

JULIANO OLIVEIRA DE PONTES

APLICAÇÕES RICAS PARA INTERNET

ASSIS
2008

APLICAÇÕES RICAS PARA INTERNET

JULIANO OLIVEIRA DE PONTES

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
como requisito do Curso de Graduação, analisado
pela seguinte comissão examinadora:

Orientador: _____

Analisador (1): _____

Analisador (2): _____

ASSIS
2008

JULIANO OLIVEIRA DE PONTES

APLICAÇÕES RICAS PARA INTERNET

Trabalho de Conclusão de Curso apresentado ao
Instituto Municipal de Ensino Superior de Assis,
como requisito do Curso de Graduação, analisado
pela seguinte comissão examinadora:

Orientador: _____

Área de Concentração: _____

ASSIS
2008

AGRADECIMENTOS

A Deus, pela oportunidade e pelo privilégio de compartilhar tamanha experiência ao freqüentar este curso, por perceber e atentar para a relevância de temas que não faziam parte, em profundidade, das nossas vidas.

Ao Orientador Prof. Dr. Almir Rogério Camolesi pelo incentivo, simpatia e presteza no auxílio às atividades e discussões sobre o andamento e normatização deste Trabalho de Conclusão de Curso.

A todos os professores e seus convidados pelo carinho, dedicação e entusiasmo demonstrado ao longo do curso.

Aos colegas de classe, pela espontaneidade e alegria na troca de informações e materiais, uma rara demonstração de amizade e solidariedade.

Aos meus pais pelo empenho em me proporcionar todas as condições necessárias para que eu pudesse estudar.

Aos meus familiares pelo apoio, incentivo e paciência em tolerar a minha ausência.

A minha namorada Caroline, pela compreensão, amor e apoio em todos os momentos desta jornada.

RESUMO

A necessidade de interfaces mais interativas e uma melhor experiência de navegação para o usuário em aplicativos WEB forçaram a criação de um modelo que atendesse esses requisitos, este modelo foi titulado de Aplicações Ricas para Internet (RIA), e busca sanar as necessidades a ele impostas. O trabalho aborda os conceitos desta tecnologia, enfatizando as melhorias que ela pode proporcionar ao desenvolvimento de Aplicativos WEB. A abordagem é baseada na pesquisa e na implementação de um aplicativo que possui uma interface rica para o usuário, demonstrando de forma pratica seus conceitos e vantagens proporcionadas pelo uso desta tecnologia.

Palavras-chave: RIA. Interatividade.

ABSTRACT

The necessity of more interactive interfaces and a better experience to the user navigation in WEB application forced the creation of a model that look at that requirements, this model was titled as Rich Internet Applications (RIA), and the aim is to cure the imposed necessities of it. This work approaches the concepts of that technology emphasizing the improvements that it can provide to the development of WEB Application. This approach is based on the research and the implementation of application that possess a rich interface for the user thus demonstrating in a practical form, its concepts and advantages proportionate by the use of this technology.

Keywords: RIA. Interactivity.

LISTA DE FIGURAS

Figura 1 - Proposta do Trabalho.....	12
Figura 2 - Funcionalidade de Aplicações Ricas para Internet	15
Figura 3 - Categorização de RIA	16
Figura 4 - Código fonte MXML.	20
Figura 5 - Diagrama de Caso de Uso.....	27
Figura 6 - Diagrama de Classe (Cliente)	29
Figura 7 - Diagrama de Classe (Servidor)	30
Figura 8 - Diagrama de Seqüência para zoom no mapa	31
Figura 9 - Diagrama de Seqüência para mover mapa.....	32
Figura 10 - Diagrama de Seqüência para manipular os modos de visualização	33
Figura 11 - Diagrama de Seqüência para cadastrar ponto.....	33
Figura 12 - Diagrama de Seqüência para visualizar os pontos do mapa	34
Figura 13 - Diagrama de Seqüência para extrair informação	34
Figura 14 - Diagrama de Interface.....	35
Figura 15 - Interface de Pesquisas.....	36
Figura 16 - Informações de um ponto	36
Figura 17 - Integração das tecnologias Flex e .NET	37
Figura 18 - Classe serializada em ActionScript.....	38
Figura 19 - Classe serializada em .NET	38
Figura 20 - Etapas de Desenvolvimento	39
Figura 21 - Criação do mapa.....	39
Figura 22 - Interface do Mapa	40
Figura 23 - Chamada de método no servidor.....	40
Figura 24 - Pontos carregados no mapa	41
Figura 25 - Filtro do tipo de ponto	42
Figura 26 - Função auto completar	42
Figura 27 - Uso de <i>Bindable</i>	43
Figura 28 - Gerenciamento de janelas	44

LISTA DE TABELAS

Tabela 1 - Tabela de conversão de tipos de ActionScript para .NET	22
Tabela 2 - Tabela de conversão de tipos de .NET para ActionScript	23

LISTA DE ABREVIATURAS E SIGLAS

AJAX	<i>Asynchronous JavaScript and XML</i>
AMF	<i>Action Message Format</i>
API	<i>Application Programming Interface</i>
CSS	<i>Cascading Style Sheets</i>
DOM	<i>Document Object Model</i>
HTML	<i>HyperText Markup Language</i>
MXML	<i>Macromedia Flex Markup Language</i>
RIA	<i>Rich Internet Applications</i>
SQL	<i>Structured Query Language</i>
XHTML	<i>Extensible HyperText Markup Language</i>
XML	<i>Extensible Markup Language</i>
XSLT	<i>Extensible Stylesheet Language Transformations</i>
WCF	<i>Windows Communication Foundation</i>
WF	<i>Windows Workflow Foundation</i>
WPF	<i>Windows Presentation Foundation</i>
WWW	<i>World Wide Web</i>

SUMÁRIO

1. INTRODUÇÃO	12
2. RIA	14
2.2. CONCEITO	15
2.3. PRINCIPAIS PLATAFORMAS DISPONÍVEIS	17
2.3.1. Adobe Flex	17
2.3.2. AJAX	18
2.3.3. Microsoft Silverlight	18
2.3.4. OpenLaszlo	19
2.3.5. Sun JavaFX	19
3. TECNOLOGIAS PARA EXPERIMENTAÇÃO	20
3.1. FRAMEWORK FLEX	20
3.2. FLUORINEFX	21
3.3. FRAMEWORK .NET	23
3.4. API DE MAPAS DO GOOGLE	24
4. MODELAGEM	25
4.1. ANÁLISE DE REQUISITOS	26
4.2. CASOS DE USO	27
4.3. DIAGRAMA DE CLASSES	28
4.3.1. Cliente	29
4.3.2. Servidor	30
4.4. DIAGRAMA DE SEQÜÊNCIA	31
4.5. DIAGRAMA DE INTERFACE	35
5. IMPLEMENTAÇÃO DO EXPERIMENTO	37
5.1. INTERFACE DO MAPA	39
5.2. CARREGAMENTO DOS PONTOS	40
5.3. FILTRO DOS PONTOS	41
5.4. PESQUISA DE PONTOS	42

5.5. EXTRAIR INFORMAÇÕES	43
5.6. CADASTRAR PONTOS	44
6. CONCLUSÃO	45
REFERÊNCIAS BIBLIOGRÁFICAS	46

1. INTRODUÇÃO

Em meados dos anos 90 com o crescimento da Internet surgiram as Aplicações para Internet que tinham navegação de forma estática e forneciam pouca interatividade ao usuário. Este modelo revelou-se um sucesso pelo potencial de alcançabilidade proporcionado, mas com o passar do tempo e o aumento das exigências de software, acabaram forçando a criação de um novo modelo que as atendessem, assim, surgiram as Aplicações Ricas para Internet (RIA), que buscam fornecer as melhores funcionalidades de uma aplicação gráfica instalada e executada diretamente no computador do cliente (*Desktop*), agregando conceitos de interatividade e tecnologias de comunicação para atender as novas exigências (ALLAIRE, 2002).

Este trabalho tem como objetivos a demonstração da tecnologia RIA, como foco principal, visa difundir o conhecimento obtido no meio acadêmico, demonstrando as vantagens e melhorias que esta tecnologia pode proporcionar, conforme apresentado na Figura 1.

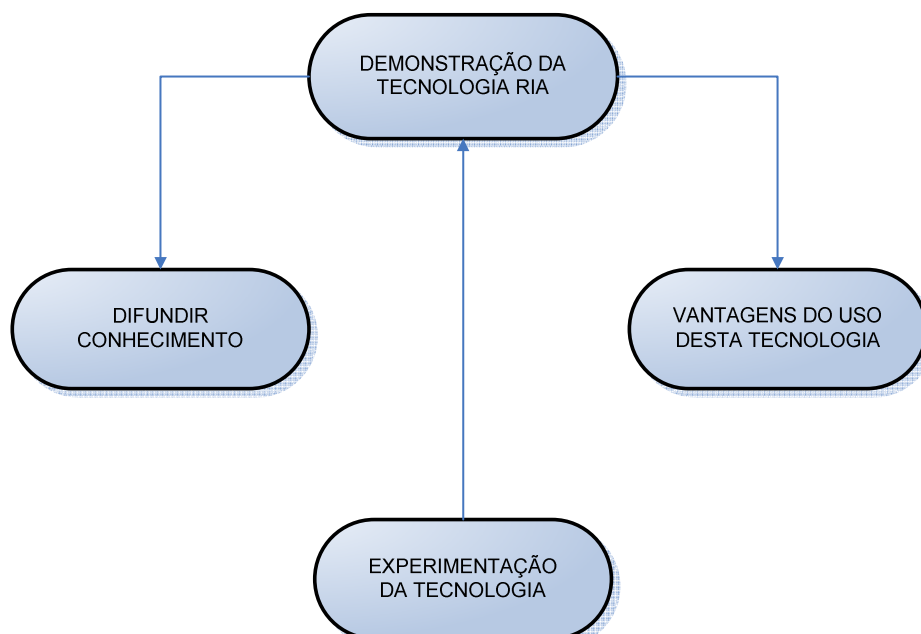


Figura 1 - Proposta do Trabalho

Com base em tendências de mercado de software Web, notou-se que as tecnologias RIA estão sendo amplamente discutidas e cada vez mais usadas em grandes

segmentos de mercado. No meio acadêmico, por meio de pesquisa verificou-se uma carência de material sobre este tema.

Perante a dificuldade de encontrar material científico sobre o tema abordado, surgiu a necessidade de neste trabalho discutir o conceito de RIA. O fato de grandes empresas utilizarem esta tecnologia tem impulsionando a tendência de mercado para este tipo de desenvolvimento. Esses fatores incentivaram ainda mais a decisão desta abordagem de estudo.

Como proposta deste trabalho, fica o estudo e experimentação da tecnologia RIA, buscando a integração entre algumas tecnologias disponíveis para desenvolver aplicativos Web que forneçam maior interação ao usuário, aumentando sua satisfação e conseqüentemente viabilizando melhores resultados perante a utilização de softwares Web, contribuindo ao desenvolvimento do negócio das empresas, proporcionando maiores lucros e diversos outros fatores.

Com isso este trabalho divide-se na seguinte forma:

No 2º capítulo será abordado o histórico, conceito e principais tecnologias disponíveis para o desenvolvimento de RIA.

O 3º capítulo trará uma descrição mais detalhadas sobre as tecnologias escolhidas para o desenvolvimento do experimento.

A modelagem e descrição do experimento a ser implementado será o tema do 3º capítulo.

O 4º capítulo trata sobre as fases realizadas no desenvolvimento do experimento.

As conclusões alcançadas obtidas na pesquisa e experimentação de RIA são abordadas no 5º capítulo.

E para finalizar, ficam disponíveis após todo o contexto desenvolvido neste trabalho, as referências bibliográficas utilizadas.

