INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA SUL-RIO-GRANDENSE - IFSUL, *CAMPUS* PASSO FUNDO
CURSO DE TECNOLOGIA EM SISTEMAS PARA INTERNET

DANIEL SANTINI

DESENVOLVIMENTO DE APLICAÇÕES RIA COM JAVAFX

Jorge Luis Boeira Bavaresco

PASSO FUNDO, 2014

DANIEL SANTINI

DESENVOLVIMENTO DE APLICAÇÕES RIA COM JAVAFX

Monografia apresentada ao Curso de Tecnologia em Sistemas para Internet do Instituto Federal Sul-Rio-Grandense, *Campus* Passo Fundo, como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet.

Orientador: Jorge Luis Boeira Bavaresco

PASSO FUNDO, 2014

DANIEL SANTINI

DESENVOLVIMENTO DE APLICAÇÕES RIA COM JAVAFX

Trabalho de Conclusão de Curso aprovado em ____/____/____ como requisito parcial para a obtenção do título de Tecnólogo em Sistemas para Internet

Banca Examinadora:

_____

Jorge Luis Boeira Bavaresco

_____

Roberto Wiest

_____

Carmen Vera Scorsatto

_____

Alexandre Tagliari Lazzaretti

PASSO FUNDO, 2014

Aos meus pais, irmão e amigos
pela compreensão e incentivo
em todos os momentos.

AGRADECIMENTOS

Primeiramente queria agradecer a Deus, por ter me iluminado e dado forças para concluir a mais esta etapa da minha vida.

Aos meus pais, Renato e Zenara, e ao meu irmão, Gabriel, por toda a ajuda, apoio e amor que me dedicaram e por sempre estarem sempre me apoiando e incentivando para que eu alcance meus objetivos.

Ao meu orientador, Prof. Jorge Luis Boeira Bavaresco, pela excelente orientação, por ser sempre atencioso e estar sempre disposto a ajudar e pelos ensinamentos e auxilio na construção desse projeto.

Ao Prof. Evandro Miguel Kuszera, pela orientação no Projeto de Conclusão I, por ter auxiliado na escolha do meu trabalho e por ter sido sempre prestativo quando precisei.

Aos meus colegas Alisson, Marina e Luana, os quais me acompanharam desde o início do curso, por todos os momentos, bons ou ruins, que passamos e compartilhamos juntos.

Por fim queria agradecer todos os familiares e amigos pela compreensão nos momentos difíceis e que sempre se demonstraram presentes na minha caminhada e aos professores do IFSUL, por todos os ensinamentos e conhecimentos passados nesse período.

“A persistência é o caminho do êxito”

Charles Chaplin

RESUMO

Este trabalho apresenta um estudo sobre o desenvolvimento de aplicações RIA com JavaFX, buscando avaliar as principais características da tecnologia, através da implementação de um estudos de caso e comparação com outras tecnologias semelhantes. Este estudo de caso, baseou-se na implementação de um sistema de controle de substituições de professores atualmente utilizado no IFSUL e a partir dele foi efetuada uma avaliação utilizando testes em diferentes sistemas operacionais, buscando responder questões relacionadas ao suporte multiplataforma do JavaFX.

Palavras-chave: Aplicações RIA; JavaFX; Suporte Multiplataforma.

ABSTRACT

This paper presents a study about the development of RIA applications with JavaFX aiming to assess the main characteristics of the technology, through the implementation of a case study and comparing it to other similar applications. This case study was based in the implementation of a system of teachers substitution's control currently used at IFSUL, and from it an evaluation was made using tests in different operational systems, trying to answer questions related to the multiplatform support of JavaFX.

Keywords: RIA Applications; JavaFX; Multiplatform Support.

LISTA DE TABELAS

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

Com a popularização da Internet, a Web está se tornando referência no desenvolvimento de aplicações, está surgindo uma tendência de que as aplicações, antes executadas isoladamente em um desktop, migrem para a Internet. As aplicações Web são projetadas para a sua utilização através de um navegador, na Internet ou em redes privadas (Intranet), aproveitando os benefícios da conexão à rede.

As aplicações desktops, por serem instaladas na maquina cliente e não necessitarem de uma comunicação constante com a Internet proporcionam uma melhor interatividade ao usuário, porém, a instalação e manutenção são umas das desvantagens das aplicações desktop em relação as aplicações Web.

Com o avanço da tecnologia, novas ferramentas foram surgindo, em que foi sendo priorizado o uso destes recursos nas aplicações web. O JavaFX surgiu como uma das soluções para a criação de aplicações ricas de internet.

A plataforma JavaFX é uma biblioteca Java que dispõe de vários recursos para a criação de aplicações ricas de internet, que permite criar aplicativos para diversas plataformas. Ela passou por uma evolução desde o lançamento da sua versão 2.0, onde foi substituída a linguagem de programação denominada JavaFX Script pela linguagem Java nativa.

Em princípio, o JavaFX possibilita que se execute uma aplicação de diversas formas, a partir do mesmo código fonte. É uma tecnologia nova, não possui muita documentação e possui muitas funcionalidades que podem ser analisadas, diante destes fatos, baseado nisso, este trabalho propõe um estudo sobre o nível de maturidade do suporte multiplataforma do JavaFX, considerando que a aplicação utilizada como estudo de caso será executada via desktop e via web em diferentes navegadores e sistemas operacionais. Este trabalho também aborda comparações baseada em vários critérios entre tecnologias semelhantes.

O trabalho apresenta em seu segundo capítulo uma introdução às abordagens e conceitos relacionados ao desenvolvimento RIA e um estudo sobre as principais tecnologias que realizam esse tipo de desenvolvimento. Em seu terceiro capítulo, aborda o desenvolvimento da aplicação que será utilizada como estudo de caso da tecnologia JavaFX e também uma comparação com a tecnologia Java Swing. No quarto capítulo serão apresentados os testes realizados com a aplicação desenvolvida e exibidos os resultados da avaliação multiplataforma realizado, demonstrando a real eficiência da tecnologia avaliada.

Por fim, o quinto capítulo apresenta as considerações finais do trabalho, o qual é seguido das referências.

1.1 MOTIVAÇÃO

O aumento significativo da utilização da Internet está fazendo com que a Web se torne o principal canal de comunicação. Desde então, os softwares começam a seguir a tendência de migrar para esta plataforma, que elimina vários obstáculos, como a disponibilidade da aplicação e possibilita uma maior independência de computador e sistema operacional. Porém ainda não se pode afirmar que a Web substitui completamente o desktop no desenvolvimento de aplicações, pois sua usabilidade é menor. Com base nisso, a tecnologia JavaFX tenta unir a dinamicidade da Web e o poder do Desktop.

A partir disso, a motivação desse estudo consiste em uma análise na área de desenvolvimento RIA, procurando através de um estudo de caso, fazer comparações com tecnologias semelhantes e uma avaliação sobre a consistência do JavaFX no desenvolvimento multiplataforma.

Como contribuição, esse trabalho visa fornecer informações que podem ser utilizadas, para introduzir o uso de tecnologias de desenvolvimento RIA, visando principalmente na tecnologia JavaFX, que é uma tecnologia considerada nova e ainda com vários pontos a serem estudados.

1.2 OBJETIVOS

1.2.1 Objetivo Geral

Estudar e avaliar o uso do JavaFX no desenvolvimento de aplicações RIA, buscando identificar as principais vantagens e desvantagens da tecnologia através da comparação com abordagens semelhantes e implementação de estudos de caso.

1.2.2 Objetivos específicos

- Estudar tecnologias para desenvolvimento de aplicações RIA.
- Aprofundar o estudo sobre a tecnologia JavaFX;
- Realizar comparações entre o JavaFX e abordagens semelhantes;
- Desenvolver uma aplicação de estudo de caso para avaliar as características do JavaFX;

2 REFERENCIAL TEÓRICO

Esta seção tem a finalidade de apresentar os principais conceitos sobre o desenvolvimento de aplicações RIAs e fornecer o embasamento teórico necessário para a compreensão do presente trabalho.

2.1 APLICAÇÕES RIA

Nos últimos anos, a Web tem se tornado a plataforma de referência para o desenvolvimento de softwares, está havendo um crescimento considerável na migração de softwares que antes eram executados isolados no Desktop para a Web. Devido a esse crescimento, está tendo a necessidade da criação de aplicações cada vez mais sofisticadas, com um nível de complexidade mais elevado, porém as tecnologias atuais estão se mostrando mais limitadas no que diz respeito à usabilidade e à interatividade no software (BOZZON et al., 2006).

A experiência do usuário em aplicações Web não é a mesma comparada ao desktop, na Web as aplicações possuem interfaces mais limitadas, já no desktop os usuários possuem uma maior liberdade de interação com o sistema. Desde então, surgiu a necessidade de unir os conceitos Web e desktop em um mesmo ambiente, de forma que fosse possível realizar, nas aplicações Web, as mesmas ações que eram possíveis nas aplicações desktop e ainda tendo a vantagem da conexão à rede. Isso se tornou possível após a criação de uma nova categoria de aplicativos, a dos aplicativos RIA (QUERINO FILHO, 2012).

O termo RIA foi introduzido pela Macromedia, em março de 2002, fazendo menção à unificação de aplicações desktops com aplicações Web, visando alavancar as vantagens e superar as desvantagens de ambas as arquiteturas (BOZZON et al., 2006).

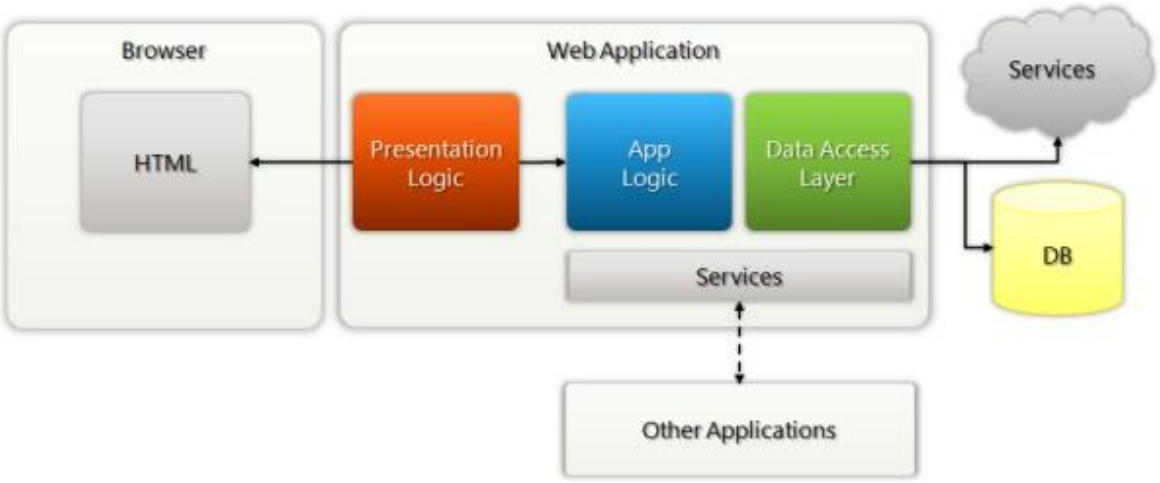
Assim como umas das características das aplicações desktops, as aplicações RIA oferecem uma interface gráfica bastante interativa, uma variedade de componentes para a personalização da interface do usuário e possibilitam que usuário realize operações tradicionais de aplicações desktop, elas são baixadas e executadas automaticamente em um navegador, desta forma ela pode ser acessada independente do lugar e da plataforma que se esteja utilizando (APRESENTANDO, s.d.).

As aplicações RIA provêm interfaces sofisticadas para representar processos e dados complexos, minimizando a transferência de dados entre cliente e servidor. Elas devem

suportar o processamento tanto no servidor como no cliente, para poder reduzir ao máximo a comunicação com o servidor (BOZZON et al., 2006).

Em uma aplicação Web tradicional, tem-se um cenário que tanto a camada de apresentação quanto à lógica da aplicação estão armazenadas no servidor, como se pode observar na Figura 1.

