FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Reengineering Web Applications to Standalone Rich Internet Applications

Steeve Alves Ferreira

Mestrado Integrado em Engenharia Informática e Computação

Orientador: Ademar Aguiar (PhD)

14 de Junho de 2010

Reengeniering Web Applications to Standalone Rich Internet Applications

Steeve Alves Ferreira

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: João Cardoso (PhD)

Vogal Externo: Alberto Silva (PhD)

Orientador: Ademar Aguiar (PhD)

---

14 de Junho de 2010

Resumo

A Web encontra-se em constante mudança e à Web actual, já cheia de interactividade, cada vez mais se pretende adicionar características das aplicações Desktop Às aplicações web que surgem com estas características dão-se o nome de Rich Internet Application (RIA).

Actualmente as limitações tecnológicas ainda restringem as funcionalidades possíveis das aplicações web que correm dentro de um navegador web, existindo a possibilidade de migrar a aplicação web para fora do Browser

Este documento ilustra os tipos de aplicações ricas para a internet, mostrando o que cada é possível efectuar dentro e fora do Browser com as tecnologias actuais, dando maior enfase às aplicações fora do Browser Com esta dissertação foi construída uma aplicação de RIA Desktop para a aplicação Web escolinhas.pt.

É efectuada uma análise às tecnologias do ponto de vista de funcionalidades com maior enfase nas funcionalidades importantes para o projecto, sendo escolhida como tecnologia o Silverlight.

São descritos os dois tipos de RIA Desktop, tipo cliente Desktop simples e tipo cliente Desktop com Browser dedicado. O processo de desenvolvimento dos dois tipos de RIA Desktop é descrito com algum pormenor, analisando a parte da construção da interface, persistência de dados e comunicação com a aplicação Web. É descrito igualmente as alterações necessárias a serem efectuadas do lado da aplicação Web.

A implementação de uma RIA Desktop do tipo cliente Desktop com Browser dedicado, apresenta vantagens, mas a sua implementação com as tecnologias actuais, apesar de possível com limitações, não é muito facilitada.

Abstract

The Web is constantly changing, and the current web, already full of interactivity, increasingly wants to add features to it of desktop applications. Web applications that come with these features have the name RIA or rich internet application.

At present technological limitations still restrict the possible features of Web applications that run inside a browser, with the existing possibility of releasing the application from the Web browser.

This document illustrates the kinds of rich internet applications, showing what each might be possible within and outside of the browser with current technology, giving greater emphasis to the applications outside the browser. This essay was built a RIA desktop for the Web application escolinhas.pt.

This work has an examination of technology in terms of features with emphasis on features important to the project. In the end the Silverlight technology was chosen as.

Here are described two types of RIA desktop, desktop client type and desktop client with dedicated browser type. The process of developing these two types of RIA desktop is described in some detail by analysing the construction part of the interface, persistence and data communication with the web application. It is described equally the witch necessary changes are made on the web server.

The implementation of a RIA desktop with dedicated browser, has advantages, but its implementation with current technology, although possible with limitations, is not that easy.

Agradecimentos

Aproveito este espaço para agradecer em primeiro lugar ao meu orientador o Prof. Dr Ademar Aguiar.

Os meus sinceros agradecimentos para o Mestre Nuno Baldaia da empresa Tecla Colorida pelo apoio e preocupação no meu trabalho.

Por fim gostaria de agradecer a todos os que de forma directa ou indirecta contribuíram para a escrita desta dissertação e no desenvolvimento do projecto.

Índice

1 Introdução **1**
1.1 Contexto/Enquadramento 1
1.2 Projecto 1
1.3 Motivação 2
1.4 Objectivos 2
1.5 Estrutura da Dissertação 3

2 Estado da Arte **5**
2.1 Introdução 5
2.2 Evolução dos Browsers e da Web 7
2.2.1 WorldWideWeb 8
2.2.2 Erwise 8
2.2.3 MidasWWW 8
2.2.4 Mosaic 8
2.2.5 Arena 9
2.2.6 Netscape Navigator 9
2.2.7 Microsoft Internet Explorer 9
2.2.8 Opera 10
2.2.9 Amaya 11
2.2.10 Safari 11
2.2.11 Firefox 11
2.2.12 Chrome 11
2.2.13 Versões da Web 12
2.3 Capacidades de aplicações Desktop impossíveis na Web 14
2.4 Surgimento das Rich Internet Applications 14
2.5 Migração de Desktop para a Web 16
2.6 Migração da Web para o Desktop 20
2.7 HTML5 21
2.8 Tecnologias disponíveis 22
2.8.1 Ajax 22
2.8.2 Web Based 23
2.8.2.1 Adobe Flash 23
2.8.2.2 Adobe Flex 26