2. RIA

Esse capítulo descreve o conceito de RIA (*Rich Internet Applications*), histórico, e as principais tecnologias disponíveis para o desenvolvimento de aplicações utilizando esta tecnologia.

2.1. HISTÓRICO

O explosivo crescimento da Internet e do WWW (*World Wide Web*) em meados dos anos 90, impulsionou a criação de um novo modelo de aplicações, que utiliza os computadores pessoais ligados à Internet, chamada de Aplicações de Internet. Este modelo enfatiza um baixo custo de desenvolvimento e a entrega da aplicação mais rápida, é baseado em páginas HTML (*HyperText Markup Language*) e poderosos servidores de aplicações que criam e enviam as páginas aos navegadores (*browsers*) de Internet. Fornece ao cliente uma navegação de forma estática, logo, todo tipo de interação que o cliente tiver com a página é enviado ao servidor para que processe a requisição, efetue validações e consultas, e devolva a página ao cliente com as informações atualizadas (ALLAIRE, 2002).

Este modelo revelou-se um sucesso, devido ao amplo potencial de alcançabilidade, mas com a demanda da sofisticação das aplicações, com passar do tempo e com a limitação de seu desenvolvimento, começaram a ter um resultado frustrado, muitas vezes confuso, causando uma infeliz experiência ao usuário.

Tal cenário forçou a criação de um novo modelo que atendesse aos requisitos das aplicações e fornecesse maior interatividade e riqueza à experiência com o usuário. Em Março de 2002, a Macromedia (ALLAIRE, 2002), por meio de um *White Paper*¹ introduziu o termo RIA, ou seja, Aplicações Ricas para Internet que utiliza alguns princípios do modelo antigo de Aplicações de Internet, porém busca fornecer as melhores funcionalidades de uma Aplicação *Desktop*, com alcançabilidade e baixo custo de desenvolvimento de Aplicações Web, agregando conceitos de interatividade e tecnologias de comunicação como demonstrado na Figura 2,

¹ Documento técnico de divulgação

atendendo assim os novos requisitos para desenvolvimento de aplicações. O conceito de RIA teve anteriormente outros nomes, *Remote Scripting* pela Microsoft em 1998 e *X Internet* pela Forrester Research em 2001.

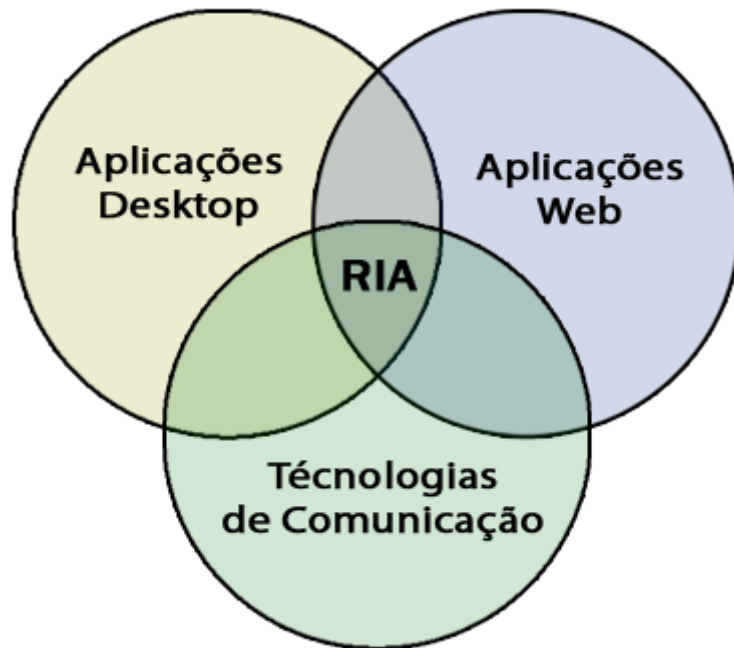


Figura 2 - Funcionalidade de Aplicações Ricas para Internet

2.2. CONCEITO

Tal modelo utiliza o mesmo conceito arquitetônico das Aplicações para Internet, baseando-se nos computadores pessoais com acesso a Internet ou Intranet, contudo, ele centraliza a aplicação no lado do cliente, executando a mesma diretamente nos navegadores, efetuando o envio de requisições aos servidores apenas para consultas mais complexas que não possam ser executadas no cliente, acesso à base de dados ou servidores multimídia, diminuindo muito o tráfego de rede, tornando a navegação mais rápida e elevando a satisfação do usuário. Disponibiliza também, recursos interativos que no mínimo se comparam as interações em aplicações *Desktop* encontradas atualmente (FAIN, RASPUTNIS, TARTAKOVSKY, 2007). A Figura 3 ilustra a categoria que as Aplicações Ricas para Internet se enquadram, mostrando as tecnologias que são abrangidas por esse conceito, e a categorização de processamento que pertencem, mostrando o quanto relacionado estão perante as Aplicações Web e Aplicações *Desktop*.

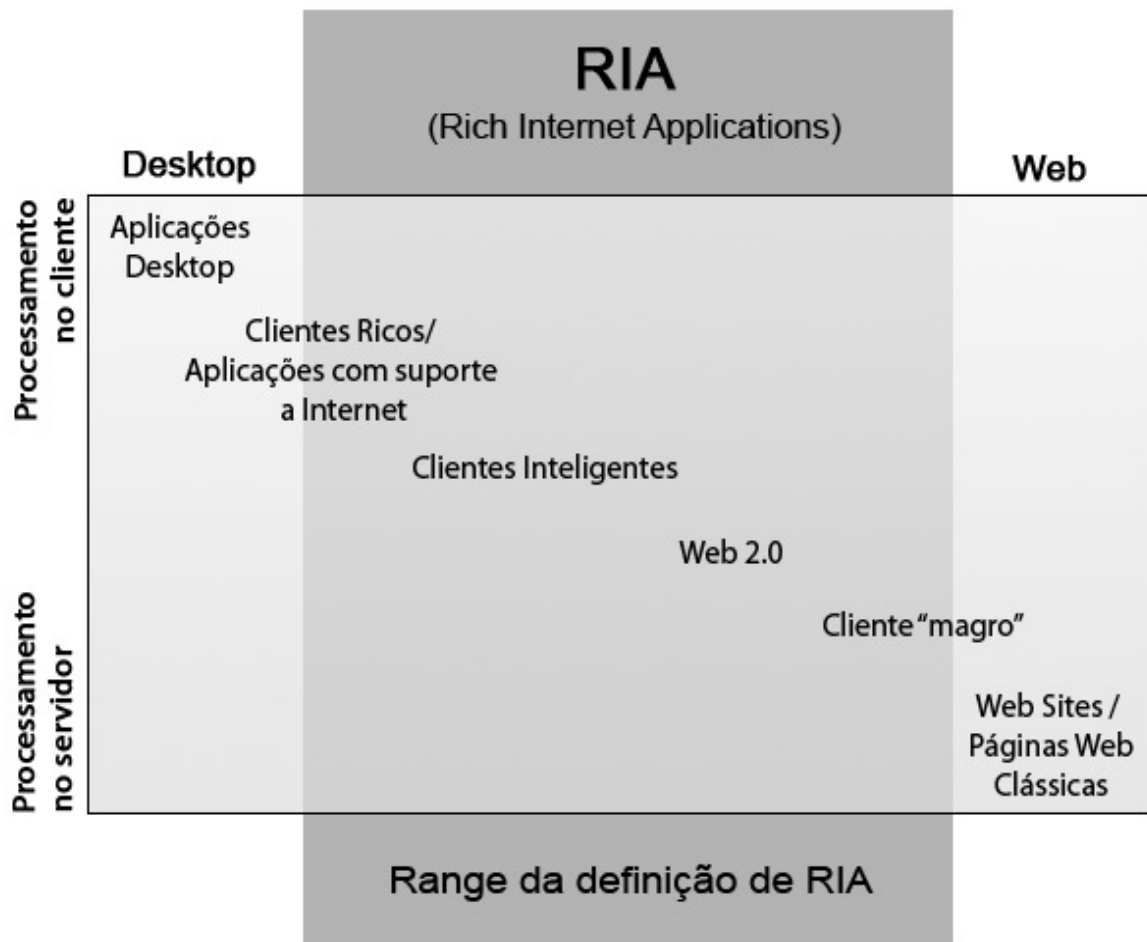


Figura 3 - Categorização de RIA (MORITZ, 2008, p. 6)

A Figura 3 demonstra a categoria de aplicações que a tecnologia RIA abrange, descrevendo quais tipos de aplicações são englobados em seu contexto. No eixo vertical da Figura 3, classifica-se a quantidade de processamento que é feita no servidor e no cliente, no eixo horizontal é classificado se as Aplicações estão relacionadas com Aplicações *Desktop* ou Aplicações *Web*. Cada tipo de aplicação citado na Figura 3 pode ser descrito abaixo:

As Aplicações *Desktop* compreendem aplicações que rodam diretamente na máquina do cliente, necessitando ser instaladas em um sistema operacional, estas aplicações possuem alto grau de interatividade com o usuário (MORITZ, 2008), como exemplo: *Microsoft Word*, *Adobe Photoshop*.

Os Clientes Ricos / Aplicações com suporte a Internet são Aplicações *Desktop* que fazem utilização de rede, mas também funcionam desconectadas, podendo ter suas

funcionalidades limitadas (MORITZ, 2008), como exemplo: *Microsoft Outlook Express, Mozilla Thunderbird*.

Os clientes inteligentes (*Smart Client*) podem ser descritos como uma combinação dos benefícios de uma Aplicação para Clientes Ricos com a facilidade de instalação e gerenciamento das Aplicações para Cliente “magro” (MSDN).

Segundo O'Reilly (2006),

Web 2.0 é a revolução dos negócios na indústria da informática causada pela mudança da Internet como plataforma, e uma tentativa de compreender as regras para o sucesso nessa nova plataforma. Entre essas regras estão: Construir Aplicações que explorem os efeitos dessa rede para conseguir o melhor e que mais pessoas as usem.

O cliente “Magro” (*Thin Client*) / Web Sites / Páginas Web Clássicas, são Aplicações baseadas em navegadores, instaladas e atualizadas em um servidor que as disponibiliza na rede ou Internet, são de fácil manutenção, mas exigem que o navegador esteja o tempo todo conectado a rede ou Internet (MSDN).

2.3. PRINCIPAIS PLATAFORMAS DISPONÍVEIS

2.3.1. Adobe Flex

O Flex é uma das principais plataformas disponíveis para o desenvolvimento de Aplicações Ricas para Internet, é um *framework* de código aberto para construção de Aplicações para Web altamente interativas e expressivas, com implantação consistente na maioria dos navegadores, computadores pessoais e sistemas operacionais existentes. Fornece suporte à padrões de projeto, utiliza-se de uma linguagem declarativa baseada em XML (*Extensible Markup Language*) chamada MXML (*Macromedia Flex Markup Language*), para desenvolvimento da interface com o usuário e comportamento da Aplicação, para cuidar da lógica da Aplicação dispõe da Linguagem ActionScript, uma poderosa linguagem orientada a objeto (ADOBE, 2007).

2.3.2. AJAX

O termo AJAX (*Asynchronous JavaScript and XML*) não é uma tecnologia, e sim um conjunto de várias tecnologias, cada uma com suas funções. As tecnologias que compõem o Ajax são: XHTML, CSS, JavaScript, DOM, XML, XSLT, XMLHttpRequest (GARRETT, 2005).

As especificações definem o termo XHTML (*Extensible HyperText Markup Language*) como uma junção do HTML com o XML, tornando-a uma linguagem de marcação (W3C). O CSS (*Cascading Style Sheets*) é um mecanismo que adiciona estilos às páginas Web, separando a formatação de estilos da estrutura principal do documento (W3C).