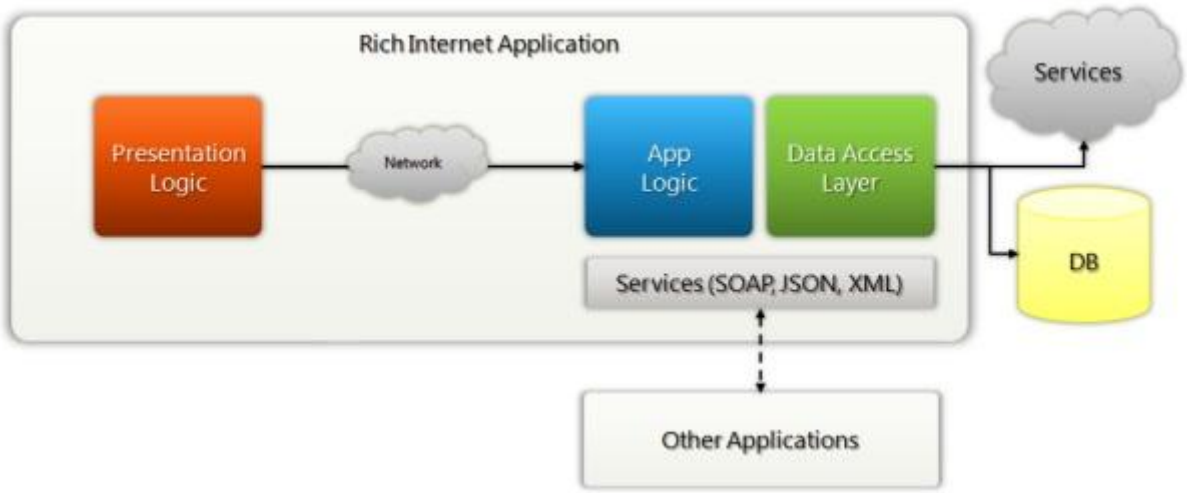
Figura 1 – Arquitetura das aplicações Web tradicionais



Fonte: Macoratti.net, 2013.

O que distingue, como se pode observar na Figura 2, a arquitetura das aplicações RIA das aplicações Web tradicionais é o fato de que as aplicações RIA utilizam mecanismos de renderização mais robustos no lado do cliente, a parte correspondente ao cliente contém toda a parte gráfica da aplicação, desta maneira há um equilíbrio entre o processamento que ocorre no servidor e o que ocorre no cliente, pois, sem contar o suporte inicial em que o servidor envia o interface para o cliente, o servidor fica responsável apenas pelo processamento da lógica do negócio e não se encarrega mais de toda a requisição remota à interface para os dados processados. Desta forma, as aplicações RIA proporcionam uma melhor resposta, possibilitando mais agilidade na interação entre a aplicação e o usuário, reduzindo o volume de dados transmitidos. Geralmente, as aplicações RIA reagem mais rápido do que as aplicações Web tradicionais (ROCHA, 2010).

Figura 2 – Arquitetura das aplicações RIA



Fonte: Macoratti.net, 2013.

A partir da Tabela 1, apresentada por Bozzon et al. (2006), pode-se observar que as aplicações RIA são formadas pela combinação das funcionalidades das aplicações desktops com as da Web.

Tabela 1 - Comparação entre aplicações desktop, Web e RIA

Funcionalidade	**Desktop**	**Web**	**RIA**
Ciente Universal (Browser)	Não	Sim	Sim
Instalação do cliente	Complexo	Simples	Simples
Capacidade de interação	Rica	Limitada	Rica
Lógica do negócio no lado do servidor	Sim	Sim	Sim
Lógica do negócio no lado do cliente	Sim	Limitada	Sim
Necessidade de recarregamento de toda a página	Não	Sim	Não
Comunicação servidor-cliente	Sim	Não	Sim
Funcionamento off-line	Sim	Não	Sim

Fonte: BOZZON et al., 2006.

As principais tecnologias de desenvolvimento de aplicações RIA atualmente são: JavaFX, Adobe Flex, Silverlight e HTML5 que serão apresentadas a seguir.

2.1.1 JavaFX

O JavaFX é uma tecnologia criada pela Sun Microsystems, tendo como princípio ser uma ferramenta que propõe o desenvolvimento de aplicações ricas para a Internet (RIAs), ela busca uma plataforma-cliente contando com um conjunto de gráficos e pacotes de mídia para a criação e a geração de experiências avançadas de internet em todos os dispositivos utilizados, isto é, foi projetada para suportar uma variedade de plataformas e dispositivos visando a reutilização de código, levando em consideração sempre o slogan do Java: "*Write once run anywhere*", em português: "Escreva uma vez, execute em qualquer lugar."

A primeira versão do JavaFX foi lançada oficialmente em dezembro de 2008, sendo liberado o release 1.0. Inicialmente foi lançado para os sistemas operacionais Windows e Mac OS X. Foi criada uma linguagem para trabalhar com os recursos do JavaFX, denominada JavaFX Script.

Em fevereiro de 2009, a plataforma ganhou mais consistência com o lançamento do release 1.1, que forneceu suporte à criação de aplicativos para dispositivos móveis e a correção de bugs da versão anterior.

Com o lançamento da versão 1.2, em junho de 2009, foi adicionado o suporte para as plataformas Linux e Solaris, além da adição de novas bibliotecas de interface, obtendo um aprimoramento no seu desempenho e correções de bugs da versão anterior.

Na versão 1.3, lançada em abril de 2010, a plataforma obteve um amadurecimento em relação aos releases anteriores, tendo boa parte dos seus problemas resolvidos. Também foram adicionados novos controles de interface, melhorias de layout, suporte a gráficos 3D e um emulador de TV que permitia compilar e testar aplicações com o novo perfil de TV.

Até este momento, o JavaFX utilizava a linguagem JavaFX Script, uma linguagem simples e intuitiva semelhante ao Java, porém possuía detalhes que não agradavam os desenvolvedores, por isso não estava obtendo um grande potencial e sendo esquecida pela maioria dos programadores. Mas, desde que a Oracle comprou a Sun Microsystems, o JavaFX passou por uma reformulação, onde o suporte ao JavaFX Script foi descontinuado, e com o lançamento da sua versão 2.0, em outubro de 2011, surgiu um novo conceito sobre a

plataforma. Agora o JavaFX é escrito totalmente em Java, a partir deste momento os desenvolvedores poderiam utilizar todo o seu conhecimento em Java para desenvolver em JavaFX sem a necessidade de aprender uma nova linguagem (ARAÚJO e GIONGO, 2012).

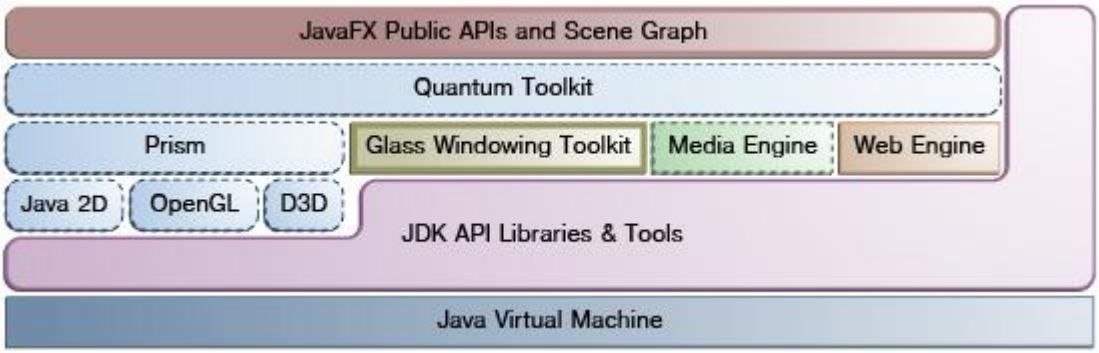
O JavaFX permite a construção de aplicações utilizando o modelo Model-view-controller (MVC), ou seja, as funcionalidades do sistema ficam separadas em camadas, a interface do usuário funciona totalmente separada da lógica da aplicação, o que facilita o desenvolvimento, pois o programa pode ser construído e testado em partes. Para isso, o JavaFX introduz uma linguagem de marcação denominada FXML como uma opção à codificação em Java, ela é uma linguagem baseada em XML usada para definir a interface com o usuário (ORACLE, 2013a).

Por padrão, o JavaFX disponibiliza três formas de execução das aplicações: a primeira utiliza o Java Web Start, onde é possível realizar o download do aplicativo e executá-lo no desktop, outra forma de execução é feita através de um navegador, onde o aplicativo é executado quando o navegador for iniciado, a aplicação interage com a rede sempre que necessitar. Na última forma, a aplicação é instalada diretamente no desktop e não há comunicação entre a aplicação e a internet (ARAÚJO e GIONGO, 2012).

2.1.1.1 Arquitetura

A arquitetura do JavaFX foi elaborada para abstrair a complexidade no tratamento de dispositivos com arquiteturas diferentes e proporcionar uma forma simples e intuitiva de desenvolvimento de aplicações, no nível mais baixo da arquitetura do JavaFX está a Java Virtual Machine (JVM), que é responsável pelo funcionamento multiplataforma do Java, ela faz com que a aplicação seja executada independente do Sistema Operacional que esteja sendo utilizado (LOPES e TAVARES, 2010).

Figura 3 - Arquitetura do JavaFX



Fonte: Oracle, 2013

Conforme pôde ser observado na Figura 3, o JavaFX possui uma arquitetura dividida em camadas bem definidas e cada uma possui uma determinada função na arquitetura de uma aplicação JavaFX.

a) Media Engine

Fornece suporte para os componentes visuais e de áudio e possibilita à aplicação um comportamento consistente em todas as plataformas executadas. As funcionalidades de mídia fornecidas pelo JavaFX são separadas em três componentes distintos: o objeto de mídia, que representa um arquivo de mídia; o MediaPlayer, que desempenha um arquivo de mídia; e uma MediaView, que é um nó que mostra a mídia (ORACLE, 2013b).

b) Web Engine

Possibilita que a interface do usuário e funcionalidades de navegação sejam visualizados e executados na Web, é baseado em WebKit, que é um motor de renderização utilizado em navegadores para renderizar páginas Web. Este componente incorporado no navegador é composto por duas classes: *WebEngine*, que fornece à página da Web a capacidade de navegação; e a *WebView*, a qual encapsula um objeto WebEngine e incorpora o conteúdo HTML em uma Scene de uma aplicativo (ORACLE, 2013b).

c) Glass Windowing Toolkit

É o nível mais baixo da pilha de gráficos JavaFX. Sua responsabilidade é de fornecer serviços de operações nativas, como o gerenciamento de janelas e temporizadores. Ele também é responsável pelo gerenciamento da fila de eventos, usa a funcionalidade fila de eventos nativo do sistema operacional para agendar o uso de

threads, ao contrário do Abstract Window Toolkit (AWT), que administra sua própria fila de eventos (ORACLE, 2013b).

d) JavaFX Public APIs and Scene Graph

O conjunto de APIs públicas do JavaFX define uma série de classes para simplificar a criação de interfaces ricas. Segundo a Oracle, "o Scene Graph é considerado o ponto de partida para a construção de um aplicativo JavaFX, é uma árvore hierárquica de nós que representa todos os elementos visuais de interface visuais do aplicativo"(2013b).

O JavaFX utiliza duas classes principais para formar a estrutura básica de uma aplicação: *Stage*, que é a classe que representa a janela do programa e pode ser considerada como um palco da aplicação, nela irá conter um *Scene*, que é a classe que possui um container para todo o conteúdo em um Scene Graph, ela fornece um container para os nós da aplicação, ou seja, os elementos da interface do usuário. Nessa estrutura, sempre irá conter um nó raiz denominado Root, que por sua vez conterá outros nós: com exceção dele, um único nó pode ter somente um nó pai e não ter ou ter muitos nós filhos, além disso, pode possuir várias propriedades, customizam seu comportamento e aparência como efeitos de sombra, opacidade, propriedades que manipulam eventos, entre outros (ARAÚJO e GIONGO, 2012).

O Scene Graph também inclui as primitivas gráficas, como retângulos e texto, controles de layout, imagens e mídia. É ele que é o principal responsável por simplificar o trabalho de criação de interfaces do usuário.

e) Sistema Gráfico

O sistema gráfico do JavaFX suporta tanto cenas 2D quanto 3D, ele possibilita um software de renderização quando o hardware gráfico do sistema é insuficiente para suportar a renderização gráfica. Segundo a Oracle (2013b), são implementados dois pipelines de aceleração gráfica:

- Prism – é ele quem processa os trabalhos de renderização. O Prism suporta uma ampla gama de GPUs disponíveis atualmente no mercado, como NIVDEA, ATI e Intel, possibilitando uma acelerada renderização gráfica em seu aplicativo. Ele possui paths de renderização que permitem a execução do aplicativo, independente do sistema que ele está sendo executado. Para isso, quando a aceleração por hardware está disponível, utiliza o DirectX em sistemas Windows, OpenGL no MAC e Linux, e Java2D quando a aceleração gráfica por hardware não está disponível, o que é um elemento importante, pois garante que a aplicação sempre

seja executada, porém o desempenho é melhor quando são usados paths de renderização por hardware, pois assim a construção de gráficos 3D é feita através do processador da placa gráfica e não usando os recursos da CPU.

