aplicação Silverlight possa oferecer transições e efeitos visuais de primeira linha, sem o custo extra de tempo e implementação completa de cada efeito.

A plataforma Silverlight é um investimento da Microsoft no promissor mercado de aplicações RIA, que oferece a maioria dos recursos de seus concorrentes e contêm diferenciais como, por exemplo, a utilização de diversas linguagens de programação para o desenvolvimento das aplicações. No entanto, como é uma tecnologia recente, não abrange uma participação muito ampla no mercado, mas que conforme Krill (2009) tende a mudar decorrente ao investimento que se tem feito na tecnologia e ao crescimento das aplicações RIA.

3.2.2. JavaFX e Microsoft Silverlight

Conforme Horn (2009), a plataforma Silverlight tem sua arquitetura orientada pelo *framework* .NET e utiliza-o para o gerenciamento e controle da execução das aplicações. Essa característica possibilita o Silverlight oferecer suporte a mais linguagens de programação. Já o JavaFX restringe a criação das aplicações a linguagem JavaFX Script e tem seu gerenciamento controlado pela

[5] Conforme a Microsoft (2010), ATOM é um protocolo para publicação de dados que são regularmente atualizados. Possui similaridades com o RSS.

máquina virtual Java. Ambas as plataformas utilizam mecanismos de JIT para aumento no desempenho das aplicações.

O JavaFX permite a reprodução de arquivos de mídia proprietários, como o WMV, mas desde que os *codecs* necessários estejam instalados corretamente. Como um produto da Microsoft, o Silverlight possui a vantagem de ter nativamente suporte ao formato WMV já que esse também é uma tecnologia da empresa. Uma clara vantagem do Silverlight é a possibilidade de processamento de vídeos e imagens diretamente no processador da placa de vídeo, aliviando o processador principal do computador para outras tarefas. Ambas as plataformas permitem a reprodução de vídeos em alta definição e possuem suporte a áudio digital nativamente.

Outra funcionalidade interessante na plataforma, segundo a Microsoft (2010) é o amplo suporte a tecnologias utilizadas na *web*. A plataforma consegue comunicar com código JavaScript presente em páginas da Internet, possui completo suporte a requisições AJAX e abstrai camadas de mais baixo nível na utilização do suporte a rede. Já o JavaFX, conforme citado na subseção 2.3.1, comunica-se com o JavaScript através do recurso *LiveConnect,* que oferece recursos similares ao Silverlight.

Uma característica importante do Silverlight é fornecer muitos componentes gráficos já prontos e integrados em suas ferramentas de apoio ao desenvolvimento. Isso pode ser visto em ferramentas como Visual Studio e Express Studio. O JavaFX também possui componentes pré-fabricados, mas conforme citado na seção 2.4.4, ainda carece de alguns essenciais, como um componente para seleção de arquivos. Além disso, o *plugin* JavaFX Composer ainda é limitado e não possui a mesma expressividade do software Visual Studio.

A plataforma JavaFX, por utilizar o JRE, possui a vantagem de suportar vários dispositivos e sistemas operacionais distintos. Conforme a Sun Microsystems (2010), atualmente é possível executar aplicações JavaFX em celulares, TVs e nos sistemas operacionais Windows, Mac OS X, Linux e Solaris. Já o Silverlight, conforme a Microsoft (2010), é restrito aos ambientes Windows e Mac OS X. Porém, há alternativas não oficiais que possibilitam que aplicações Silverlight possam ser executadas em outros ambientes não suportados.

Ambas as plataformas oferecem recursos similares, mesmo com tecnologias distintas, para a criação de aplicações RIA. O JavaFX possui a

vantagem de oferecer melhor portabilidade da aplicação para diferentes ambientes operacionais enquanto o Silverlight contém a peculiaridade de suportar várias linguagens de desenvolvimento. As plataformas estruturam-se em bases tecnológicas sólidas e com grande reconhecimento do mercado e equiparam-se em suporte a tecnologias e efeitos gráficos.

4. CONSIDERAÇÕES FINAIS

As aplicações RIA trouxeram de volta o desenvolvimento de aplicativos de processamento no lado cliente. Isso só foi possível devido à evolução das tecnologias vigentes e ao âmbito de crescimento da Internet no cenário mundial.

Grandes corporações têm adotado o conceito de RIA em suas aplicações de forma a oferecer em tempo hábil aplicações ricas em conteúdo e com capacidade de execução em vários sistemas operacionais. Além disso, aplicativos RIA com foco na *web* tem sido uma forma de oferecer acesso ao sistema de software sem que o usuário necessite preocupar-se com requisitos mínimos para execução das aplicações em seus computadores pessoais.

O problema no desenvolvimento de aplicações nativas para os sistemas operacionais é a enorme complexidade em se tratar APIs específicas e fornecer interfaces similares em todos os ambientes. Além disso, não há uma forma simplificada de distribuição da aplicação sem que haja necessidade de várias versões do mesmo software, dificultando assim a geração e controle de versões.