O Javascript é uma linguagem *script* orientada a objeto independente de plataforma que roda no navegador do cliente, ela pode conectar-se aos objetos carregados nas aplicações e pode manipulá-los, alterando o estilo das páginas Web (MDC, 2007).

O DOM (*Document Object Model*) é uma plataforma que permite a manipulação dinâmica do conteúdo de uma página Web, estrutura e estilo.

Ele transforma os elementos da página Web em uma hierarquia de objetos que por meio da manipulação *Javascript* pode-se criar, modificar e remover elementos da página Web dinamicamente (W3C).

A linguagem XML é um formato de texto simples, flexível e bastante utilizada para intercambio de dados (W3C).

O XSLT é uma linguagem de transformação de documentos XML, que utiliza regras para transformar o documento XML original em outro documento XML transformado a partir das regras configuradas (W3C).

O objeto XMLHttpRequest é uma API que possibilita que scripts do lado do cliente possam acessar dados no servidor (W3C).

2.3.3. Microsoft Silverlight

Segundo o MSDN,

Microsoft Silverlight é uma implementação do *framework* .NET independente de plataforma para construir e exibir a nova geração de experiência em mídia e RIA para a Web. Silverlight unifica as capacidades do servidor, da Web, do *Desktop*, do código gerenciado e linguagens dinâmicas, a declaração e programação tradicional e a força do WPF (*Windows Presentation Foundation*).

“O .NET *framework* é um modelo de programação de código gerenciado da Microsoft para criar aplicativos em clientes, servidores e dispositivos móveis ou incorporados do Windows” (MSDN).

O WPF é um componente do *framework* .NET que disponibiliza um modelo de programação para construção de Aplicações *Desktop* e incorporam uma rica interface com o usuário, mídia e documentos (MSDN).

2.3.4. OpenLaszlo

É uma plataforma de código aberto para criação de RIA e se divide em 3 componentes (OPENLASZLO) citados abaixo :

O Compilador OpenLaszlo, compila o código fonte da aplicação em arquivo Swf², que roda na maioria dos navegadores.

Um *framework* de execução, que fornece componentes de interface, dados e serviços de rede.

E um Servlet³ que permite a execução de tipos de mídia adicional.

2.3.5. Sun JavaFX

O “JavaFX é uma família de produtos para criação de RIA” (SUN, 2008), fornece ambiente de execução e ferramentas para que desenvolvedores e designers facilmente construam Aplicações Ricas para Internet

² Formato de arquivo multimídia.

³ Tecnologia que insere novos recursos a um servidor.

3. TECNOLOGIAS PARA EXPERIMENTAÇÃO

Este capítulo tem o objetivo de conceituar as tecnologias envolvidas na experimentação de uma aplicação, que será desenvolvida com intuito de demonstrar a aplicação dos conceitos de RIA. Serão utilizadas as seguintes tecnologias para tal experimentação: Adobe Flex, *framework* FluorineFx, *framework* .NET e a API⁴ de Mapas do Google para Adobe Flex.

3.1. FRAMEWORK FLEX

O *framework* Flex fornece funcionalidades para o desenvolvimento de RIA, é composto por uma linguagem declarativa, serviços de aplicação, biblioteca de componentes e conectividade a dados.

A MXML é uma linguagem de marcação, baseada em XML utilizada para construir a interface com o usuário nas aplicações construídas com Flex, essas características fazem com que a linguagem MXML torne-se mais estruturada e menos ambígua que o HTML, fornece componentes mais ricos que HTML e é renderizada no Flash Player (ADOBE, 2007). A Figura 4 exemplifica a linguagem MXML.

```
<?xml version="1.0" encoding="utf-8"?>

<mx:Application xmlns:mx="http://www.macromedia.com/2003/mxml">

    <mx:TextInput id="source" width="100"/>
    <mx:Button label="Copy" click="destination.text=source.text"/>
    <mx:TextInput id="destination" width="100"/>

</mx:Application>
```

Figura 4 - Código fonte MXML (ADOBE, 2007).

Para manipulação da lógica da aplicação diretamente no cliente, é utilizado a linguagem orientada à objetos ActionScript, citada anteriormente. A ActionScript é

⁴ Interface de Programação de Aplicativos

uma linguagem *script*, baseada no padrão internacional de linguagens de programação para *script* ECMAScript, e é executado pela AVM (*ActionScript Virtual Machine*) construída dentro do *Flash Player*⁵ (ADOBE, 2007).

O *framework* Flex disponibiliza para o desenvolvedor, bibliotecas de classes e serviços de aplicação que facilitam o desenvolvimento de RIA. Estes serviços incluem *data binding*, *drag-and-drop*, manipulação de interface, efeitos de movimento, transição e sistema de estilos, para manipular a aparência dos componentes de interface. Também disponibiliza bibliotecas de componentes, que contém todos os componentes necessários para desenvolvimento de interface, como: botões, *checkbox*, botões do tipo radio, *data grids* complexos, editor de texto e muitos outros.

Fornecer suporte para acesso de dados e a lógica de negócios em servidores de *back-end* por meio do Flex *Remoting* que tem funcionalidades para transportar os dados para a aplicação RIA, utiliza um formato chamado AMF (*Action Message Format*) para o transporte de dados que suporta transferência de forma binária e serializada garantido a integridade dos dados. O Flex *Messaging* disponibiliza a funcionalidade de publicar alterações de dados em tempo real e *data push*.

3.2. FLUORINEFX

FluorineFX é um *framework* de código aberto que faz a integração de uma aplicação Flex com um servidor com *framework* .NET. Com a sua utilização, é possível a implementação de Flex *Remoting* e Flex *Messaging*. Também atua como uma ponte entre a aplicação e o servidor.

Este *framework* funciona com enorme transparência, parecendo que a aplicação RIA esta acessando o servidor diretamente, ele também faz a conversão dos tipos de dados de uma linguagem para outra, facilitando o desenvolvimento. As conversões podem ser visualizadas na Tabela 1 e Tabela 2.

⁵ *Plugin* proprietário instalado no navegador (*browser*) que permite a execução de recursos multimídia e aplicações

ActionScript 2	ActionScript 3	.NET
undefined/null	undefined/null	null
Number	Number	Byte, SByte, UInt16, Int16, UInt32, Int32, UInt64, Int64, Decimal, Single, Double, Enum
---	int	Byte, SByte, UInt16, Int16, UInt32, Int32, UInt64, Int64, Decimal, Single, Double, Enum
Boolean	Boolean	System.Boolean
Date	Date	System.DateTime
String	String	String, Char, Guid, Enum
XML	XML	System.Xml.XmlDocument
Array	Array	Array, IList, System.Collections.ArrayList
Array Associativo	Array Associativo	System.Collections.Hashtable
---	mx.collections.ArrayCollection	FluorineFx.AMF3.ArrayCollection
---	flash.utils.ByteArray	byte[], FluorineFx.AMF3.ByteArray
Objeto	Objeto	FluorineFx.ASObject (Hashtable), objeto em .NET que possua um construtor sem parâmetros

Tabela 1 - Tabela de conversão de tipos de ActionScript para .NET (FLUORINEFX)

.NET	ActionScript 2	ActionScript 3
null, DBNull, System.Data.SqlTypes.INullable(when IsNull = true)	null	null
Byte/Sbyte, UInt16/Int16, UInt32/Int32, SqlByte, SqlInt16, SqlInt32	Number	int
UInt64/Int64, Decimal, Single, Double, SqlInt64, SqlSingle, SqlDouble, SqlDecimal, SqlMoney	Number	Number
System.Enum	Number	Number
System.Boolean, SqlBoolean	Boolean	Boolean
System.DateTime	Date	Date
SqlDateTime	-	-
System.String, System.Char, SqlString	String	String
System.Guid, SqlGuid	String	String
System.Xml.XmlDocument	XML	XML
System.Array	Array	Array

System.Collections.IList	Array	Array (quando "useLegacyCollection" está marcado), mx.collections.ArrayCollection
System.Collections.Hashtable	Objeto (sem tipo)	Objeto (sem tipo)
System.Collections.IDictionary		
System.Data.DataTable	RecordSet	DataTable Object(ASObject)
Sytem.Data.DataSet	Array associativo de objetos do Tipo RecordSet	Array associativo de objetos do tipo DataTable(ASObject)
FluorineFx.ASObject	Object	Object
System.Exception		
FluorineFx.AMF3.IExternalizable	---	Um objeto tipado implementando flash.utils.IExternalizable
FluorineFx.ASObject com a propriedade Tipo preenchida, qualquer outro tipo	Objeto Tipado/Objeto	Objeto Tipado/Objeto
FluorineFx.AMF3.ByteArray	---	flash.utils.ByteArray

Tabela 2 - Tabela de conversão de tipos de .NET para ActionScript (FLUORINEFX)

3.3. FRAMEWORK .NET

Segundo MSDN,

O *.NET Framework* é um modelo de programação de código gerenciado da Microsoft para criar aplicativos em clientes, servidores e dispositivos móveis ou incorporados do Windows. Os desenvolvedores podem usar o .NET para criar aplicativos de vários tipos: aplicativos Web, aplicativos para servidores, aplicativos de cliente inteligente, aplicativos de console, aplicativos de banco de dados e muito mais.

Para que possa fornecer suporte à criação de todos esses tipos de aplicativos, o *framework* .NET disponibiliza uma série de bibliotecas que fornecem suporte para o desenvolvimento como: o WPF que proporciona ao usuário experiências mais ricas na execução de aplicativos Desktop para Windows, o WCF (*Windows Communication Foundation*) para comunicações entre todos os aplicativos da empresa, o WF (*Windows Workflow Foundation*) para criação do fluxo de trabalho em qualquer aplicativo, e o ASP .NET para criação de aplicativos Web interativos (MSDN).

3.4. API DE MAPAS DO GOOGLE

A API de Mapas da Google é uma ferramenta para manipulação de mapas de forma interativa, que disponibiliza controles para manipulação do mapa, como controle de Zoom que permite que se manipule o campo visual do mapa, controle para mover o mapa e controle dos tipos de mapa, que podem ser definidos como mapa, imagem de satélite, terreno e híbrido (mapa + imagem de satélite).

Esta API também oferece sobreposições, que são objetos inseridos sobre o mapa em um ponto fixo, estes objetos podem designar ponto, linha ou área sobre o mapa.

Outra funcionalidade que esta disponível, são os serviços, como por exemplo o Geocoding, que atua na conversão de um endereço específico em coordenadas geográficas (GOOGLE).

4. MODELAGEM

Para poder alcançar os objetivos propostos, este capítulo discute a modelagem de um software para a experimentação de RIA.

O software tem como objetivos demonstrar a utilização de RIA e suas funcionalidades por meio do desenvolvimento de uma aplicação Web interativa, que é baseada no mapa da cidade de Assis. A mesma, irá fornecer de forma interativa informações pontuais sobre importantes referências nesta cidade, denominadas pontos de referência e pesquisas sobre estas informações.

Além das funcionalidades definidas, devem ser exibidas as informações alfanuméricas, previamente cadastradas em um banco de dados, há também informação ilustrativa como fotos dos pontos de referência.

A aplicação deve proporcionar uma melhor experiência de navegação ao usuário, trazendo recursos disponíveis em aplicações *Desktop*, como o recurso de arrastar e soltar objetos (*drag-and-drop*), menu flutuante e outros.