- Quantum Toolkit – utilizado na vinculação entre o Prism e o Glass Windowing Toolkit e por torná-los disponíveis para a camada responsável pela criação da interface com o usuário. É responsável também por gerenciar as regras de threading relacionadas à renderização de eventos lançados.

2.1.1.2 Recursos necessários para executar aplicações

As aplicações JavaFX possuem suporte aos principais sistemas operacionais e navegadores atualmente. Para executar aplicações JavaFX, necessita-se ter ambiente em tempo de execução do Java (JRE) com a versão mais atual, pois o JavaFX RunTime é instalado acoplado com o JRE.

2.1.2 Adobe Flex

O Flex foi lançado pela empresa Macromedia em 2004, após a aquisição pela Adobe da empresa Macromedia em 2005 passou a ser chamado de Adobe Flex.

O Flex é um framework composto por uma série de componentes orientados ao desenvolvimento de aplicações RIA, permite a criação de aplicações para dispositivos móveis, para a Web e para o desktop usando o mesmo modelo de programação. É apresentado como uma das soluções para a problemática de desenvolvimento de interfaces ricas para aplicações Web atualmente (ADOBE, 2013).

O Flex utiliza a linguagem MXML para definir a interface da aplicação, ela é uma linguagem de marcação que possibilita a desvinculação da parte visual da aplicação com a parte lógica, é a partir dela que são construídos todos os elementos visuais. Além disso, ela pode ser utilizada para definir aspectos não visuais da aplicação, como o acesso aos dados do lado do servidor, e fazer ligações entre os componentes de interface do usuário (ADOBE, 2011).

Cada componente criado em MXML tem um comportamento padrão e estático, a linguagem MXML não especifica o que deve ser feito quando uma ação ocorrer sobre determinado componente. Para manipular os eventos, o Flex utiliza a linguagem *Action Script*, é uma linguagem orientada a objetos e procedural, tem a função de manipular todos os eventos que ocorrem na interface visual da aplicação (ADOBE, 2011).

O Flex é usado apenas na camada de visão da aplicação, o acesso aos dados e a interação com o servidor ocorre através do protocolo AMF (*Action Message Format*), que é responsável por oferecer o serviço de mensagens para a troca de dados entre um banco de dados e o aplicativo cliente. O servidor, por sua vez, pode ser construído com linguagens como: Ruby, Java, PHP, entre outras (ADOBE, 2011).

2.1.2.1 **Arquitetura**

A arquitetura de uma aplicação Flex é composta por uma árvore de componentes conhecida como lista de exibição, ela tem como sua raiz o *Stage*, que é considerado o palco da aplicação, onde estão dispostas todas as janelas de um aplicativo. Na camada abaixo está a classe *SystemManager*, que é a classe pai de todos os objetos visualizáveis na aplicação, ela é responsável por criar a classe que escuta todos os eventos nos componentes e adicionar o objeto da aplicação no *stage*. Na estrutura do Flex, todos os componentes visuais, incluindo controles e containers, são considerados nós folha da árvore e são subclasses da classe *DisplayObject* (ADOBE, 2011).

2.1.2.2 Recursos necessários para executar aplicações

O Flex possui suporte para os principais sistemas operacionais e navegadores atualmente. As aplicações desenvolvidas com o Flex são compiladas em um arquivo Flash, desta forma, para poder interpretar e executar aplicações Web é necessário ter um navegador atualizado, com um plugin[1] do *Flash Player*. Já nas aplicações desktop, o Flex requer o software *Adobe AIR runtime*, que é o ambiente em tempo de execução da Adobe, ele permite

[1] Plugin: Plugin é considerado todo programa, ferramenta ou extensão que se encaixa a outro programa principal para adicionar mais funções e recursos a ele.

que as aplicações Flex sejam executadas independente da plataforma utilizada, dispensando a necessidade de um navegador (ADOBE, 2013).

2.1.3 Silverlight

O Silverlight é uma tecnologia criada pela Microsoft, que permite a criação de aplicações ricas para a Web, desktop e Windows Phone. Ele é um plugin do Framework.NET, o que torna possível que as aplicações desenvolvidas em Silverlight sejam multiplataforma abrangendo um maior número de navegadores (MICROSOFT, 2013a).

O Silverlight é um subconjunto do WPF, que incorpora todas as funções do Framework.NET, permitindo uma maior usabilidade na interface do usuário nas aplicações Web. Porém, as aplicações Web desenvolvidas em WPF ficam limitadas ao sistema operacional Windows, apesar de rodarem em um navegador, funcionam restritas à plataforma da Microsoft, ou seja, só funcionam se estiverem sendo executadas no Windows e com o Framework.NET instalado (MICROSOFT, 2013a).

O Silverlight usa, para definir a interface com o usuário, uma linguagem declarativa denominada *Extensible Application Markup Language* (XAML), é a partir dela que são construídos todos os componentes visuais da aplicação, ela permite inicializar e definir as propriedades dos objetos presentes no Framework.NET. Ela também é usada para declarar estilos e modelos aplicados à base lógica dos controles da aplicação (MICROSOFT, 2010).

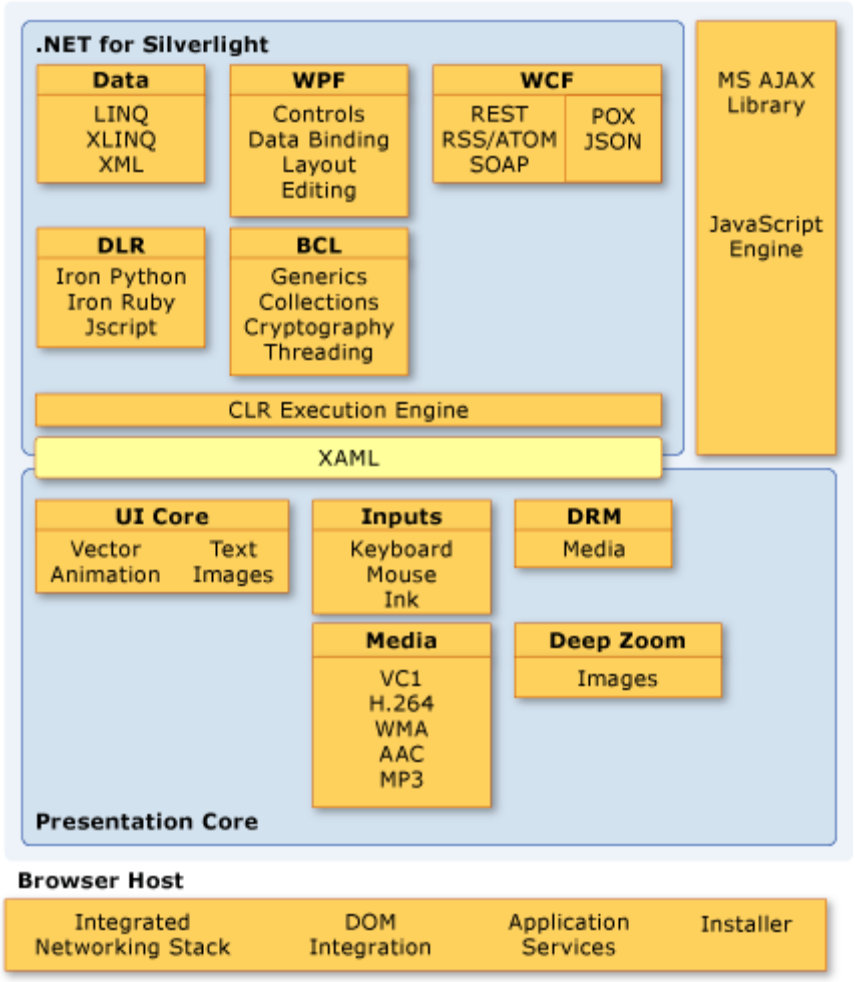
Com as marcações definidas em XAML, os elementos visuais contidos na interface do usuário ficam totalmente desvinculados da lógica da aplicação, que por sua vez pode ser escrita em Visual Basic ou C# (MICROSOFT, 2013a).

2.1.3.1 Arquitetura

A arquitetura do Silverlight, como se pode observar na Figura 4, é dividida em duas partes principais, além de um componente de instalação e atualização. O primeiro componente é o núcleo da apresentação do framework, que possui componentes e serviços orientados para a interface e a interação do usuário. O segundo componente é o

Framework.NET que contém bibliotecas e componentes da dão suporte à criação das aplicações.

Figura 4 - Arquitetura do Silverlight



Fonte: Microsoft, 2013.

O núcleo de apresentação da plataforma Silverlight é composto pelos seguintes componentes (MICROSOFT, 2013b):

- **UI Core**: É responsável por renderizar vetores gráficos, animações e textos;
- **Inputs**: É responsável por monitorar os dispositivos de hardware de entrada como teclado, mouse e outros dispositivos;
- **DRM**: Permite o gerenciamento digital de componentes de mídia;
- **Media**: É responsável pelo gerenciamento de vários tipos de arquivos de áudio e vídeo;
- **Deep Zoom**: Permite a utilização de zoom em imagens de alta resolução;

- **XAML**: Fornece um analisador a linguagem XAML. A linguagem XAML é o principal ponto de interação entre a camada de apresentação com o Framework.NET.

A camada do Framework.NET fornece um conjunto reutilizável de componentes, controles e elementos de interface do usuário que podem ser incorporados em aplicações escritas em Silverlight, ela possui os seguintes componentes (MICROSOFT, 2013b):

- **Data**: suporta o LINQ (*Language Integrated Query*) e LINQ-to-XML, os quais facilitam o processo de integração e o trabalho com fontes de dados diferentes. Também suporta o uso de XML e classes de serialização para manipular os dados;
- **WPF**: fornece os principais conjuntos de controles presentes do WPF para a criação da interface do usuário;
- **WCF**: *Windows Communication Foundation* é responsável por simplificar o acesso para serviços e dados remotos. Ele inclui objetos HTTP request, suporte para JSON, POX e serviços SOAP.
- **DLR**: *Dinamic Language RunTime* é responsável pela compilação e execução de linguagens de scripts como JavaScript e IronPython.
- **BCL**: *Base Class Library* é a biblioteca base do Framework.NET, que fornece as funções de programação essenciais para o controle da aplicação.
- **CLR**: Common Language RunTime é quem faz o gerenciamento da memória, verificações de segurança e controles de exceções.

2.1.3.2 Recursos necessários para executar aplicações

O Silverlight possui suporte para os principais sistemas operacionais e navegadores atualmente. Ele é um plugin de tamanho considerado pequeno que pode ser instalado nos navegadores. Através deste plugin, o Silverlight é considerado multiplataforma e não privado somente ao sistema operacional Windows como o WPF.

2.1.4 HTML5

O HTML (Hypertext Markup Language) é uma linguagem de hipertexto, desenvolvida pelo grupo de trabalho W3C (World Wide Web Consortium) que é responsável por manter o

padrão de código da linguagem. O HTML foi criado para ser uma linguagem independente de plataformas, browser e outros meios de acesso. Desta forma, a programação utilizando HTML significa uma redução de custos, pois, o código criado pode ser lido por diversos meios, ao invés de criar versões diferentes para diversos dispositivos (W3C, s.d.).

Quando o HTML4 foi lançado, foram definidas boas praticas a serem seguidas pelos desenvolvedores ao criar seus códigos. Mas, desde então, problemas como a separação da estrutura do código e princípios de acessibilidade foram sendo considerados. O HTML4 também não facilitava na manipulação dos elementos, ao criar um efeito gráfico sobre um elemento era necessário à criação de scripts longos, sujeitos a bugs e que poderiam não ser compatível com determinados navegadores (W3C, s.d.).

Enquanto o W3C focava o seu trabalho na segunda versão do XHTML, um grupo chamado WHATWG (*Web Hypertext Application Technology Working Group*) trabalhava em uma versão do HTML que buscava uma maior flexibilidade na criação de Web sites e sistemas baseados na Web. O WHATWG foi fundado por desenvolvedores de empresas como Mozilla, Opera e Apple em 2004. Eles não estavam contentes com os rumos que a Web estava tomando e por isso, estas organizações se juntaram para escrever o que hoje seria chamado de HTML5 (W3C, s.d.).

Por volta de 2006, o trabalho do grupo WHATWG passou a ser reconhecido mundialmente e principalmente pelo W3C, que reconheceu o trabalho do grupo e anunciou que trabalharia juntamente com o WHATWG na produção do HTML5.