O conceito de aplicações RIA permite que esses problemas sejam sanados, pois o gerenciamento e controle da aplicação normalmente são realizados por sistemas que atuam como camadas de abstração entre a aplicação e os sistemas operacionais.

Em visto disso, o JavaFX − criado pela Sun Microsystems − surgiu para simplificar e permitir que interfaces mais aprazíveis sejam criadas e executem de forma uniforme em vários sistemas operacionais. O JavaFX também é uma tecnologia que oferece uma forte integração com o Java, o que possibilita que muitos componentes sejam reutilizados, facilitando assim a adaptação à plataforma JavaFX da vasta gama de desenvolvedores Java.

Nesse aspecto, o estudo prova, a partir da especificação técnica da plataforma JavaFX, que muitas funcionalidades complexas são integradas a tecnologia, como o extenso suporte a arquivos de mídia. Através de um exemplo de pequeno porte, é apresentado também a linguagem JavaFX Script, que possibilita que interfaces sejam elaboradas de forma declarativa e dinâmica. Ademais, devido ao ambiente controlado sobre uma máquina virtual (JRE, no caso), as aplicações podem ter aspectos semelhantes no ambiente *desktop* e *web*.

Atualmente, reunir funcionalidades de mídia em aplicações nativas é dispendioso e requer a utilização de várias bibliotecas distintas para oferecer suporte a diferentes tipos de arquivos. Isso acarreta uma complexidade extra no desenvolvimento da aplicação e é necessário um conhecimento prévio de todas APIs das bibliotecas utilizadas.

Através do desenvolvimento do JXPlayer, este estudo conclui que a plataforma JavaFX permite que mídia rica pode ser agregada a aplicação de forma simples e com resultados eficazes. Conclui-se também que devido a restrições de licenciamento, não é possível executar alguns formatos de arquivos − como o WMV, por exemplo − sem requisitos adicionais de instalação. Isso obviamente se torna um incômodo para o usuário que apenas deseja executar a aplicação e espera que ela funcione conforme esperado.

O exemplo JXPlayer possibilitou unir os conceitos aprendidos sobre o JavaFX e a parte prática. Com a utilização das ferramentas de apoio ao desenvolvimento foi possível criar uma aplicação rica em efeitos gráficos, com suporte a mídia e portável. Este estudo permite concluir que apesar de as ferramentas de produtividade ajudarem no desenvolvimento da aplicação, o *plugin* JavaFX Composer, ainda é imaturo para o desenvolvimento de aplicações mais complexas, pois apresenta instabilidade e carece de implementação de recursos. No entanto, conclui-se que a linguagem utilizada no projeto, JavaFX Script, é de fácil aprendizado, permite acesso a funcionalidades da plataforma Java e oferece recursos técnicos muito atrativos para desenvolvedores interessados em uma linguagem dinâmica.

O JavaFX é uma tecnologia emergente e este estudo apresentou as características técnicas plataforma, além de demonstrar o nível de complexidade no desenvolvimento do aplicativo JXPlayer utilizando as ferramentas de produtividade disponíveis. Conclui-se que o JavaFX agrega novas perspectivas no desenvolvimento de aplicações RIA e, ao mesmo tempo, permite que o legado da plataforma Java possa ser utilizado. É uma tecnologia promissora, mas por ser recente, ainda necessita de melhores IDEs e maior número de componentes pré-fabricados.

Este estudo possibilitou efetuar comparativos com as tecnologias de propósitos semelhantes: Adobe AIR e Silverlight. Isso foi realizado de forma a identificar vantagens de cada plataforma. Com a experiência obtida no

desenvolvimento do JXPlayer, conclui-se que o JavaFX destaca-se na portabilidade das aplicações e integração com a plataforma Java, mais ainda carece de ferramentas de apoio ao desenvolvimento mais bem estabelecidas no mercado. Já o Adobe AIR oferece uma plataforma mais integrada e com sistemas de software de apoio mais elaboradas e concisas, mas que necessitam de licenciamento para uso. E por fim, o Silverlight destaca-se por permitir a utilização de diversas linguagens de programação na criação de aplicações RIA, porém a tecnologia é restrita a poucos sistemas operacionais.

REFERÊNCIAS

ADOBE. **Deliver Rich Internet Applications on the desktop.** Disponível em: <http://www.adobe.com/uk/products/air/pdfs/air_flex_datasheet.pdf>. Acesso em: 01 jun. 2010.

ADOBE. **Adobe Flash Player.** Disponível em: <http://www.adobe.com/br/products/flashplayer/>. Acesso em: 04 jun. 2010.

ADOBE. **Perguntas frequentes.** Disponível em: <http://www.adobe.com/br/products/air/faq/>. Acesso em: 01 jun. 2010.