Para a concepção deste aplicativo, em primeira instância foi feita a modelagem do sistema, na busca de descrever as necessidades a serem atendidas, foi elaborado uma análise de requisitos, após esta análise o diagrama de casos de uso foi modelado para poder situar que tipos de serviços o sistema deverá disponibilizar e para que tipo de necessidades esse sistema terá aplicação. Após estas etapas, foi possível o levantamento das entidades e elaboração do modelo das classes que será descrito no diagrama de classes, As interações entre os objetos instanciados nesta aplicação, serão denotadas por meio do diagrama de seqüência, onde o diagrama de interface será responsável por modelar a interface ao usuário e descrever onde alguns dos principais conceitos de RIA estarão disponíveis neste aplicativo.

4.1. ANÁLISE DE REQUISITOS

Na concepção deste aplicativo foram levantados os requisitos necessários para que o experimento atenda os objetivos propostos, a seguir estão listados os requisitos do software proposto.

Este aplicativo deve basear-se no mapa da cidade de Assis, sendo assim, a aplicação deve controlar para que o mapa esteja localizado corretamente na posição desejada.

Um aplicativo baseado em mapa deve conter comandos de manipulação geográfica, desta forma o aplicativo proporcionará ao usuário, a manipulação dos níveis de visualização da aplicação com a utilização de comandos de Zoom de maior escala e Zoom de menor escala, o aplicativo também deve disponibilizar para o usuário, comando para a restauração da visualização ao nível inicial, com visualização total do mapa proposto.

O usuário também deve dispor de comando para movimentação do mapa, arrastando o mesmo.

O mapa em questão deve proporcionar diferentes modos de visualização ao usuário, sendo no formato de mapa, imagens de satélite, híbrido (Imagem de satélite e mapa) e terreno, deve também fornecer comandos para manipulação destes modos de visualização.

As informações sobre os pontos de referências devem ser demonstradas sobre o mapa de forma interativa, permitindo a interação do usuário com estas informações de forma simples e direta.

Quando solicitada uma informação, o aplicativo deve disponibilizar informações alfanuméricas e fotos sobre o local escolhido. O usuário deve ter como interagir com essas informações de maneiras diferenciadas, com comandos de pesquisa e auxílio a navegação das informações. Devem ser representadas informações de saúde e educação e principais pontos da cidade de Assis.

Para enriquecimento da base de dados, o aplicativo deve permitir que o usuário possa efetuar o cadastro de novos pontos de referência.

4.2. CASOS DE USO

O Diagrama de Casos de Uso é utilizado para demonstrar as funcionalidades de um sistema, após a análise dos requisitos, é possível por meio deste tipo de diagrama, modelar as funções ou serviços que o sistema deve prover, entendendo assim, o uso que o sistema terá e pra que tipo de aplicações será empregado.

A Figura 5 exibe o Diagrama de Casos de Uso proposto na modelagem deste experimento de RIA.

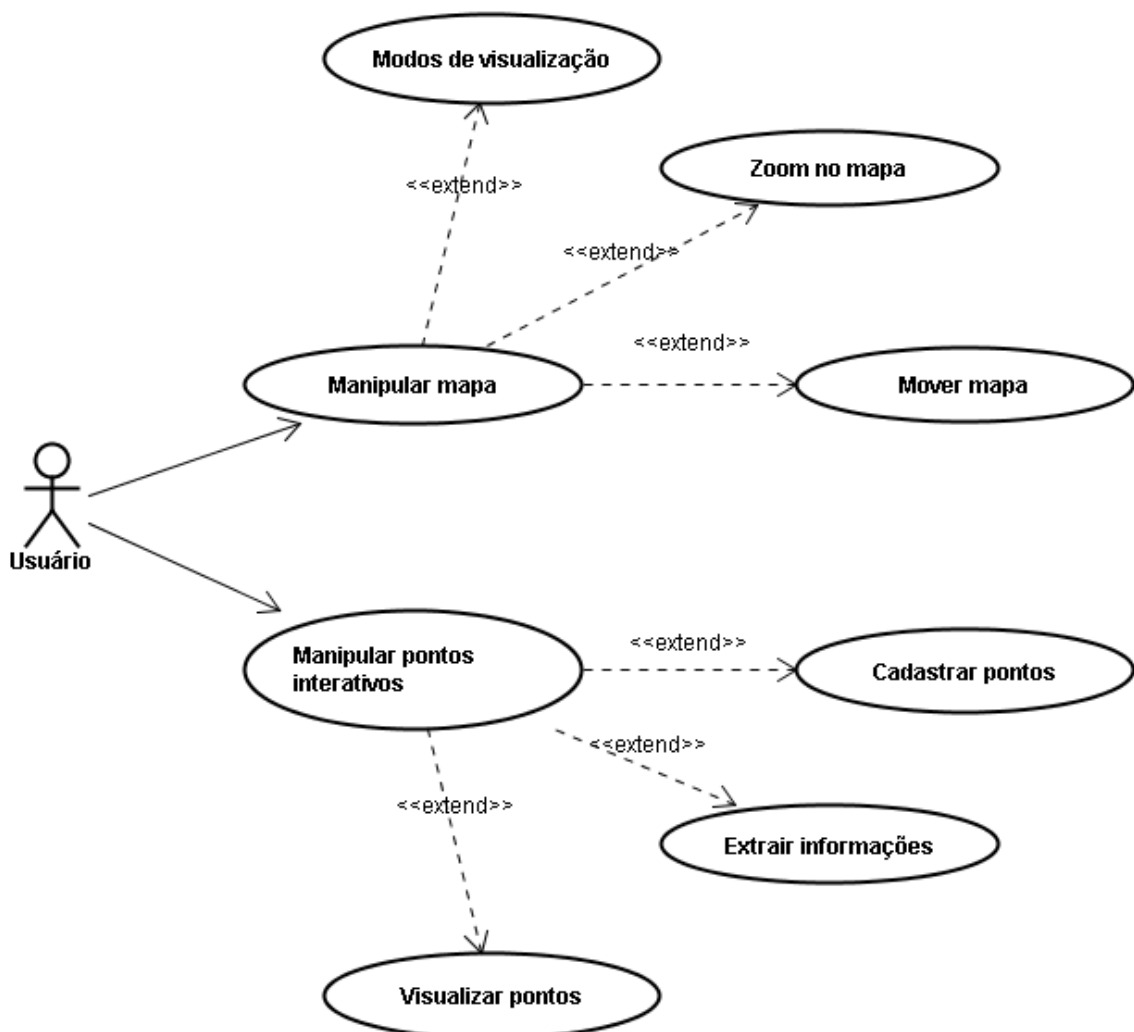


Figura 5 - Diagrama de Caso de Uso

Basicamente, este experimento será composto por dois casos de uso, Manipular mapa e Manipular pontos interativos.

O caso de uso Manipular mapa trata das interações do usuário perante o mapa, ele relaciona-se com outros casos de uso que podem ser acionados por meio de uma ação do usuário. Em sua especificação, fica claro os tipos de manipulação que o usuário pode ter com mapa, por meio deste caso de uso pode ser acionado o caso de uso Zoom no mapa, que deve disponibilizar serviços de manipulação gráfica do nível de visualização do usuário no mapa.

É possível também efetuar mudanças do modo de visualização do mapa com a utilização do caso de uso Modos de visualização, e a movimentação do mapa fica à cargo do caso de uso Mover mapa, que especifica que este determinado serviço deve estar disponível.

Outro caso de uso descrito é o Manipular pontos interativos, que por sua vez, relaciona-se com outros três casos de uso por meio da ação do usuário. Na manipulação dos pontos interativos é possível visualizar pontos que estejam previamente cadastrados na base de dados na definição do caso de uso Visualizar pontos, a visualização da informação destes pontos fica modelada no do caso de uso Extrair informações e o cadastro dos pontos é modelado pelo caso de uso Cadastrar ponto.

Sendo assim, esta seção demonstra de forma simples, o escopo de serviços e funcionalidades que o experimento a ser desenvolvido terá que disponibilizar.

4.3. DIAGRAMA DE CLASSES

Após a análise dos requisitos necessários para o desenvolvimento da aplicação proposta neste trabalho e a modelagem dos casos de uso desta aplicação é possível descrever o formato das classes e objetos que se faz necessário para o desenvolvimento do aplicativo. Em uma aplicação orientada a objetos, tais itens são as principais primitivas para a construção deste tipo de aplicação, sendo que um sistema orientado à objetos é composto por classes e objetos que interagem entre si para a execução dos serviços necessários ao funcionamento do aplicativo (STADZISZ, 2002, p. 15). Desta forma, é apresentada nas subseções seguintes a modelagem do diagrama de classes proposto para o desenvolvimento da experimentação proposta neste trabalho.

Para a implementação deste aplicativo baseado em RIA é necessário a modelagem de dois diagramas de classe, um para o lado cliente e outro para o lado servidor, tal fato ocorre para que possa concretizar a transferência de objetos entre as duas tecnologias, que serão envolvidas no desenvolvimento.

4.3.1. Cliente

O diagrama de classes do lado cliente modela a parte do aplicativo disponível à interação do usuário, ou seja, a sua interface. Como dito anteriormente, uma aplicação RIA roda praticamente toda no navegador do usuário, só recorre aos recursos do servidor para acesso à base de dados ou manipulações complexas, a partir destes conceitos, foi desenvolvido o diagrama exibido pela Figura 6.

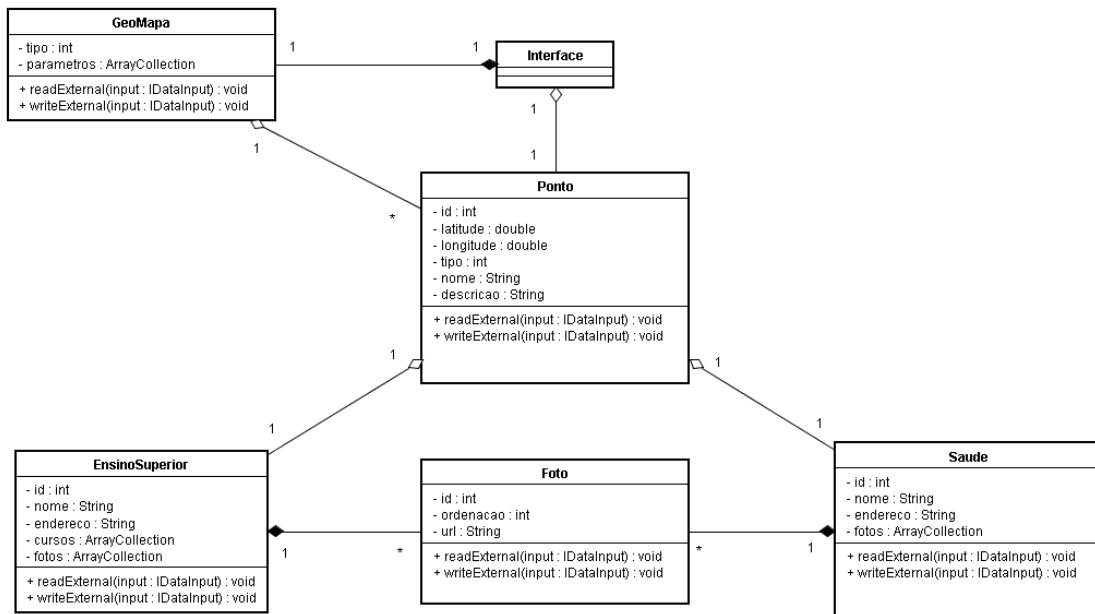


Figura 6 - Diagrama de Classe (Cliente)

Basicamente o diagrama de classes da Figura 6, modela as classes necessárias para a representação das informações propostas neste trabalho, contando também com uma classe para manipulação da interface e classe para guardar o status do mapa a ser manipulado.

A classe *GeoMapa* descreve o objeto necessário para manter o status do mapa e para designar o modo de operação que o sistema possa estar.