O HTML5 surgiu da necessidade de encapsulamento dos recursos oferecidos pela Web, tendo como meta centraliza-los em uma única tecnologia, evitando assim que seja necessária a inserção de plug-ins nos navegadores (MENDES, 2011). Segundo Mendes (2011), o principal objetivo do HTML5 é facilitar a manipulação dos elementos modificando os objetos de maneira transparente ao usuário.

Antes do HTML5 não existia nenhum padrão universal de construção de elementos. No HTML5 foi definido um padrão de tags que permite marcar de forma padronizada as principais estruturas de uma pagina, como cabeçalho, rodapé, menus e etc. Esta padronização permite que se utilize a melhor estrutura de tags marcando de forma simples os principais pontos da pagina (W3C, s.d.).

O HTML utiliza como método padrão para construção de aplicações ricas o DOM (*document object model*) que é a interface entre a linguagem Javascript e os objetos HTML. A linguagem JavaScript é uma linguagem de objetos de script interpretada, que é responsável por adicionar dinamicidade as paginas HTML, ela permite operar objetos DOM e manipula-

los através da programação, obtendo métodos e propriedades para recuperar, modificar, atualizar e remover partes do documento HTML (GUISSET, 2011).

2.1.4.1 Arquitetura

A estrutura básica do HTML5 continua sendo praticamente a mesma das versões anteriores da linguagem como se pode observar na Figura 5:

Figura 5 – Estrutura do HTML5

```
<!DOCTYPE HTML>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" type="text/css" href="estilo.css">
    <title></title>
</head>
<body>

</body>
</html>
```

Fonte: W3C

A estrutura básica de um documento HMTL é formada pelos seguintes elementos (W3C, s.d):

- Doctype: O Doctype deve ser a primeira linha de código, ela indica para o navegador qual a especificação de código utilizar. Nessa linha ocorre a única mudança em relação às versões anteriores.
- HTML: O código HTML é composto por uma serie de elementos organizados em forma de árvore onde um elemento é filho de outro elemento e pode ter diversos filhos. O principal elemento dessa árvore é o elemento HTML.
- HEAD: A tag Head é onde fica as informações sobre a pagina e o conteúdo ali publicado.
- BODY: O elemento Body define o corpo do documento, é nele que contém todo o conteúdo de um documento HTML, como textos, hiperlinks, imagens, tabelas e etc.

2.1.4.2 Recursos necessários para executar aplicações

Um fator importante na execução das aplicações escritas em HTML5 é o fato de que elas não necessitam da instalação de nenhum plugin ou extensão nos navegadores para que a aplicação seja executada. Porém o principal obstáculo das aplicações criadas em HTML5 é a incompatibilidade com algumas versões dos navegadores atuais, pode ocorrer de que a aplicação funcione perfeitamente em um determinado navegador, mas em outro ela não se comporte corretamente. Pode ocorrer também o caso de um usuário que utilize um navegador desatualizado que não possui suporte ao HTML5, nesse caso a aplicação não irá funcionar.

A Tabela 2 apresenta a relação de compatibilidade entre os navegadores e alguns módulos do HTML5.

Tabela 2 - Tabela de compatibilidade entre navegadores e o HTML5

	Safari	Chrome	Opera	Firefox	IE8	IE9
Local Storage	S	S	S	S	S	S
Histórico de Sessão	S	S	S	S	S	S
Aplicações Offline	S	S	N	S	N	N
Novos tipos de campo	S	S	S	N	N	N
Form: Autofocus	S	S	S	N	N	N
Form: Autocomplete	N	N	S	N	N	N
Form: Required	S	S	S	N	N	N
Vídeo, Áudio e Camvas Text	S	S	S	S	N	S

Fonte: W3C

2.2 SWING

Quando o Java foi introduzido ele continha uma biblioteca de classes chamada AWT (Abstract Windows Toolkit), que era responsável por fornecer os elementos básicos para a programação da interface do usuário. O AWT é uma biblioteca gráfica completa que possui toda a implementação de janelas, tratamento de eventos, layouts, componentes gráficos como listas, botões etc. Utiliza chamadas nativas ao sistema gráfico do Sistema Operacional para desenhar os componentes de tela (HORSTMANN e CORNELL, 2010).

No entanto, essa biblioteca não proporcionava grandes recursos para fornecer aos usuários uma experiência consistente e ainda possuía problemas quando o aplicativo fosse executado em plataformas distintas.

Em 1996, a Netscape criou uma biblioteca para criação da interface chamada de IFC (Internet Foundation Classes), essa biblioteca possui um novo conjunto de recursos gráficos os quais tinham uma aparência e comportamento idênticos em todas as plataformas em que o programa executava. A Sun trabalhou com a Netscape para aperfeiçoar essa abordagem, criando uma biblioteca de interface com o usuário chamada Swing. O Swing não é considerado uma substituição completa da AWT, mas oferece mais recursos e componentes (HORSTMANN e CORNELL, 2010).

O Swing é a principal biblioteca para criação de aplicações desktop com Java, ele permite que as aplicações sejam executadas de duas maneiras: Standalone, ou seja, diretamente no desktop ou Java Web Start, na qual o browser é apenas um facilitador para instalar e iniciar as aplicações. O Swing trouxe para o Java uma biblioteca gráfica completa, com um conjunto de componentes para tornar as aplicações mais ricas e interativas. Permite criar sistemas multiplataforma, internacionalizáveis, com aparência nativa ou customizada, extensível ao ponto de se poder criar novos componentes e integrá-los à sua aplicação sem dificuldades, totalmente orientado a objetos, e que utiliza padrões de projeto em seu código-fonte.

A arquitetura do Swing baseia-se no MVC para a organização de seus componentes. Isso significa que cada componente possui classes que cuidam de sua visualização (*view*), de seus dados (*model*) e da interação entre um e outro (*controller*).

2.3 UML

Para se construir um projeto, por mais básico que ele seja sempre há necessidade de fazer um planejamento para ele seja bem sucedido. Para construir uma casa, antes de iniciar a construção é necessário um planejar, fazer a planta da casa, utilizar desenhos para representar como serão definidas as partes da casa e no desenvolvimento de software não é diferente disso, para que um sistema seja eficiente é necessário que tudo seja planejado e modelado de acordo com as suas especificações (BOOCH; RUMBAUGH, JACOBSON, 2012).

A UML (*Unified Modeling Language*) permite construir modelos que expliquem as características ou comportamentos de uma aplicação (NOGUEIRA, 2005). Construindo modelos podemos compreender melhor o sistema que estamos desenvolvendo, com a UML é

possível a visualização, a especificação, a construção e a documentação dos elementos que fazem parte de um sistema.

A modelagem permite compreensão de um sistema, os melhores modelos estão relacionados à realidade, mas nenhum modelo é inteiramente suficiente para representar um sistema. Quanto maior a variedade de modelos utilizados, melhor será a representação do sistema.

Os diagramas na UML são as formas que serão usadas para representar o sistema. Segundo (BOOCH; RUMBAUGH, JACOBSON, 2012, p. 28), "Um diagrama é a representação gráfica de um conjunto de elementos, geralmente representadas como gráficos de vértices (itens) e arcos (relacionamentos)". Um diagrama constitui uma projeção de um sistema, pois descreve vários aspectos da modelagem para a UML através das notações definidas pelos seus diagramas.

O diagrama casos de uso é um dos mais importantes na construção de softwares orientados a objetos, permite a modelagem de aspectos dinâmicos de sistemas. Ele é aplicado para fazer a modelagem da visão de casos de uso do sistema, permitindo visualizar, especificar e documentar o comportamento de um determinado elemento. Esse diagrama faz com que os sistemas, fiquem mais acessíveis e compreensíveis por detalhar uma visão externa do sistema (BOOCH; RUMBAUGH, JACOBSON, 2012).

O diagrama de atividades também é um modelo disponível na UML para a modelagem de aspectos dinâmicos de sistemas, é utilizado para criar descrições de casos de uso. Ele é baseado num gráfico de fluxo, mostrando a modelagem do fluxo de um objeto à medida que ele se passa de um estado para o outro (BOOCH; RUMBAUGH, JACOBSON, 2012).

O diagrama de classes são os mais frequentes nas modelagens de sistemas, são usados para fazer a modelagem da visão estática do sistema, mostrando um conjunto de classes, interfaces e atributos e seus relacionamentos (BOOCH; RUMBAUGH, JACOBSON, 2012).

2.4 JAVA PERSISTENCE API

A programação orientada a objetos difere muito do esquema entidade relacional e necessário pensar das duas maneiras para fazer um único sistema. Para representarmos as informações no banco de dados, utilizamos tabelas e colunas. Quando trabalhamos com uma aplicação Java, seguimos o paradigma orientado a objetos, onde representamos nossas informações por meio de classes e atributos. Além disso, podemos utilizar também herança,

composição para relacionar atributos, polimorfismo, entre outros. Esse buraco entre esses dois paradigmas gera bastante trabalho, pois, a todo o momento devemos transformar objetos em registros e registros em objetos (Apostila de Treinamento K19, 2012).

O Java Persistence API (JPA) fornece um modelo de persistência para o mapeamento objeto relacional a desenvolvedores para o gerenciamento de dados relacionais em aplicativos Java, ela permite descrever objetos de maneira declarativa, oferece linguagem de consulta e ferramentas para manipular entidades (ORACLE, 2006c).

JPA é apenas uma especificação, não um produto, ela não pode realizar a persistência ou qualquer outra coisa por si só. A JPA é apenas um conjunto de interfaces, e requer uma implementação. Há diversas implementações open-source da JPA, possibilitando uma liberdade de escolha do provedor, o que será usada neste trabalho é a do Hibernate.

O Hibernate é um framework de persistência Java open source que oferece persistência automatizada dos objetos em uma aplicação Java para um banco de dados relacional. Desta forma o mapeamento dos atributos entre uma base tradicional de dados relacionais e o modelo objeto de uma aplicação é feita pelo Hibernate, através do uso de arquivos (XML) ou anotações Java.

O Hibernate funciona como intermediário entre a aplicação e o banco de dados, ele é chamado de ferramenta ORM (Object Relational Mapping), considerada uma técnica de mapeamento de objeto relacional que permite fazer uma relação dos objetos com os dados que os mesmos representam (Apostila de Treinamento K19, 2012).

Desta forma, podemos dizer que o Hibernate facilita a comunicação entre a aplicação Java e o Banco de Dados, diminuindo a complexidade e oferecendo a opção de reaproveitamento de código.

3 ESTUDO DE CASO

Este capitulo visa apresentar os detalhes da arquitetura do projeto que será utilizado para estudo de caso. Inicialmente é apresentada uma descrição de como está atualmente o controle de substituições no IFSUL campus Passo Fundo, após são abordados os requisitos funcionais e não-funcionais, os diagramas de casos de uso e diagrama de classes, em seguida é apresentado os aspectos e funcionalidades das telas da aplicação. Por fim é feita uma breve comparação entre o swing e o JavaFX.

3.1 PROBLEMA

O objetivo da aplicação é controlar as substituições dos professores do IFSUL – Campus Passo Fundo. Uma substituição ocorre quando um professor não pode exercer suas atividades docentes por diversos motivos, como por exemplo, atestado de saúde, convocação judicial, viagens de trabalho, participação em congressos, seminários, etc. Atualmente, no campus Passo Fundo, o professor que por ventura necessitar de uma substituição deve preencher uma ficha com informações sobre suas atividades e quais serão os docentes responsáveis por aplicá-las. Esse documento deve ser enviado aos coordenadores de curso para ciência da substituição e devidos encaminhamentos, quando necessário.

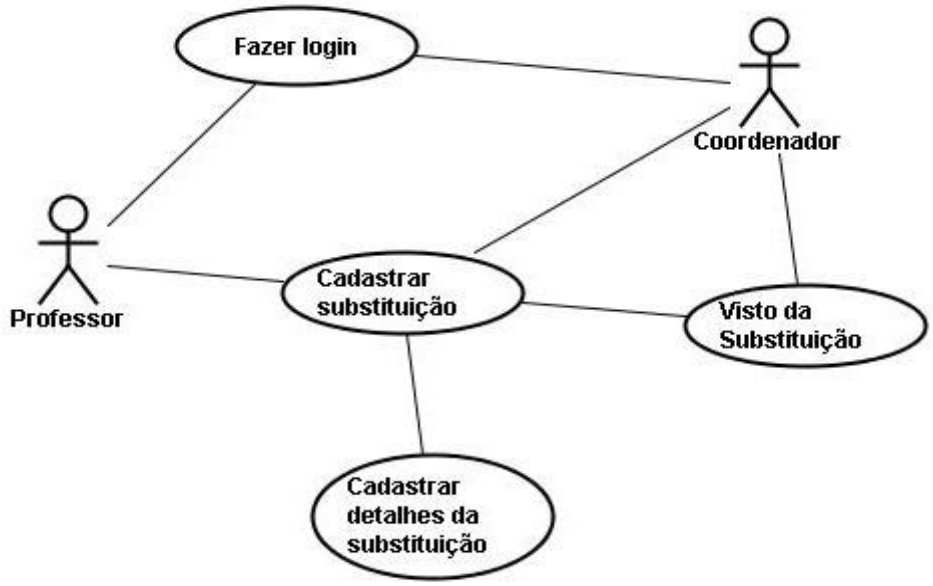
Para melhorar o fluxo de informações entre os coordenadores de curso, surgiu a necessidade de implementar um sistema para controlar as substituições, de forma que essas informação estejam disponíveis para consulta via Web.