ALLAIRE, Jeremy. **Macromedia Flash MX:** A next-generation rich client. Disponível em: <http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf>. Acesso em: 02 mar. 2010.

ALVES, Alan. **Aplicações web ricas e acessíveis**. Disponível em: <http://www.slideshare.net/synergiaufmg/aplicaes-web-ricas-e-acessveis-3702594>. Acesso em: 05 mar. 2010.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. **Requisitos Ergonômicos para Trabalho de Escritórios com Computadores.** Disponível em: <http://www.abntcatalogo.com.br/norma.aspx?ID=1825>. Acesso em: 03 mar. 2010.

BALL, Tom. **Elephants and the JavaFX Script Compiler.** Disponível em: <http://weblogs.java.net/blog/2007/11/10/elephants-and-javafx-script-compiler>. Acesso em: 10 jun. 2010.

BAR, Terrence. **JavaFX 1.2:** Technical Introduction. Disponível em: <http://www.slideshare.net/terrencebarr/javafx-bringing-rich-internet-applications>. Acesso em: 04 mar. 2010.

CHAMBERS, Mike. et al. **Adobe Integrated Runtime (AIR) for JavaScript Developers Pocket Guide.** Editora O'Reilly., 2007.

CHEN, Doris. **JavaFX:** The Plataform for Rich Internet Applications. Disponível em: <http://www.devnexus.com/static/2009/presentations/Keynote_JavaFX_Doris.pdf>. Acesso em: 03 mar. 2010.

DURÃES, Ramon. **Introdução ao Microsoft Silverlight.** Disponível em: <http://imasters.uol.com.br/artigo/6095/aspnet/introducao_ao_microsoft_silverlight>. Acesso em: 04 jun. 2010.

ELIS, Diego. **O que é Tableless**. Disponível em: <http://www.tableless.com.br/o-que-etableless>. Acesso em: 11 jun. 2010.

GILES, Jonathan. **Internal JavaFX Activity.** Disponível em: <http://nick-software.blogspot.com/2010/03/internal-javafx-activity.html>. Acesso em: 04 mar. 2010.

PETERS, Greg. **E a Internet reinventou os negócios.** Disponível em: <http://www.strategia.com.br/Arquivos/E_a_internet_reinventou.pdf>. Acesso em: 04 mar. 2010.

HORN, Shannon. **Microsoft Silverlight 3:** A Beginner's Guide. Editora McGraw-Hill Osborne., 2009.

JENSEN, Jens. F. **Interactivity:** Tracing a new concept in media and communication studies. Disponível em: <http://www.organicode.net/jenson.pdf>. Acesso em: 04 mar. 2010.

MICROSOFT. **Comece a criar uma experiência mais profunda na Web.** Disponível em: <http://msdn.microsoft.com/pt-br/magazine/cc163404.aspx>. Acesso em: 03 jun. 2010.

MICROSOFT. **Features Matrix.** Disponível em: <http://www.silverlight.net/getstarted/overview.aspx>. Acesso em: 03 jun. 2010.

MICROSOFT. **Get started with Silverlight.** Disponível em: <http://www.silverlight.net/getstarted>. Acesso em: 02 jun. 2010.

MICROSOFT. **Silverlight Architecture.** Disponível em: <http://msdn.microsoft.com/en-us/library/bb404713(v=VS.95).aspx>. Acesso em: 02 jun. 2010.

NOVELL. **Moonlight**. Disponível em: <http://www.mono-project.com/Moonlight>. Acesso em: 08 jun. 2010.

PAMPLONA, Vitor. **Web Services via J2SE e J2ME.** Disponível em: <http://imasters.uol.com.br/artigo/2741/java/web_services_via_j2se_e_j2me>. Acesso em: 08 jun. 2010.

REHEM, Serge; TELEMACO, Ulisses. **Applets:** A nova geração de plugins Java. Disponível em: <http://www.slideshare.net/serge_rehem/javabahia-applets-novojavaplugin-3797961>. Acesso em: 11 jun. 2010.

SUN MICROSYSTENS. **JavaFX Composer.** Disponível em: <http://wiki.netbeans.org/JavaFXComposer>. Acesso em: 21 maio. 2010.

SUN MICROSYSTENS. **JavaFX Script:** Overview**.** Disponível em: <http://www.sun.com/software/javafx/script/>. Acesso em: 13 mar. 2010.

SUN MICROSYSTENS. **Perguntas gerais.** Disponível em: <http://javafx.com/pt_BR/faq/#1.1>. Acesso em: 10 mar. 2010.

SUN MICROSYSTENS. **Saiba mais sobre a tecnologia Java.** Disponível em: <http://www.java.com/pt_BR/about>. Acesso em: 11 jun. 2010.

WEAVER, James L. *et al*. **Plataforma Pro JavaFX:** Desenvolvimento de RIA para dispositivos móveis e para área de trabalho por scripts com a tecnologia Java. Rio de Janeiro: Editora Ciência Moderna Ltda., 2010.