As classes *EnsinoSuperior*, *Saude* e *Foto*, são entidades que deverão ser utilizadas para armazenar as informações do sistema suprimindo a necessidade de uma consulta no lado servidor, neste caso, as mesmas serão enviadas para processamento e devolvidas com os dados requisitados.

É também composto de uma classe para gerenciamento da interface com o usuário.

4.3.2. Servidor

A modelagem do diagrama de classes representando o lado do servidor possui semelhança com a do lado cliente, devido à possibilidade da transferência de objetos entre ambos. Cada entidade definida no lado cliente deve ter a sua representação no lado do servidor de forma idêntica, pois, para que ocorra à transição de objetos entre os dois lados, estes devem ser iguais.

Esta modelagem abrange também, as classes necessárias para persistência de informações na base de dados. A Figura 7 ilustra a modelagem abordada neste tópico.

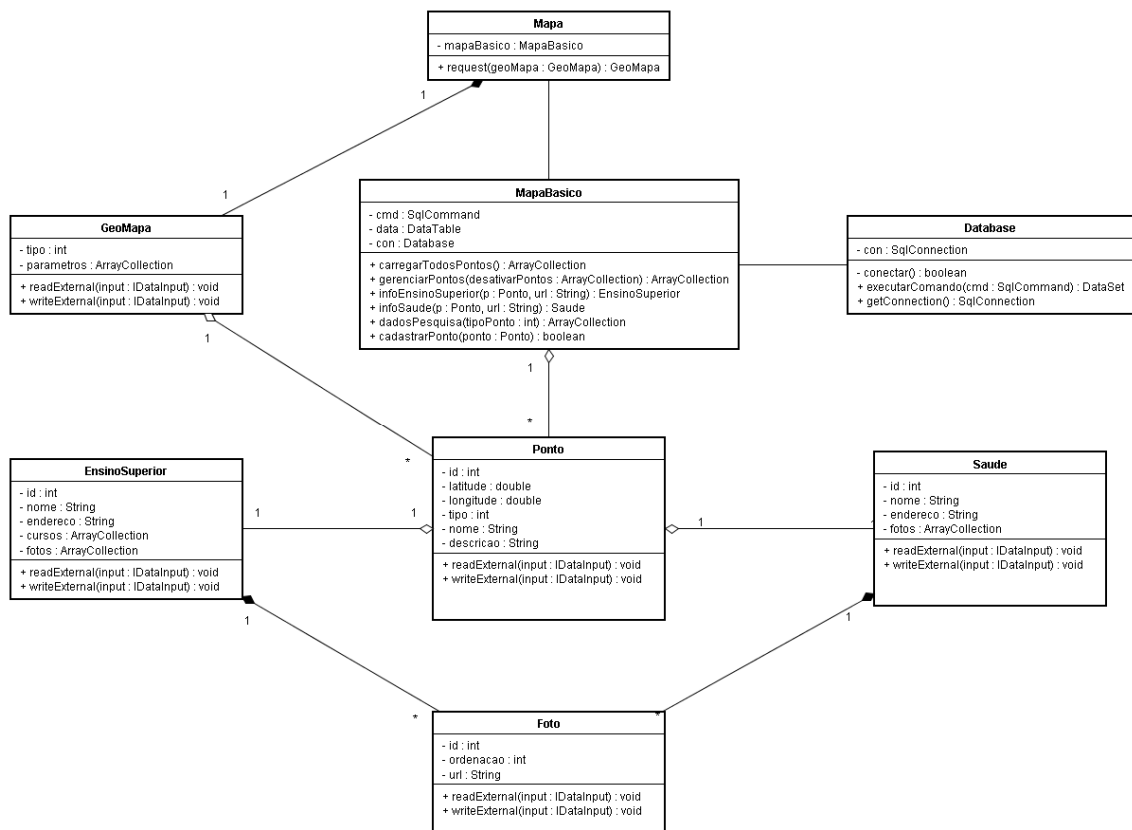


Figura 7 - Diagrama de Classe (Servidor)

Na representação da Figura 7, é possível distinguir a diferença entre os diagramas do lado cliente e servidor, pois possui classes voltadas para a manipulação dos serviços de acesso à dados, ao contrário do diagrama do lado do cliente, que possui classes para manipulação da interface com o usuário.

É comum, a ambos os lados, a classe utilizada para manter o status do mapa chamada *GeoMapa*, em circunstâncias das informações contidas neste objeto serem necessárias aos dois níveis de processamento abordados.

4.4. DIAGRAMA DE SEQÜÊNCIA

Um diagrama de seqüência é formulado por um conjunto de objetos, nele é demonstrado as comunicações necessárias entre os objetos, para que possa ser executado os processos do sistema. Por meio de uma linha de tempo estes diagramas descrevem a seqüência de comunicação entre os objetos (STADZISZ, 2002, p. 29).

A partir desta definição, foram elaborados os seguintes diagramas de seqüência, para os casos de uso anteriormente definidos.

Caso de Uso zoom no mapa é ilustrado na Figura 8.

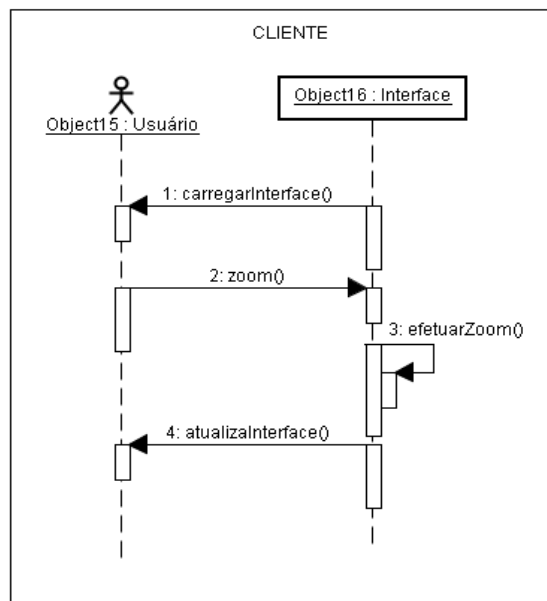


Figura 8 - Diagrama de Seqüência para zoom no mapa

A descrição deste caso de uso enfatiza que após o carregamento da interface para o usuário o mesmo deve efetuar o comando de zoom que é processado pela própria interface e envia a atualização gráfica ao usuário.

A Figura 9 ilustra o Diagrama de Seqüência do Caso de Uso mover mapa.

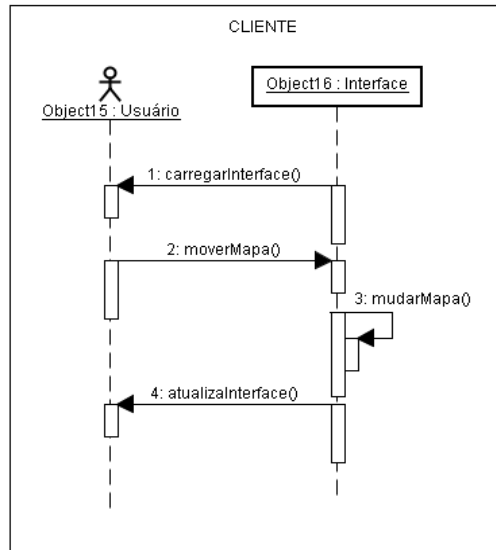


Figura 9 - Diagrama de Seqüência para mover mapa

Neste diagrama o usuário, após o carregamento da interface do mapa, movimenta o mesmo de acordo com a sua necessidade, é enviada uma requisição para a interface que processa a nova localização e disponibiliza a interface atualizada ao usuário.

Na Figura 10 está ilustrado o Diagrama de Seqüência do Caso de Uso modos de visualização.

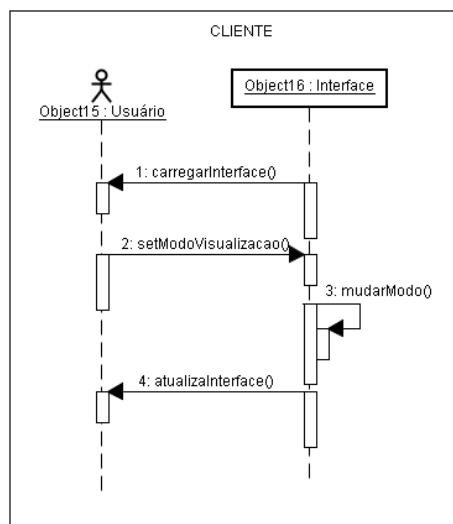


Figura 10 - Diagrama de Seqüência para manipular os modos de visualização

Com o carregamento da interface o usuário pode acionar o modo de visualização que desejar, esta opção é enviada para a interface que processa o modo selecionado e disponibiliza a interface atualizada ao usuário.

Nos Diagramas de Seqüência apresentados, pode-se destacar o fato dos mesmos interagirem apenas com a interface, isso como citado anteriormente, é possível por meio da tecnologia RIA, que centraliza a aplicação do lado do cliente, tornando muito mais rápido o processamento do aplicativo.

A seguir, teremos a descrição de outras interações do sistema, mas com o diferencial da necessidade de acesso a recursos no lado do servidor.

A Figura 11 disponibiliza o Diagrama de Seqüência para o Caso de Uso cadastrar ponto.

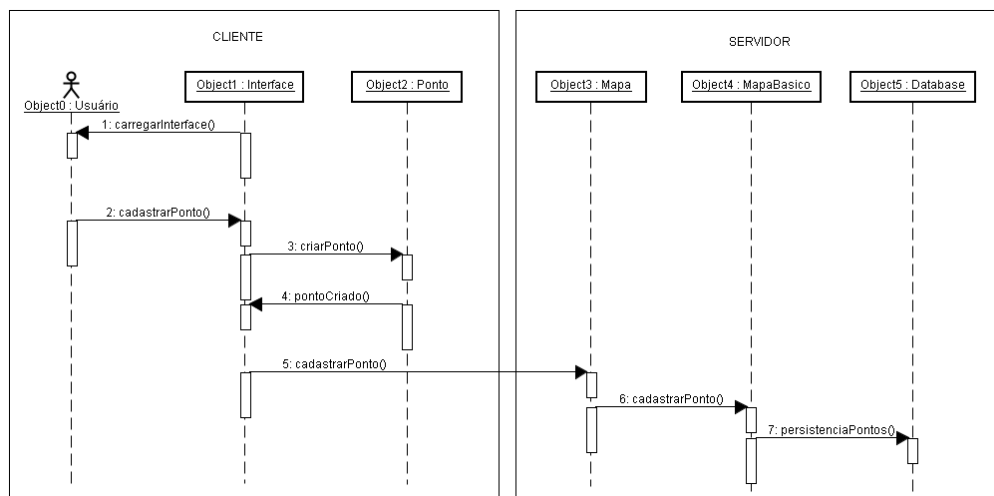


Figura 11 - Diagrama de Seqüência para cadastrar ponto

O diagrama acima descreve a comunicação necessária para cadastrar um ponto no mapa.

Após o carregamento da interface do mapa, o usuário efetuará o cadastro do ponto na tela, esta solicitação é enviada à interface, que por sua vez, cria um objeto *Ponto* e instância com os dados recebidos do usuário, na seqüência, envia a requisição do cadastro do ponto ao servidor interagindo com o objeto da classe *Mapa*, que faz uso da manipulação da classe *MapaBasico* e efetua a persistência do ponto na base de dados.

Na Figura 12 está ilustrado o Diagrama de Seqüência do Caso de Uso visualizar pontos.