3.2 REQUISITOS DO SISTEMA

O sistema de controle de substituições de professores para o IFSUL possui como funcionalidades o cadastro da substituição e os detalhes das atividades que devem ser aplicados pelo professor substituto. Para acessar o sistema o professor deve primeiramente efetuar o login no sistema. Toda substituição deve receber o parecer do coordenador de curso, que também precisa ter feito login no sistema para analisar a substituição e aprová-la ou não.

A Figura 6 representa o diagrama casos de uso da aplicação que foi construído a partir do documento de substituição atualmente utilizado no campus Passo Fundo.

Figura 6 – Diagrama Casos de Uso da aplicação



Fonte: Do Autor

Com base no diagrama de casos de uso proposto na Figura 6, pode-se observar os seguintes requisitos funcionais definidos para a aplicação, apresentados na Tabela 3.

Tabela 3 - Requisitos do sistema

Requisitos funcionais	
Manutenção dos professores	Permitir a inclusão, alteração e remoção de professores.
Manutenção das substituições	Permitir a inclusão, alteração e remoção de substituições.
Manutenção dos detalhes das substituições	Permitir a inclusão, alteração e remoção dos detalhes da substituição.
Manutenção dos cursos	Permitir a inclusão, alteração e remoção de cursos.
Controle de login para professores e coordenadores	Fazer um controle de login de professores e coordenadores, sendo que os professores devem ter um acesso mais restrito e os coordenadores acesso a todas as funcionalidades do sistema.
Análise de uma substituição feita por um coordenador	Permitir que um coordenador analise as substituições de todas os professores e aprove ou não a sua substituição.

Fonte: Do Autor

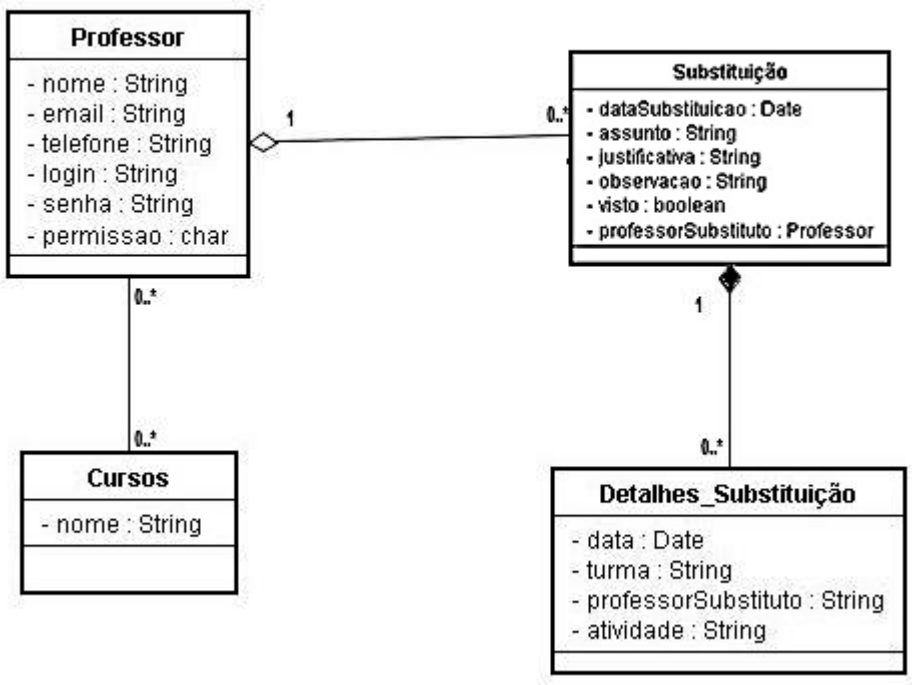
Tabela 4 - Requisitos não funcionais

Requisitos não funcionais	
Banco de dados	O banco de dados utilizado é o PostgreSQL.
Área restrita	Restringir para que apenas os coordenadores tenham acesso a todas as substituições.

Fonte: Do Autor

A partir da analise do documento de uma substituição também foi possível criar o diagrama de classes da aplicação, conforme a Figura 7. O diagrama de classes que também pertence à linguagem de modelagem UML, ele permite a visualização das classes que irão compor o sistema com seus respectivos atributos e relacionamentos entre as elas.

Figura 7 - Diagrama de classes da aplicação



Fonte: Do Autor

Esta aplicação tem potencial para avaliar as características da tecnologia JavaFX, elencadas neste projeto.

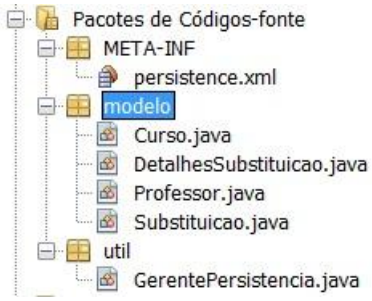
3.3 ARQUITETURA DA APLICAÇÃO

A partir da análise de requisitos pode-se perceber que o sistema é considerado simples, mas que pode ser explorado e adicionado novas funcionalidades conforme surgir necessidade, permitindo que futuramente o Instituto possa utiliza-lo. Na fase de implementação utilizou-se algumas bibliotecas e ferramentas que aceleraram o seu desenvolvimento.

Depois de analisado os diagramas, foi possível identificar as classes e atributos que irão compor a aplicação. O projeto foi desenvolvido com persistência, isto é, mantendo os dados salvos em um banco de dados, o servidor de dados escolhido foi o PostgresSQL, pois foi o servidor utilizado no curso. A tarefa de modelar o banco de dados ficou definida utilizando uma persistência automatizada utilizando como provedor de persistência o Hibernate.

A aplicação é construída com base no modelo MVC, com a completa separação das camadas, sendo todas elas bem definidas. Para descrever a camada de modelo da aplicação é criado um projeto Java, o qual é responsável apenas por ser o modelo do projeto e realizar as operações relacionadas ao banco de dados. A Figura 8 representa a estrutura de arquivos que compõem o projeto responsável pela camada de modelo.

Figura 8 – Estrutura de arquivos da camada de modelo

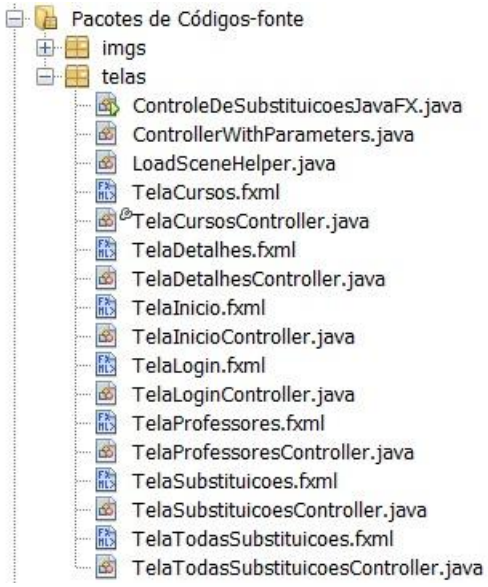


Fonte: Do Autor

Para as camadas de visão e de controle é construído um projeto JavaFX, o qual possui arquivos responsáveis por representar as funcionalidades da aplicação. No JavaFX, para criar uma tela o desenvolvedor possui duas opções: a primeira é criar em um arquivo Java e declarar os elementos gráficos instanciando objetos das classes requeridas, porem dessa forma a separação das camadas de visão e de controle fica mais confusa e os arquivos se tornam mais complexos; já a segunda forma é criar arquivos FXML, a partir deles é possível declarar os elementos como se fossem tags XML, para cada arquivo FXML quando criado, é

necessário especificar uma classe Java controladora, dessa forma são criados dois arquivos padrão, um que possui os elementos visuais que serão dispostos na tela e outro no qual deve ser programado todo o controle dos elementos. Pode-se observar a estrutura de arquivos da aplicação na figura abaixo.

Figura 9 – Estrutura de arquivos da camada de visão e controle



Fonte: Do Autor

3.4 JAVAFX E SWING

A necessidade de determinar o mais adequado no desenvolvimento de tecnologia e proporcionar uma melhor produtividade no desenvolvimento de aplicações modernas requer uma análise baseada em parâmetros o que facilita a comparação entre as duas tecnologias.

Conforme o estudo realizado nas tecnologias JavaFX e Swing, é possível destacar que ambas tecnologias são relacionadas à linguagem Java e como parâmetros de comparação é levado em consideração os aspectos do desenvolvimento de uma interface Desktop, suas dificuldades e facilidades e a aceitação do produto pelo mercado. Vale ressaltar que, apesar da possibilidade de utilização de JavaFX em aplicações Web, porém, nesse capitulo a ênfase será dada no desenvolvimento de uma aplicação Desktop.

Abaixo serão elencados alguns parâmetros, os quais permitem realizar uma comparação entre as tecnologias estudadas:

- **Comunidade de desenvolvedores**: A aceitação dentro da comunidade de desenvolvedores é um fator importante para determinar a maturidade da tecnologia

estudada. Considerando um cenário altamente conectado e a crescente utilização de fóruns de discussão para troca de experiências, tendem a oferecer uma base mais sólida para desenvolvedores iniciantes e também aos que já possuem alguma experiência.

O Swing, por ser uma tecnologia de desenvolvimento desktop com uma maturidade boa possui uma ótima movimentação nos blogs e fóruns de desenvolvimento em Java. No fórum oficial da Oracle o Swing possui milhares de mensagens em tópicos relacionados ao Swing, alem dos diversos fóruns brasileiros como o GUJ e o JavaFree que possuem muitos membros ativos que constantemente debatem sobre assuntos relacionados a tecnologia.

O JavaFX após a mudança para a versão 2.0 tem causado uma boa impressão aos desenvolvedores, tem feito aumentar o interessarem pelo JavaFX, tanto como alternativa ao Swing no desenvolvimento de aplicações desktop, como também para aplicações web. Existe um fórum oficial do JavaFX que disponibiliza conteúdo e muitos questões para debates entre os usuários. Já os fóruns brasileiros que não possuíam muita movimentação na versão antiga estão cada vez mais com mais membros e mais movimentados. A comunidade de desenvolvedores de terceiros tem mostrado um forte interesse em JavaFX, e uma série de soluções de terceiros estão atualmente disponíveis ou em desenvolvimento.

- **Suporte IDE**: A IDE Netbeans possui suporte nativo ao Swing, não é necessário a instalação de nenhum plug-in para iniciar o desenvolvimento podendo usar a possibilidade de arrastar os componentes para a janela. Já IDE Eclipse, para pode trabalhar com Swing e usar os componentes de maneira fácil como é feita no Netbeans, é necessário a instalação de um plug-in.

  Na IDE NetBeans, o JavaFX já está incluso, na mesma versão que inclui o Java EE. Desta forma, basta a instalação normal da interface para que o desenvolvedor já possa iniciar o desenvolvimento de sua aplicação. Para o desenvolvimento utilizando a API JavaFX, o Eclipse não possui uma versão para download que inclua suporte a esta tecnologia. O desenvolvedor que quiser utilizar esta IDE possui duas opções. Uma delas é baixar ou atualizar o kit de desenvolvimento do Java (JDK) na versão 7 ou superior, pois esta já possui o JavaFX integrado, e incluir a biblioteca dentro do projeto no Eclipse. Outra alternativa é utilizar um plug-in chamado e(fx)clipse, que é considerado uma ferramenta de terceiros. Este plug-in deve ser instalado e posteriormente deve-se informar ao Eclipse a pasta do JavaFX-SDK.

- **Documentação**: A documentação da tecnologia Swing pode ser acessada diretamente da página oficial da Oracle. La é encontrado toda a documentação das classes e métodos que compõe a hierarquia do Swing. Na seção destinada aos tutoriais é possível encontrar desde por onde começar com o Swing até conteúdos mais complexos, há também exemplos da utilização dos componentes disponíveis, como trabalhar com layouts e tratamento de eventos, por fim, uma documentação bem completa.