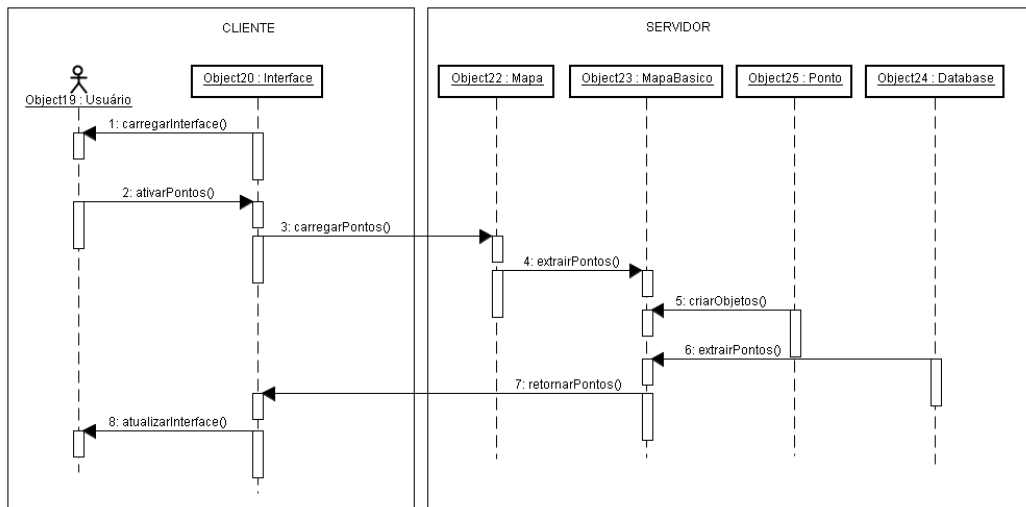


Figura 12 - Diagrama de Seqüência para visualizar os pontos do mapa

Na representação acima, o usuário, após o carregamento da interface, solicita a ativação dos pontos existentes no mapa, a interface por sua vez, envia essa requisição para a classe *Mapa* no lado servidor, a classe *Mapa* solicita a extração dos pontos a classe *MapaBasico* que cria os objetos *Ponto* e efetua a busca das informações na base de dados. Com as informações obtidas, o objeto *Ponto* é preenchido e enviado de volta para a interface no lado cliente, que disponibiliza a visualização dos pontos ao usuário.

A Figura 13 exibe o Diagrama de Seqüência para o Caso de Uso extrair informação.

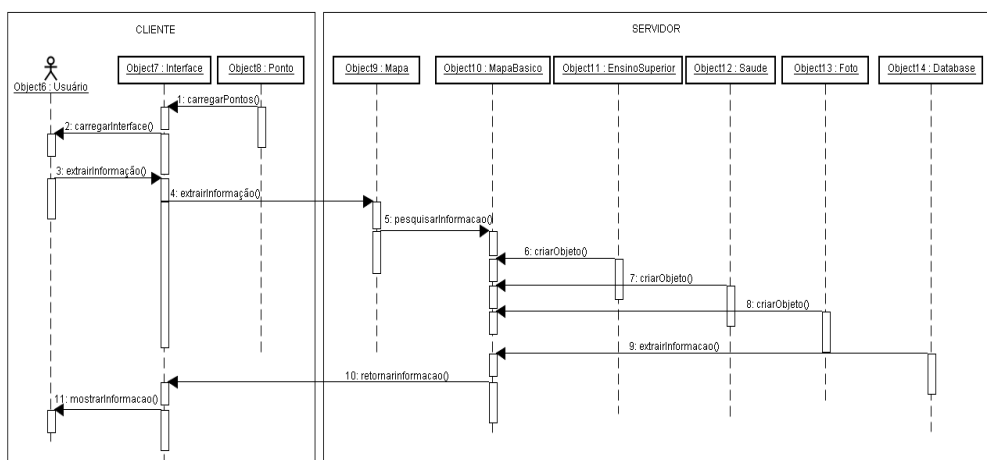


Figura 13 - Diagrama de Seqüência para extrair informação

Neste diagrama, o processo para visualizar informações sobre os pontos existentes no mapa é descrito, após o carregamento do mapa e dos pontos existentes, o usuário efetua a requisição da informação de um dos pontos que foram exibidos a ele, a interface envia a solicitação para a classe *Mapa* no lado do servidor, que por sua vez solicita a informação a classe *MapaBasico*, esta cria um objeto do tipo *EnsinoSuperior* ou *Saude* e um objeto *Foto* e efetua a pesquisa das informações na base de dados, preenche os objetos anteriormente criados e devolve a informação para a interface no lado do cliente, que disponibiliza a mesma ao usuário.

4.5. DIAGRAMA DE INTERFACE

O fato das aplicações RIA destacarem uma melhor experiência de navegação e interatividade ao usuário formulou a iniciativa de estar desenvolvendo um Diagrama de Interface para designar perante a modelagem do aplicativo, o formato de visualização e poder destacar a forma de interações do aplicativo desenvolvendo um modelo visual.

Para atingir os objetivos propostos neste trabalho, está descrito no diagrama seguinte, a interface da aplicação, enfatizando as possíveis interações devido ao uso de RIA e a maior facilidade de apresentação de informações que esta tecnologia proporciona.



Figura 14 - Diagrama de Interface

A interface ilustrada acima definiu o posicionamento do mapa, definiu também, a disposição do filtro dos níveis de ponto. O filtro deve utilizar o conceito de arrastar e soltar para poder manipular a ativação e desativação dos níveis de ponto.

Na ativação da interface de pesquisa, o mapa deve deslocar-se sobre o menu de filtro, escondendo o mesmo e passando a disponibilizar do seu lado esquerdo o menu de pesquisas, como pode ser visualizado na Figura 15.

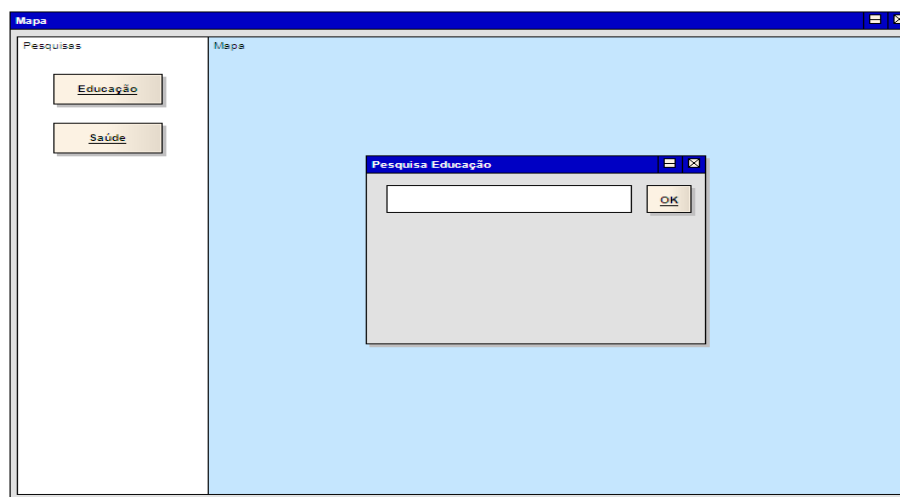


Figura 15 - Interface de Pesquisas

Ao selecionar uma pesquisa, o sistema deverá exibir uma janela com as opções, esta janela deve disponibilizar botões para minimizar, maximizar e fechar, assim como uma janela em uma aplicação *Desktop*, o filtro das pesquisas deve fornecer a funcionalidade de auto completar.

Na fase de visualização de informações, o mesmo conceito de janelas deve ser utilizado.

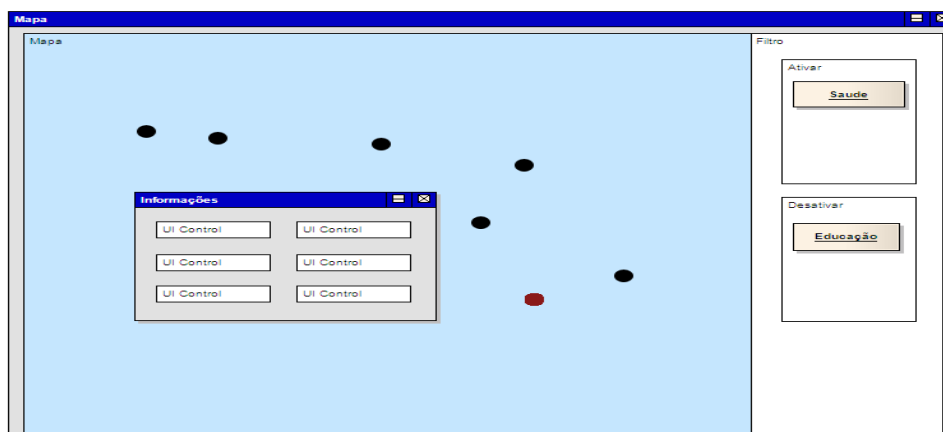


Figura 16 - Informações de um ponto

5. IMPLEMENTAÇÃO DO EXPERIMENTO

Importante atenção é dada a este capítulo, pois nesta seção são desenvolvidos os objetivos do trabalho proposto.

O aplicativo proposto, foi implementado utilizando as tecnologias Flex e .NET. Tal integração foi possível pela existência de um *framework*, chamado FluorineFX que atua como uma ponte entre as duas aplicações.

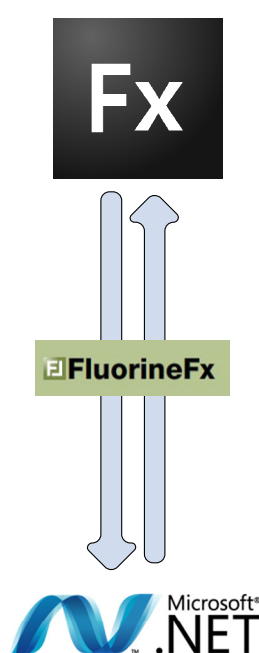


Figura 17 - Integração das tecnologias Flex e .NET

Foi necessário um estudo prévio do *framework* FluorineFX, para entender a maneira de estar utilizando-o no aplicativo. Este *framework* atua no transporte dos objetos entre as tecnologias escolhidas. Devido a este fato, foi necessário o estudo e implementação de serialização aos objetos que poderiam ser transportados entre as tecnologias abordadas.

“Serialização é o processo de conversão do estado de um objeto, em uma forma que possa ser persistida ou transportada” (MSDN). A Figura 18 e a Figura 19 demonstram a implementação de uma classe serializada em ActionScript e .NET respectivamente.

```

public class GeoMapa implements IExternalizable
{
    private var _tipo:int;
    private var _parametros:ArrayCollection;

    public function get Tipo():int { return _tipo; }
    public function set Tipo(value:int):void { _tipo = value; }

    public function get Parametros():ArrayCollection { return _parametros; }
    public function set Parametros(value:ArrayCollection):void { _parametros = value; }

    public function GeoMapa():void
    {
        _parametros = new ArrayCollection();
    }

    public function readExternal(input:IDataInput):void
    {
        _tipo = input.readInt();
        _parametros = input.readObject() as ArrayCollection;
    }

    public function writeExternal(output:IDataOutput):void
    {
        output.writeInt(_tipo);
        output.writeObject(_parametros);
    }
}

```

Figura 18 - Classe serializada em ActionScript

Toda classe que necessitar ser serializada nestas duas plataformas, deve estender a interface *IExternalizable* e implementar os métodos *ReadExternal* e *WriteExternal*, que são responsáveis pela leitura e escrita do objeto no momento do transporte entre as linguagens.

```

public class GeoMapa : IExternalizable
{
    private int _tipo;
    private ArrayCollection _parametros;

    public int Tipo
    {
        get { return _tipo; }
        set { _tipo = value; }
    }

    public ArrayCollection Parametros
    {
        get { return _parametros; }
        set { _parametros = value; }
    }

    #region IExternalizable Members
    public void ReadExternal(IDataInput input)
    {
        _tipo = input.ReadInt();
        _parametros = (ArrayCollection)input.ReadObject();
    }

    public void WriteExternal(IDataOutput output)
    {
        output.WriteInt(_tipo);
        output.WriteObject(_parametros);
    }

    #endregion
}

```

Figura 19 - Classe serializada em .NET

Após este estudo, foi possível definir as etapas de desenvolvimento do experimento, que são descritas na Figura 20.

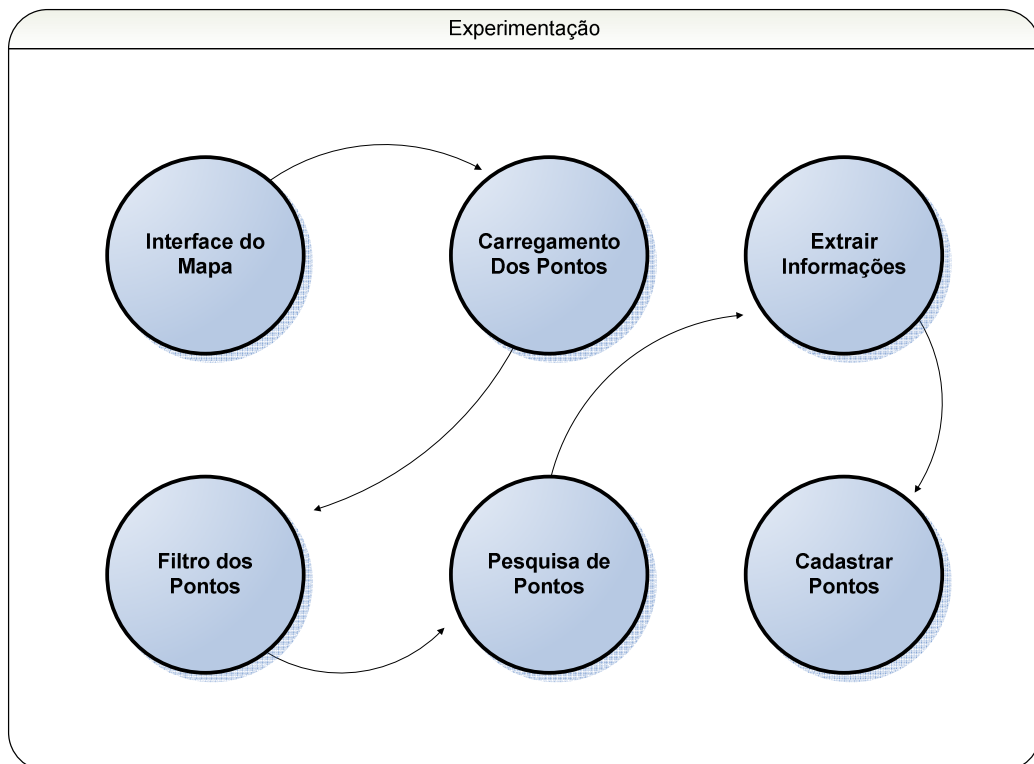


Figura 20 - Etapas de Desenvolvimento

5.1. INTERFACE DO MAPA

Para o desenvolvimento da interface do mapa, ficou bem claro o destaque sobre o conceito de RIA que descreve a centralização do aplicativo no lado Cliente. O fato da API de mapa ser nativa ao Flex, fez com que o carregamento do mapa fosse executado totalmente à partir do navegador do cliente. O carregamento do mapa é feito pela utilização da classe *Map*, disponibilizada pela API da Google, nela é inserido a chave que a Google disponibiliza para poder visualizar o mapa.

```

map = new Map();
map.addListener(MapEvent.MAP_READY, onMapReady);
map.key = "ABQIAAAASskX8LTxgJ_au3w9yEe_ohQrg_rpI06L26Fgvj1Q0ZNA0rQTohSxGiZm50j0EBPHm7QrQtEy40t5XA";
  
```

Figura 21 - Criação do mapa

À partir das instruções citadas anteriormente, ao adicionar o mapa na interface, já obtemos a sua visualização, ainda assim, é possível manipular o posicionamento e adicionar alguns comandos básicos, como o controle de Zoom, os modos de visualização e o comando para mover o mapa, ilustrados na Figura 22.

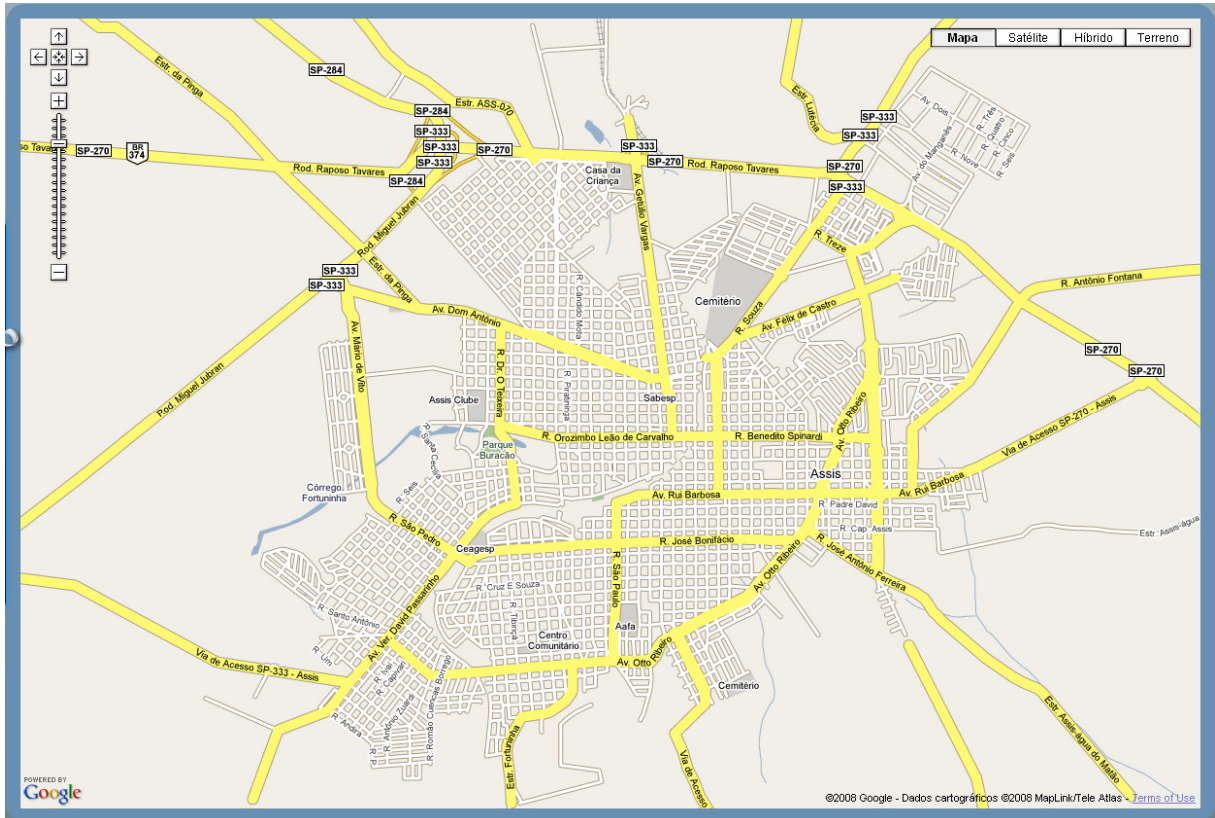


Figura 22 - Interface do Mapa

5.2. CARREGAMENTO DOS PONTOS

Na fase de carregamento dos pontos, houve o primeiro contato com o lado Servidor, e por sua vez, a necessidade de utilizar serialização de objetos, citada anteriormente. Para a possibilidade de comunicação é necessário fazer um mapeamento da classe que deseja acessar no lado servidor, isso é feito utilizando um *RemoteObject*, ou seja, objeto remoto. Após esse mapeamento, é possível efetuar a chamada de um método no lado servidor, à partir daí, com esse mapeamento é possível invocar métodos no lado do servidor e enviar objetos para serem processados.

```
var remote:RemoteObject;
remote = new RemoteObject();

remote.destination = "fluorine";
remote.source = "geoprocessamento.API.Mapa";
remote.request();
```

Figura 23 - Chamada de método no servidor

Assim, tornou-se possível a requisição dos pontos que estão gravados na base de dados, e por meio da manipulação de *ActionScript* foram criados marcadores com a API de mapas que representam os pontos na tela.

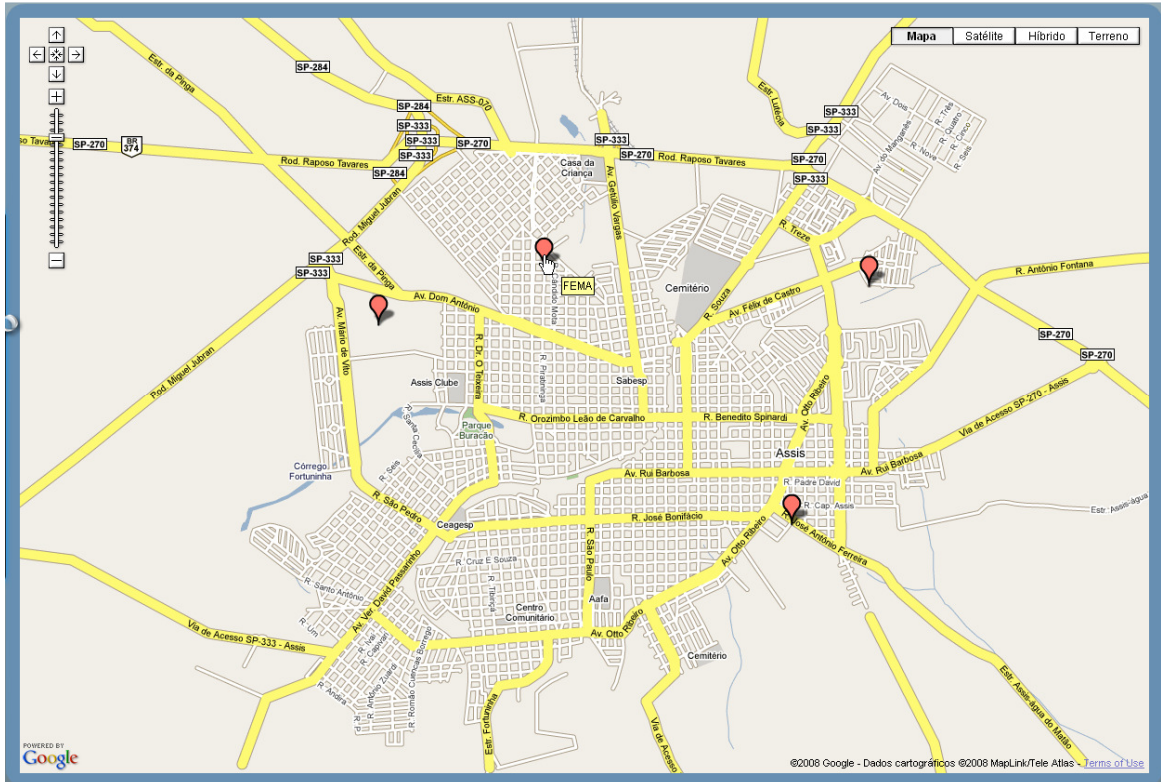


Figura 24 - Pontos carregados no mapa

5.3. FILTRO DOS PONTOS

Na criação do filtro de pontos foi utilizado o recurso interativo conceituado em RIA de arrastar e soltar objetos.

O Flex proporciona isso de maneira muito fácil, existe a propriedade *dragEnabled*, que faz com que os objetos referenciados passem à aceitar a função de arrastar e soltar. Assim, de forma interativa, o filtro pode ser aplicado e demonstrar somente o tipo de itens que estiver no quadro Ativo.