  A documentação do JavaFX também pode ser acessada diretamente da página oficial da Oracle, na seção destinada à tecnologia JavaFX. Dentro dela é possível visualizar uma lista completa de opções para aqueles que desejam se familiarizar com a tecnologia. As opções incluem vídeos demonstrativos que apresentam algumas características importantes da plataforma, links para artigos que abordam assuntos relacionados ao JavaFX como boas práticas de programação, uma visão geral apresentando uma linha do tempo da API e futuras funcionalidades, além do acesso ao conteúdo completo da documentação da API, demonstrando os pacotes, classes e métodos de todos os componentes do JavaFX. Além disso, o site do JavaFX apresenta, uma aplicação denominada JavaFX Ensemble, que é um mostruário de todos os componentes e funcionalidades da plataforma em sua versão 2.0. É possível visualizar exemplos de animações, efeitos 3D, componentes de aplicações como caixas de texto, além de uma extensa gama de gráficos animados. Todos os exemplos podem ser baixados como projetos da IDE NetBeans e podem ser usados como auxílio pelos desenvolvedores.
- **Aparência e componentes visuais**: O Swing é considerado uma evolução da biblioteca AWT, portanto possui quase todos os componentes que já existiam no AWT, porem com uma interface gráfica mais evoluída e com maiores efeitos. O Swing é composto por um conjunto de componentes com uma vasta capacidade de configuração e funcionalidades gráficas para construção de interfaces gráficas adicionando interatividade para aplicações desktop Java. Seus componentes são implementados inteiramente em Java, do qual permite criar interfaces que possuam a mesma aparência em diversos sistemas operacionais. Ao contrario do AWT, onde a renderização dos componentes era feita pelo sistema operacional o Swing renderiza seus componentes através próprio Java, fazendo com que eles tenham uma melhor performance. Conta também com o pacote Java3D que fornece conjunto de classes

para desenho 3D de gráficos, texto e imagem de alta qualidade e o JavaSound que permite controlar a entrada e saída dos meios de comunicação de som e áudio.

O JavaFX 2.0 é baseado em um paradigma de grafos de cena, em oposição à renderização estilo tradicional de modo imediato. Os grafos de cena do JavaFX são como uma estrutura de dados em árvore que mantém nós básicos baseados em vetor. Esta estrutura permite que a cena seja renderizada conforme um nó raiz ao qual os outros nós estão submetidos. Este conceito segue o modelo de criação de cenas de jogos, onde a cena toda é baseada em grafos de cena.

Utilizando este paradigma, os componentes JavaFX permitem, além da configuração CSS padrão, também a configuração de intensidades de luz e sombra, o que gera um efeito visual muito interessante. A abordagem baseada em grafos permite ao JavaFX a inclusão de animação em diversas partes da cena. As animações acontecem pela movimentação dos nós baseados em seu nó raiz. O JavaFX utiliza as características da placa gráfica do computador para gerenciar as modificações da cena, permitindo um melhor desempenho. Todas estas animações podem ser configuradas diretamente durante a construção da interface ou em resposta a algum evento. A lista de componentes do JavaFX apresenta, desde os básicos de interação via formulário, até os mais avançados, incluindo uma seleção de gráficos, todos permitindo interatividade, além de componentes 3D, players e controles de multimídia, além de várias opções de efeitos que permitem aperfeiçoar visualmente a interface.

- **Usabilidade**: A grande vantagem das aplicações desktop é justamente proporcionar um aumento significativo na usabilidade de sistemas e programas, difundindo seu uso e reduzindo a curva de aprendizado necessária para utilizá-los, sem restringir seu acesso pela necessidade de um conhecimento específico. O Swing, de uma forma geral, é uma API que agrega diversas funcionalidades ao desenvolvimento de interfaces baseadas em janelas. Possibilita o uso integrado de tecnologias para auxílio de portadores de necessidades especiais através de monitores e teclados adaptados aumentando sua acessibilidade.

  O JavaFX por ser um considerado uma tecnologia RIA, foi projetado para proporcionar ao usuário uma boa usabilidade tanto na web quanto no desktop, o desenvolvedor pode contar com muitos recursos gráficos para deixar sua aplicação com uma fácil navegabilidade. O Swing trabalha com o conceito de janelas, por exemplo, quando se precisa criar um cadastro é comum ser criado uma nova janela para realizar esse processo, já o JavaFX trabalha com o conceito de Stage, que

representa a janela principal do programa e Scene, que representa todo o conteúdo que será apresentado, dessa forma, em vez de abrir uma nova janela é apenas substituído a scene.

- **Portabilidade**: Por ser totalmente em Java, o Swing segue seu lema de ser escrito uma vez e ser executo em qualquer lugar, por isso, grande parte da complexidade das classes e métodos do Swing está no fato da API ter sido desenvolvida tendo em mente o máximo de portabilidade possível. Favorece-se, por exemplo, o posicionamento relativo de componentes, em detrimento do uso de posicionamento fixo, que poderia prejudicar usuários com resoluções de tela diferentes da prevista. Com Swing, não importa qual sistema operacional, qual resolução de tela, ou qual profundidade de cores: sua aplicação se comportará da mesma forma em todos os ambientes.

  O JavaFX esta totalmente integrado como um recurso do Java JRE e do Java Development Kit (JDK). O JDK está disponível para todas as principais plataformas de desktop (Windows, Mac OS X e Linux) e como o JavaFX a partir do Java 7 é considerado um recurso nativo do Java as aplicações desenvolvidas em com ele terão suporte aos sistemas operacionais citados.

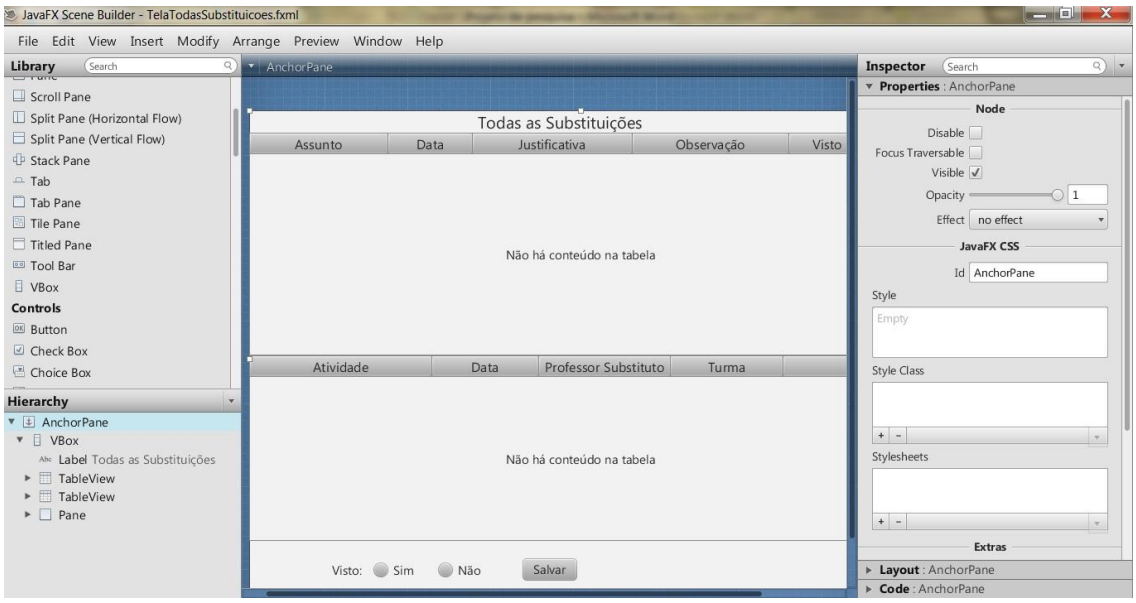
- **Aceitação pelo mercado**: O Swing é uma API Java mais madura, já esta no mercado há um bom tempo e consequentemente possui um vasto repertorio de documentação e também um número maior de usuários ativos e tópicos em fóruns. Existem diversas aplicações de grande porte construídas com Swing que estão ativas atualmente, por exemplo, o Netbeans e o Eclipse. Porem o Swing recentemente não está mais recebendo muitas novidades em atualizações, é uma tecnologia que esta sendo considerada estática no mercado e que leva ao desenvolvedor pensar se ainda é vantajoso desenvolver utilizando Swing.

  A tecnologia JavaFX ainda caminha entre as características apresentadas no início deste tópico. A plataforma passa ainda por um processo de afirmação dentro da comunidade Java, muito pelo fato de ter uma mudança drástica em sua linguagem de definição de interfaces a partir de sua versão 2.0. Para uma aplicação que é focada em RIA, esta mudança acabou causando uma quebra em sua evolução. Por outro lado, ela é uma tecnologia que pode ser considerada como uma boa aposta na evolução do Java como uma plataforma cliente rica, pois pode ser utilizada tanto para aplicações desktop quanto para web. O JavaFX é uma das alternativas para o uso do Swing e tem uma opinião bem dividida nos fóruns sobre o incentivo no desenvolvimento utilizando o mesmo.

- **Opções para desenvolvimento da interface**: No desenvolvimento de interfaces o Swing oferece a área de design, que é considerada à janela principal para criar e editar telas de interfaces gráficas do Java, nessa área é feita exibição gráfica dos componentes dispostos na tela que pode ser customizada utilizando uma lista de componentes denominada paleta, que contem guias para Swing e componentes awt, bem como gerentes de layout e todo conjunto de componentes. Utilizando a paleta é possível que a interface seja desenhada arrastando e soltando componentes na interface que será criada. Java Nativo: Apresentado como alternativa para a criação de interfaces com os recursos da paleta, o Java Nativo permite instanciar objetos que representem componentes na tela, através desses objetos é possível acessar e modificar seus atributos para modificar e personalizar os componentes.

  Para o desenvolvimento de interfaces utilizando a plataforma JavaFX, as tecnologias utilizadas são as seguintes: Linguagem FXML, que é uma linguagem de marcação baseada em XML para construção de grafos de objetos Java. Ele oferece uma alternativa conveniente para a construção de tais grafos em código processual, e é ideal para definir a interface de usuário de um aplicativo JavaFX, uma vez que a estrutura hierárquica de um documento XML possui uma estreita ligação com a estrutura do grafo de cena JavaFX. Dentro da estrutura FXML, os elementos podem representar a instância de uma classe, uma propriedade de uma classe ou até mesmo um bloco de código em script. As declarações seguem a estrutura de árvore de um arquivo XML comum. Java Nativo: Uma alternativa que a plataforma JavaFX oferece para a criação da interface é utilizar a própria linguagem Java. Este suporte permite que os objetos de tela sejam declarados e instanciados da mesma forma que se usa em uma aplicação Java comum. Todos os objetos possuem atributos que definem as suas configurações. Scene Builder: É uma aplicação externa fornecida pela Oracle que permite que a interface seja desenhada arrastando e soltando componentes dentro de uma área determinada, conforme se pode observar na Figura 10. Possibilita ao desenvolvedor a visualização da interface em tempo real de desenvolvimento, o que facilita muito principalmente em relação ao design da tela. É importante citar também que ambas as tecnologias apresentadas oferecem suporte a configuração por CSS. Incluindo um arquivo com as configurações determinadas, os componentes de tela assumem estas configurações conforme são determinadas dentro da programação das telas.

Figura 10 – JavaFX Scene Builder



Fonte: Do autor

- **Facilidade de desenvolvimento**:

A plataforma JavaFX oferece uma forma de desenvolvimento para web muito similar à forma utilizada em aplicações desktop. Dentre as vantagens verificadas, podem ser citadas: Ferramenta para desenho de tela: A ferramenta Scene Builder possibilita a criação das telas da aplicação por meio de uma interface gráfica, muito parecida com a modelagem de aplicações desktop. O usuário pode arrastar componentes para dentro de uma região que representa a área visível da tela e os ordenar conforme a sua vontade, com a possibilidade de alteração de cores, fontes, tamanhos, entre outras propriedades. Java nativo: Além do Scene Builder, a plataforma JavaFX oferece, a partir da versão 2.0, a possibilidade de criação da interface utilizando código da linguagem Java. Ao invés de utilizar as tags da linguagem FXML, o desenvolvedor que já possui experiência com Java, pode modelar a sua interface declarando objetos da mesma forma que utiliza em suas aplicações comuns. Essa característica possibilita um ganho de tempo com a familiarização de uma nova linguagem, porém o código da interface acaba se misturando ao código do controlador. Utilizando FXML, é possível realizar esta separação, muito importante para a organização do projeto.

Com base no estudo feito e a comparação sobre as tecnologias, podemos observar que ambas possuem um bom potencial para o desenvolvimento. Como as duas tecnologias são descendentes do Java é comum que apresentem características semelhantes, o swing

considerado uma tecnologia mais madura, possui mais visibilidade e atualmente uma maior utilização no mercado. Porem é possível notar que ao utilizar o JavaFX pode-se utilizar os mesmo recursos, funcionalidades e ainda tem a opção de tornar a aplicação com dois contexto sem alterar o código, ou seja, estender seu aplicativo que funcionava apenas no desktop para a web. No contexto da web, recursos HTML, CSS e JavaScript podem ser integrados em um aplicativo JavaFX. Embora o JavaFX ainda possua um futuro incerto, ele pode ser apontado como o substituto natural do Swing.

4 AVALIAÇÃO MULTIPLATAFORMA

Um grande problema para quem desenvolve sistemas Web é garantir que ele tenha o mesmo comportamento, independente do navegador que o usuário esteja utilizando. E programar isso pode ser considerado complexo, pois cada navegador tem características diferentes. O JavaFX foi projetado para funcionar nos principais sistemas operacionais de maneira efetiva e eficaz.

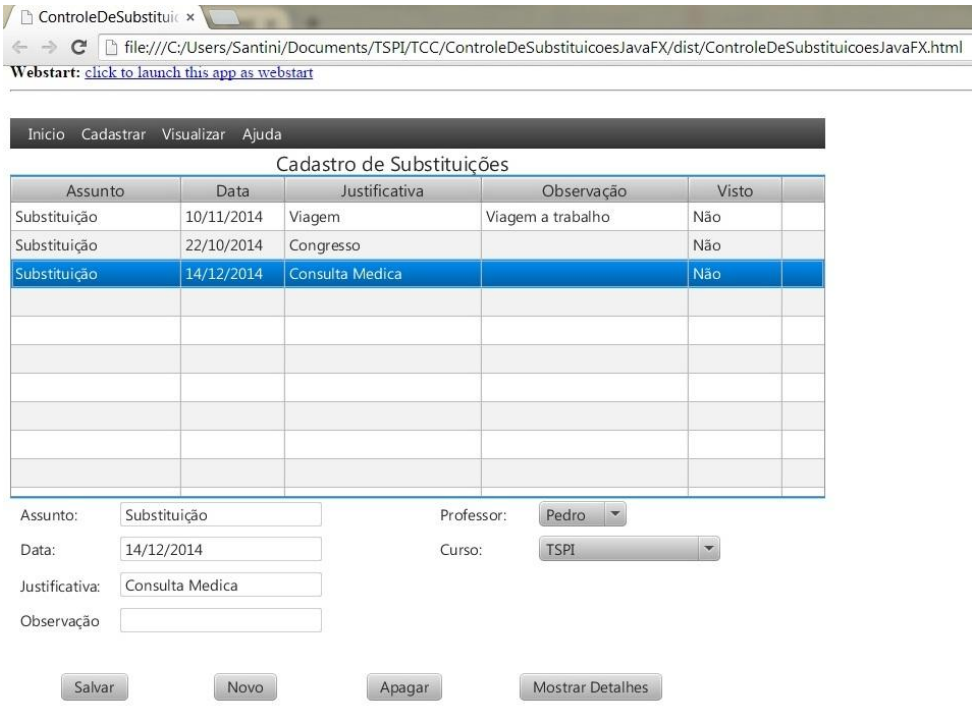
Este capitulo tem a finalidade de apresentar os testes multiplataforma realizado a partir da aplicação desenvolvida como caso de uso. Os testes serão realizados nos principais navegadores dos sistemas operacionais Windows e Linux, abordando os requisitos necessários para que a aplicação seja executada e se a aplicação se comporta de maneira consiste nos navegadores testados. A técnica usada foi configurar ambientes com múltiplas opções de software para testar a aplicação desenvolvida.

4.1 WINDOWS

No sistema operacional Windows a aplicação é avaliada nos navegadores Google Chrome, Mozilla Firefox, Opera e Internet Explorer, em cada um deles serão utilizados todas as funcionalidades desenvolvidas na aplicação, serão avaliadas questões relacionadas ao funcionamento aplicação, à usabilidade do programa e se em algum momento ele possa apresentar alguma falha ou diferença na forma de exibir o conteúdo.

- **Google Chrome:**

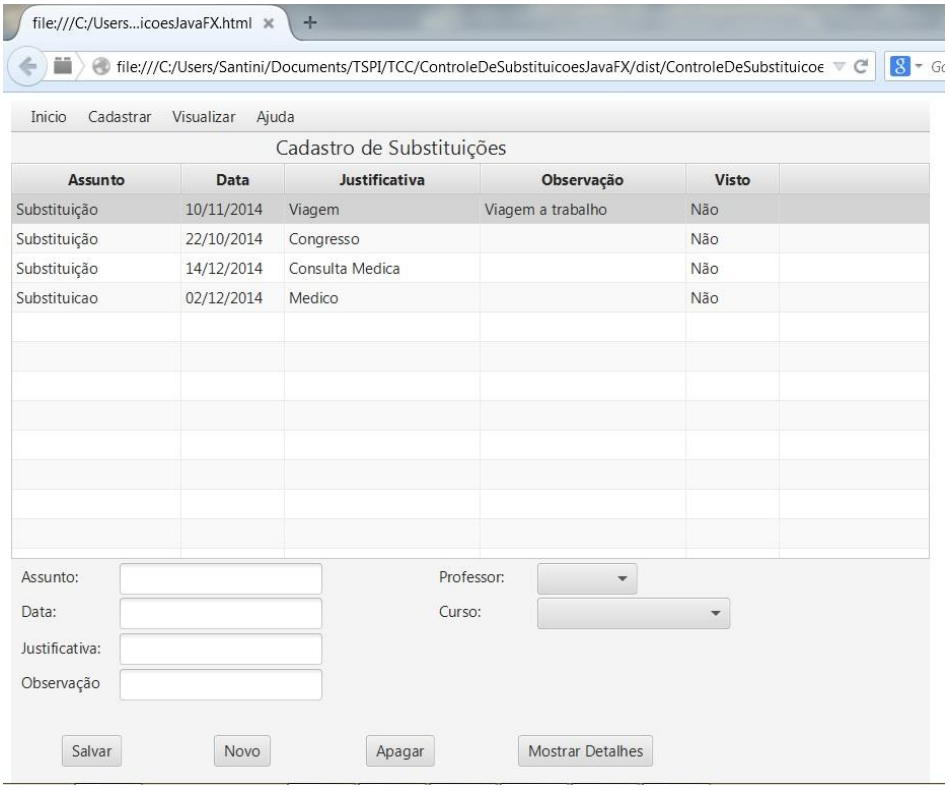
Figura 11 – Aplicação sendo executada no Google Chrome



Fonte: Do Autor

- **Mozilla Firefox:**

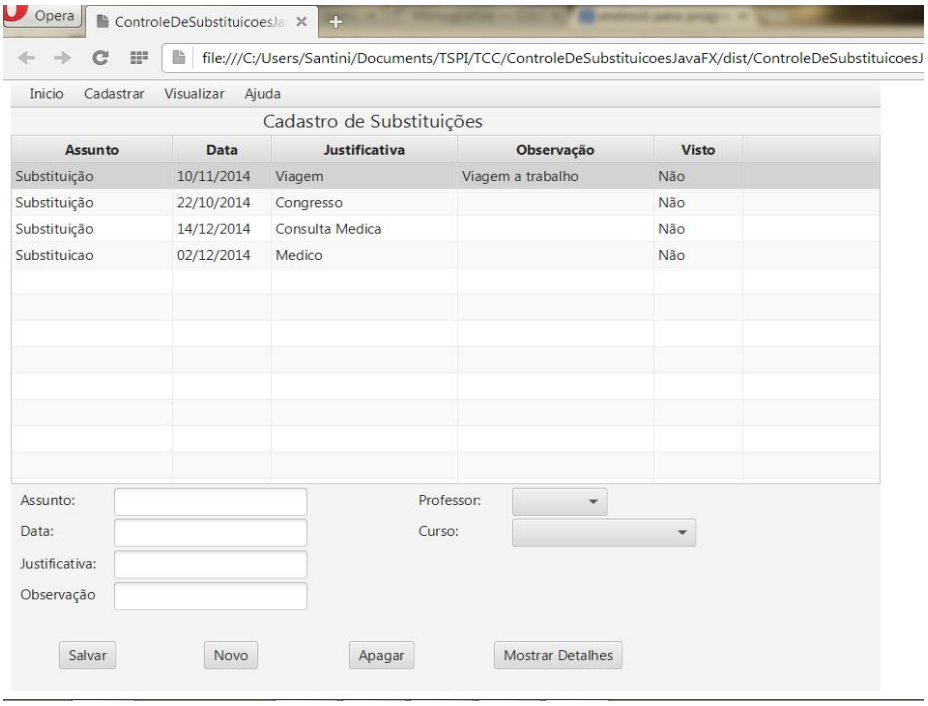
Figura 12 - Aplicação sendo executada no Mozilla Firefox



Fonte: Do Autor

- **Opera:**

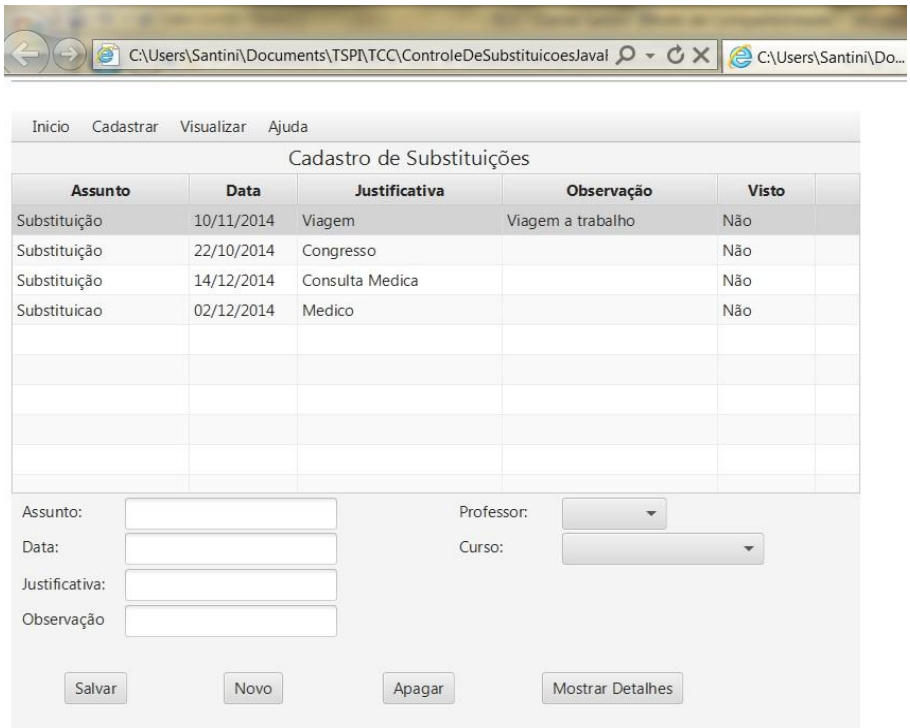
Figura 13 - Aplicação sendo executada no Opera



Fonte: Do Autor

- **Internet Explorer**:

Figura 14 - Aplicação sendo executada no Internet Explorer



Fonte: Do Autor

O primeiro ponto destacado é o fato de ser necessário o cliente possuir o *Java Runtime Environment* (JRE) instalado na sua maquina, por descender do Java é comum que isso aconteça, ao contrario a aplicação não será executada em nenhum ambiente testado. Para execução nos navegadores não é necessário nenhum plugin.

Na execução da aplicação diretamente no desktop, a qual foi usada no desenvolvimento do projeto em nenhum momento se mostrou inconsistente, desempenhando todas as operações com êxito.

Na execução via Web, como o JavaFX é uma tecnologia executada no lado do cliente, primeiramente quando aplicação é executada é feito o download da mesma para o computador do cliente, em nenhum navegador ocorreu problemas nessa etapa.

Na exibição do programa os navegadores Mozilla Firefox, Opera e Internet Explorer apresentaram o mesmo conteúdo com os mesmos temas e cores, já o Google Chrome apresentou uma diferença na exibição, apresentando um tema mais escuro em todas as funcionalidades testadas, outro aspecto analisado no Google Chrome é o fato de que a aplicação não se ajustou corretamente nos padrões que foi desenvolvida, tendo alguns componentes sendo escondidos como se pode observar na Figura 13, fato que não é apresentado por nenhum outro navegador.