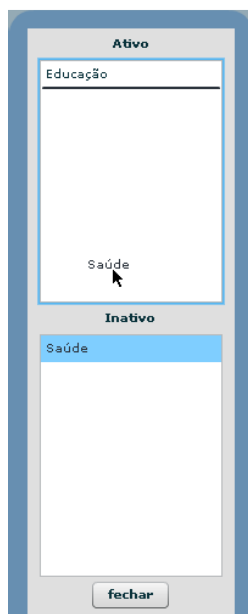


Figura 25 - Filtro do tipo de ponto

5.4. PESQUISA DE PONTOS

Para a pesquisa de Pontos, foi criada uma caixa de texto que disponibiliza o recurso de auto completar, sempre encontrado em aplicações *Desktop*. Após o carregamento dos dados é possível aplicar uma função chamada *filterFunction* que recebe um método para a realização de filtros na fonte de dados do componente, desta forma, a função de auto completar foi implementada e designada na propriedade *filterFunction* da caixa de texto. Com isso a caixa de texto inicialmente recebe todos os valores, à partir da digitação do usuário ela vai filtrando e mostrando uma lista de itens que contenham as letras digitadas ao usuário.

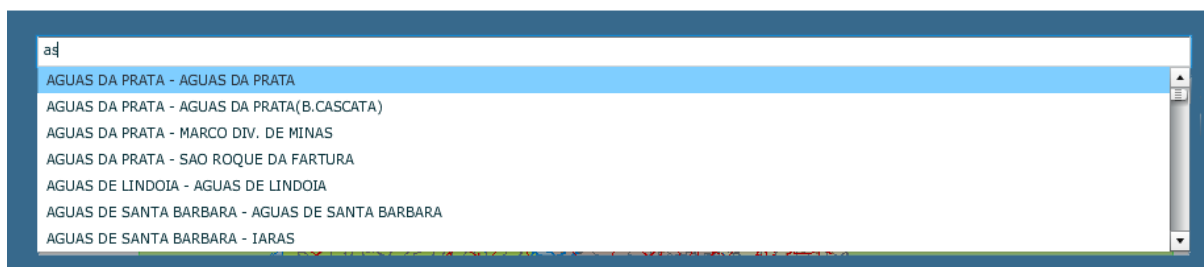
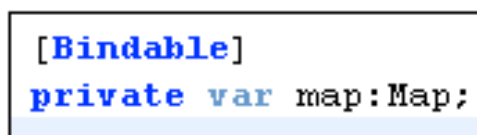


Figura 26 - Função auto completar

5.5. EXTRAIR INFORMAÇÕES

Nesta fase, após o clique sobre um ponto, é feito a busca dos dados referentes no servidor. Para o preenchimento desses dados na interface foi utilizada a *metadata tag* [*Bindable*] na plataforma Flex, que consiste em enviar uma instrução ao compilador que gera o código necessário para sua propriedade ser a fonte de dados para outras propriedades (ADOBE, 2007), um atributo referenciado com esta instrução pode tornar-se uma fonte de dados a outros atributos e no momento que ele for atualizado, automaticamente, os atributos que o tenham utilizado como fonte de dados, recebem a atualização.

A screenshot of code within a rectangular box. The code consists of two lines: the first line is "[Bindable]" and the second line is "private var map:Map;". The text is in a monospaced font, with the first line in blue and the second line in black. The box has a light blue background and a black border.

```
[Bindable]
private var map:Map;
```

Figura 27 - Uso de *Bindable*

Dessa forma os componentes que recebem os dados, foram referenciados a uma propriedade *Bindable* e automaticamente, quando é preenchida, reflete para todos os componentes.

Na interface foi utilizado o componente *FlexMDI*, que fornece um gerenciador e criador de janelas semelhantes as das aplicações *Desktop*, as janelas possuem comandos para maximizar, minimizar, fechar, organizar janelas e agrupar em cascata. Essas funcionalidades têm destaque importante, pelo fato de ser possível gerenciar uma quantidade maior de informações em uma tela

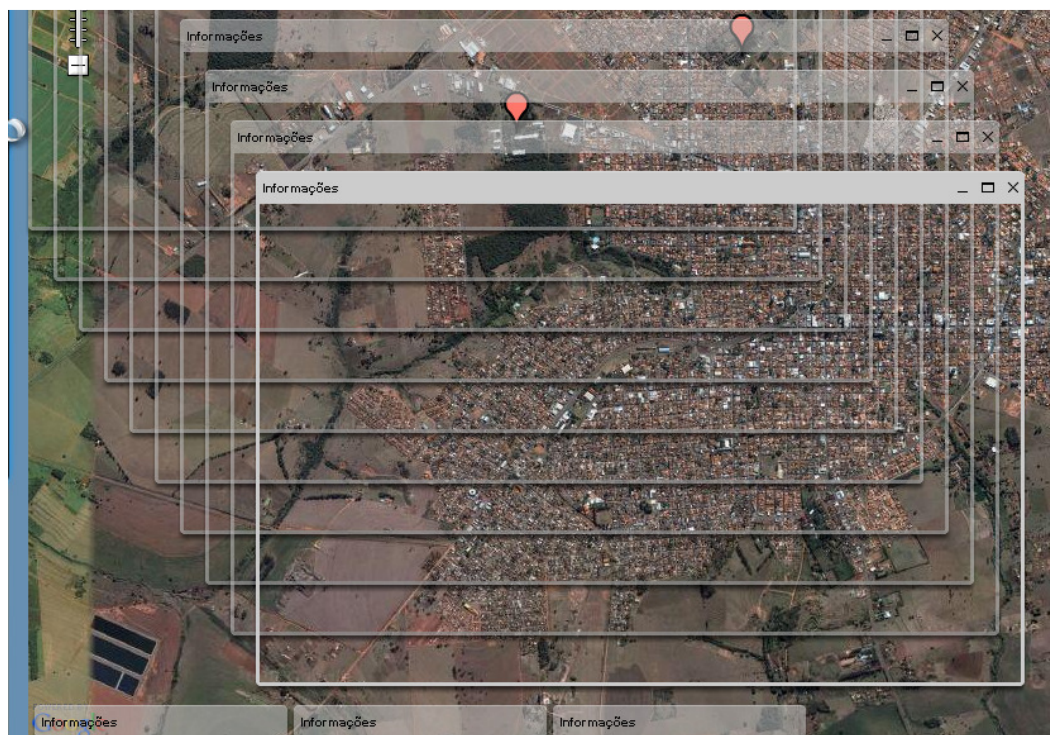


Figura 28 - Gerenciamento de janelas

5.6. CADASTRAR PONTOS

No cadastramento dos pontos foi utilizado o recurso de *stored procedure*, estas são instruções SQL previamente compiladas e guardadas na base de dados (TECHFAQ), que facilitam muito o desenvolvimento, pois, ficam localizadas na base de dados e não de forma estática dentro do aplicativo, tornando mais fácil a manutenção do sistema. Ao efetuar um clique em um ponto na tela, foi necessário capturar as coordenadas com a utilização da API de mapa e enviar ao servidor para que o mesmo persistisse esse ponto na base de dados com o recurso citado acima.

6. CONCLUSÃO

Realmente a tecnologia Flex mostrou ser inovadora e dispor de muitos recursos interativos úteis ao desenvolvimento, estes recursos atendem a finalidade de exibição de informações muito bem e com bastante diversidade.

A parte de efeitos dos componentes deixou bem clara a elevação da satisfação visual que um aplicativo desenvolvido com tecnologia RIA pode proporcionar.

A integração entre as tecnologias Flex e .NET, atendeu de forma eficaz as necessidades do aplicativo e demonstrou ser de fácil configuração e implementação, proporcionando grande agilidade na busca de informações no servidor.

Porém, um ponto negativo nessa integração, foi o fato da duplicação das entidades e o uso de serialização, pois, para haver a transferência de objetos entre as tecnologias abordadas, é necessário que em ambos os lados tenham a classe implementada, o que gera uma quantidade de linhas codificadas maior. Na parte da serialização destes objetos, devem necessariamente ter a descrição dos atributos na mesma ordem, tanto na linguagem *ActionScript* quanto na linguagem *C#*, no caso de classes que possuam muitos atributos, fica suscetível a erros de desenvolvimento.

Como sugestão de trabalhos futuros, fica o estudo da possibilidade de sanar as dificuldades descritas acima e a continuação do desenvolvimento do aplicativo neste trabalho proposto, desenvolvendo melhorias e novas formas de interação ao usuário.

REFERÊNCIAS BIBLIOGRÁFICAS

ADOBE. **Flex overview**. Visão geral do Flex: 30 mar. 2007. Disponível em: <<http://www.adobe.com/products/flex/overview/>>. Acesso em: 23 jun. 2008.

ALLAIRE, Jeremy. Macromedia Flash MX - A next-generation rich client. **Macromedia White Paper**⁶. Mar 2002.

FAIN, Yakov; RASPUTNIS, Dr. Victor; TARTAKOVSKY, Anatole. **Rich Internet Applications with Adobe® Flex™ & Java™**. 1. ed. Woodcliff Lake: Editora SYSCON Books, 2007.

FLUORINEFX. **Documentação**. Disponível em: <<http://www.fluorinefx.com/docs/fluorine/index.html>>. Acessado em: 27 jun. 2008.

GARRETT, Jesse James. **Ajax: A New Approach to Web Applications**. 18 fev. 2005. Disponível em: <<http://www.adaptivepath.com/ideas/essays/archives/000385.php>>. Acessado em: 23 jun. 2008.

GOOGLE. **Documentação da API de Mapas para Flash**. Disponível em: <<http://code.google.com/apis/maps/documentation/flash/intro.html>>. Acessado em: 27 jun. 2008.

MDC (Mozilla Developer Center). **Core JavaScript 1.5 Guide:JavaScript Overview**. Documentação. 9 dez. 2007. Disponível em: <http://developer.mozilla.org/en/docs/Core_JavaScript_1.5_Guide:JavaScript_Overview>. Acessado em: 23 jun. 2008.

MORITZ, Florian. **Rich Internet Applications (RIA): A Convergence of User Interface Paradigms of Web and Desktop Exemplified by JavaFX**. 2008. 140 p. Trabalho de Conclusão de Curso - Ciências da Computação e Tecnologia de micro-sistema - Universidade de Ciências Aplicadas Kaiserslautern. Zweibrücken, Alemanha, 2008.

MSDN. **Biblioteca MSDN**. Disponível em: <<http://msdn.microsoft.com/en-us/library/default.aspx>>. Acessado em: 26 jun. 2008.

⁶ Documento técnico de divulgação

OPENLASZLO. **Documentação**. Disponível em: <<http://www.openlaszlo.org/>>. Acessado em: 25 jun. 2008.

O'REILLY, Tim. **Web 2.0 Compact Definition: Trying Again**. 10 dez. 2006 Disponível em <<http://radar.oreilly.com/archives/2006/12/web-20-compact-definition-tryi.html>>. Acessado em: 23 jun. 2008.

Stadzisz, Paulo César. **Projeto de Software usando a UML**. 2002. Centro Federal de Educação Tecnológica do Paraná.

SUN Microsystems. **Visão geral da tecnologia JavaFX**. 2008. Disponível em: <<http://www.javafx.com/>>. Acessado em: 25 jun. 2008.

TECHFAQ. **What is a stored procedure?**. Disponível em: <<http://www.techfaq.com/stored-procedure.shtml>>. Acessado em 28 out. 2008.

W3C (*World Wide Web Consortium*). **Padrões Web**. Disponível em: <<http://www.w3.org/>>. Acessado em: 23 jun. 2008.