Figura 15 – Execução do programa no Google Chrome.

Fonte: Do Autor

No quesito performance, todos os navegadores desempenharam todas as funcionalidades implementadas de maneira correta e sem apresentarem nenhum problema durante os teste.

Tabela 5 – Comparação da aplicação nos navegadores no Windows

	Instalação	Elementos visuais	Performance
Google Chrome	Instalação realizada com sucesso	Apresentou tonalidades, temas diferentes e também elementos que não se ajustaram corretamente na tela	Todas as operações realizadas na aplicação foram concluídas com êxito
Mozilla Firefox	Instalação realizada com sucesso	Todos os elementos visuais apresentados corretamente	Todas as operações realizadas na aplicação foram concluídas com êxito
Opera	Instalação realizada com sucesso	Todos os elementos visuais apresentados corretamente	Todas as operações realizadas na aplicação foram concluídas com êxito
Internet Explorer	Instalação realizada com sucesso	Todos os elementos visuais apresentados corretamente	Todas as operações realizadas na aplicação foram concluídas com êxito

Fonte: Do Autor

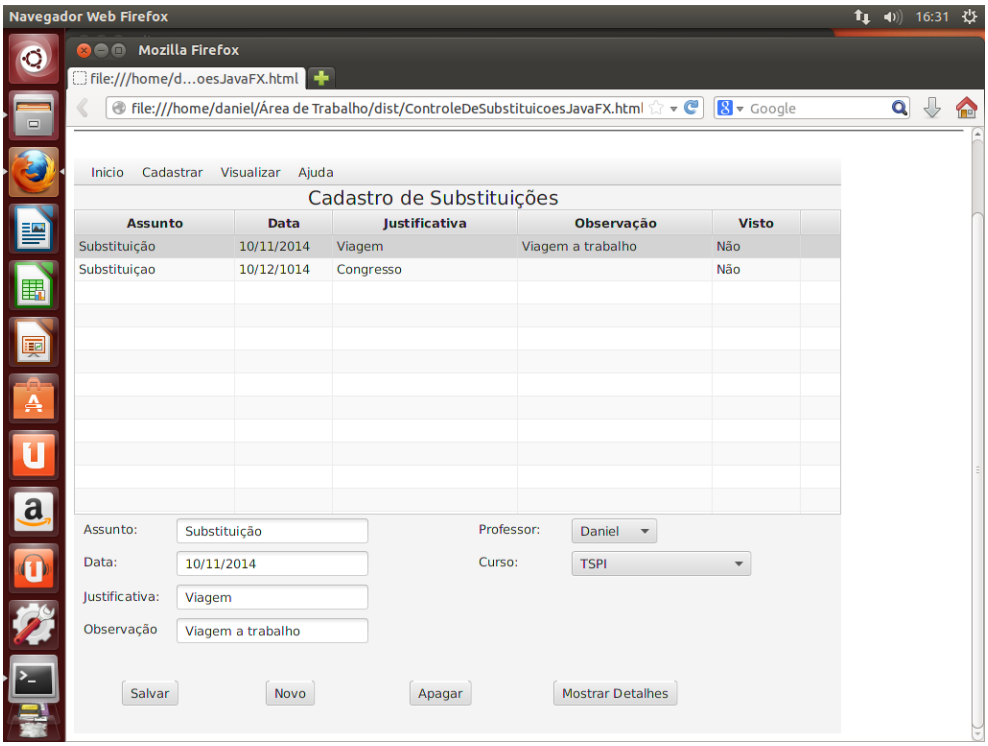
4.2 LINUX

No sistema operacional Linux, levando em consideração os testes no Ubuntu, a aplicação é avaliada nos navegadores Google Chrome, Mozilla Firefox e Opera, em cada um deles serão utilizados os mesmos testes realizados no Windows.

- **Google Chrome:** Os testes foram tentados em duas versões do Ubuntu, porém nas duas versões a instalação do plugin do Java no Google Chrome não foram efetivadas e todas as tentativas de instalação e configuração não foram concluídas, portanto a aplicação não pode ser avaliada neste sistema operacional.

- **Mozilla Firefox:**

Figura 16 – Aplicação sendo executada no Mozilla Firefox



Fonte: Do Autor

- **Opera:**

Figura 17 – Aplicação sendo executada no Opera

Fonte: Do Autor

Assim como no Windows, para poder executar a aplicação é necessário o cliente possuir o JRE ou o Java Development Kit (JDK) instalado no seu computador, destacando que as versões do JRE ou JDK devem ser a mesma ou superior ao da versão que foi construída a aplicação, caso contrario a execução irá apresentar a opção para atualizar a versão do Java. Para execução nos navegadores não é necessário nenhum plugin.

Na execução da aplicação diretamente no desktop, assim como no Windows, a aplicação se comportou de maneira eficiente e sem constar problemas na sua execução.

Na execução via Web, o navegador Mozilla Firefox realizou o download da aplicação e executou a mesma de maneira correta, porem o Opera apresentou um erro logo na instalação, dizendo que não pode apresentar o conteúdo usando este navegador e recomendando a utilização de outro navegador para realizar isso, como se pode observar na Figura 17.

O Mozilla Firefox apresentou o mesmo conteúdo com os mesmos temas e cores padrões do programa, não houve nenhuma distorção e os componentes se ajustaram de maneira correta na tela. No quesito performance, o Mozilla Firefox desempenhou todas as funcionalidades implementadas de maneira correta e sem apresentarem nenhum problema durante os teste.

Tabela 6 – Comparação da aplicação nos navegadores no Ubuntu

	Instalação	Elementos visuais	Performance
Google Chrome	Não foi possível realizar o teste	Não foi possível realizar o teste	Não foi possível realizar o teste
Mozilla Firefox	Instalação realizada com sucesso	Todos os elementos visuais apresentados corretamente	Todas as operações realizadas na aplicação foram concluídas com êxito
Opera	Erro na instalação	Não foi possível realizar este teste	Não foi possível realizar este teste

Fonte: Do Autor

CONSIDERAÇÕES FINAIS

A plataforma JavaFX apesar de seu crescente número de usuários e dos feedbacks positivos que recebe em suas comunidades, ainda passa por uma fase de experimentação. A versão 2.0 que trouxe significativas mudanças na tecnologia que ainda é muito recente, de forma que ainda não foi criado um consenso geral dentro da comunidade sobre as possibilidades de utilização desta plataforma. Vale ressaltar que a própria Oracle cita o JavaFX como possível substituto do Swing que é a sua tecnologia para desenvolvimento desktop, ou seja, há um investimento na plataforma. Isso é notado pelo suporte que é oferecido na página oficial do JavaFX, que oferece além de uma boa documentação.

O estudo de caso apresentado mostra uma visão sobre o desenvolvimento com JavaFX, a aplicação desenvolvida é considerada relativamente simples, porem pode-se observar que a curva de aprendizagem é muito grande, justamente pelo fato de não necessitar aprender uma nova linguagem de programação e sim adquirir maneiras novas de programar em Java.

Levando em consideração que atualmente existem poucas bibliografias referentes ao assunto e o JavaFX possui varias características que podem ser avaliadas, este trabalho buscou avaliar o suporte multiplataforma da tecnologia, através de testes utilizando a aplicação desenvolvida. A partir dos testes é possível perceber que aplicações desenvolvidas com JavaFX possuem uma boa tendência para funcionar multiplataforma, considerando que a tecnologia ainda passe por melhorias, possivelmente os problemas apresentados em alguns teste neste trabalho podem ser corrigidos.

A principal desvantagem desta tecnologia é o fato de que o usuário deve ter a versão do JRE atualizada para executar as aplicações, esse fator pode ser considerado problemático, pois, dificilmente usuários comuns mantêm seus computadores com o Java atualizado.

Por fim, são destacados aspectos que podem ser aprofundados e outras características a serem avaliadas futuramente:

- Aperfeiçoar a avaliação multiplataforma para abranger testes no sistema operacional Mac Os X;
- Ampliar os testes feitos, adicionado critérios como a avaliação do consumo de memória comparando com aplicações construídas em Swing;
- Estudar o uso de HTML, CSS e JavaScript em Aplicações JavaFX;
- Investigar a atualização de aplicações Swing com recursos JavaFX, como a reprodução de conteúdos gráficos e conteúdo da web incorporado.

REFERÊNCIAS

ADOBE. *Using Adobe Flex 4.6*. Disponível em: <http://www.adobe.com/go/learn_flex46_alldocumentation_en/> Acesso em: 29 mai. 2013(a).

_____. Faq. Disponível em: <http://www.adobe.com/br/products/flex/faq.html> Acesso em: 28 mai. 2013(b).

APRESENTANDO as aplicações de Internet Rica (RIAs). Capítulo 1. Disponível em: <http://goo.gl/ijt2iP> Acesso em: 25 mai. 2013.

Apostila de Treinamento K19 - Persistência JPA e Hibernate (2012) disponível em http://www.k19.com.br/downloads/apostilas/java/k19-k21-persistencia-com-jpa2-e-hibernate acessado em 01/10/2013.

BOOCH, Gray; RUMBAUGH, James; JACOBSON, IVAR. UML: Guia do Usuário. Rio de Janeiro. Elsevier. 2012.

BOZZON, Alessandro et al. Conceptual Modeling and Code Generation for Rich Internet Applications. Julho de 2006. Disponível em: <http://www.webml.org/webml/upload/ent5/1/fp553-Bozzon.pdf> Acesso em: 25 mai 2013.

GIONGO, Regis; ARAÚJO, Everton Coimbra. Introdução ao JavaFX 2.0. *Java Magazine*. Rio de Janeiro, n. 23, p. 15-25, 2012.

HORSTMANN, Cay S.; CORNELL, Gary. Core Java Volume I – Fundamentos. 8. ed. São Paulo. Pearson Prentice Hall, 2010.

LOPES, Jackson dos Santos; TAVARES, Rodrigo Oliveira. *JavaFX*: Uma abordagem ao desenvolvimento de aplicações RIA. 2010. Monografia (Graduação em Ciência da Computação) – Universidade FUMEC, Belo Horizonte, 2010. Disponível em: <http://www.ricardoterra.com.br/publications/students/2010_fumec_lopes_e_tavares.pdf> Acesso em: 10 mai. 2013.

MACORATI, José Carlos. Usando SilverLight 4, RIA Services e Entity Framework 4. Disponível em: <http://www.macoratti.net/11/01/net_slv2.htm> Acesso em: 28 mai. 2013.

MICROSOFT. Silverlight Overview. Disponível em: <http://msdn.microsoft.com/pt-br/library/bb404700(v=vs.95).aspx> Acesso em: 09 jun. 2013(a).

_____. XAML Overview. 2010. Disponível em: <http://msdn.microsoft.com/pt-br/library/cc189036(v=vs.95).aspx> Acesso em: 09 jun 2013.

_____. Silverlight Architecture. Disponível em: <http://msdn.microsoft.com/pt-br/library/bb404713(v=vs.95).aspx> Acesso em: 11 jun. 2013(b).

NOGEIRA, Admilson. UML - Unified Modeling Language - Introdução e Histórico. Disponível em: <http://www.linhadecodigo.com.br/artigo/763/uml-unified-modeling-language-introducao-e-historico.aspx> Acesso em: 22 jun. 2013.

ORACLE. What is JavaFX. Disponível em: <http://docs.oracle.com/javafx/2/overview/jfxpub-overview.htm> Acesso em: 29 abr. 2013(a).

_____. JavaFX Architecture. Disponível em: <http://docs.oracle.com/javafx/2/architecture/jfxpub-architecture.htm> Acesso em: 10 mai. 2013(b).

_____. The Java Persistence API - A Simpler Programming Model for Entity Persistence . Disponível em: <http://www.oracle.com/technetwork/articles/javaee/jpa-137156.html> Acesso em: 22 set. 2014(c).

QUERINO FILHO, Luiz Carlos. Por dentro do JavaFX 2. *Java Magazine*. Rio de Janeiro, n. 105, p. 20-32, 2012.

ROCHA, Ricardo de Almeida. *Silverlight e JavaFX*: Comparação de frameworks para desenvolvimento de aplicações ricas para a Internet. 2010. Monografia (Pós-Graduação em Desenvolvimento Orientado a Objetos - Java) – Centro Universitário de Maringá, Maringá, 2010. Disponível em: <http://www.munif.com.br/munif/arquivos/SILVERLIGHTJAVAFX.pdf?id=232> Acesso em: 25 mai 2013.