2.8.2.3 Microsoft Silverlight ... 26
2.8.2.4 Sun JavaFX ... 28
2.8.2.5 Visual WebGui ... 29
2.8.2.6 OpenLaszlo ... 32
2.8.2.7 Google Gears ... 33
2.8.3 Desktop Based ... 34
2.8.3.1 Adobe AIR ... 34
2.8.3.2 WebKit ... 35
2.8.3.3 Firefox, Chrome, Safari e Opera ... 36
2.8.3.4 WPF (Windows Presentation Foundation) ... 37
2.8.4 Comparação entre tecnologias ... 37

3 Revisão de tecnologias ... 39
3.1 Necessidades tecnológicas para este projecto ... 39
3.2 Adobe Air ... 40
3.3 Microsoft Silverlight ... 40
3.4 Web Kit ... 40
3.5 WPF ... 41
3.6 Conclusões ... 41

4 Abordagens Out Of the Browser (OOB) ... 43
4.1 RIA OOB tipo cliente desktop ... 43
4.1.1 Definição ... 43
4.1.2 Exemplos ... 44
4.1.2.1 Ebay Desktop ... 44
4.1.2.2 Microsoft Silverlight 4 Client for Facebook ... 46
4.1.3 Processo de desenvolvimento da RIA ... 47
4.1.3.1 Ambiente de desenvolvimento ... 48
4.1.3.2 Persistência ... 48
4.1.3.3 Comunicação Web ... 49
4.1.4 Reengenharia da aplicação Web ... 50
4.1.5 Necessidades/opções tecnológicas ... 51
4.1.6 Conclusões da abordagem ... 51
4.2 Ria OOB tipo “Browser Dedicado” ... 51
4.2.1 Definição ... 51
4.2.2 Exemplos ... 52
4.2.3 Processo de desenvolvimento da RIA ... 53
4.2.3.1 Construção da interface ... 53
4.2.3.2 Offline ... 54
4.2.4 Reengenharia da aplicação Web ... 56
4.2.5 Necessidades/opções tecnológicas ... 56
4.2.6 Conclusões da abordagem ... 57

5 Escolinhas Desktop ... 59
5.1 Contexto ... 59
5.2 Requisitos Funcionais ... 59

5.2.1 Submissão offline de ficheiros para o escolinhas 59
5.2.2 Criação de trabalhos utilizando ferramentas do Microsoft Office 60
5.2.3 Edição de trabalhos do tipo Microsoft Office 60
5.2.4 Submissão de vários ficheiros em simultâneo e em segundo plano para o escolinhas por drag and drop 61
5.2.5 Acesso à aplicação em offline 61
5.2.6 Acesso a toda a interface Web 62
5.3 Requisitos suplementares 62
5.4 Arquitectura Lógica 62
5.5 Arquitectura Tecnológica 65
5.6 Arquitectura Física 67
5.6.1 Máquina cliente 68
5.6.2 Máquina Servidor Web 68
5.6.3 Máquina Servidor Base de Dados 68
5.7 Protótipo 68
5.7.1 Submissão offline de ficheiros para o escolinhas 68
5.7.1.1 Interface 68
5.7.1.2 Decisões de desenho 69
5.7.2 Criação de trabalhos utilizando ferramentas do Microsoft Office 69
5.7.2.1 Interface 70
5.7.2.2 Decisões de desenho 70
5.7.3 Edição de trabalhos do tipo Microsoft Office 70
5.7.3.1 Interface 70
5.7.3.2 Decisões de desenho 71
5.7.4 Submissão de vários ficheiros em simultâneo e em segundo plano para o escolinhas por drag and drop 71
5.7.4.1 Interface 71
5.7.4.2 Decisões de desenho 72
5.7.5 Acesso à aplicação em offline 72
5.7.5.1 Interface 72
5.7.5.2 Decisões de desenho 73
5.7.6 Acesso a toda a interface Web 74
5.7.6.1 Interface 74
5.7.6.2 Decisões de desenho 74
5.7.7 Outras decisões de desenho 74
5.7.7.1 Autenticação 75
5.7.7.2 Imagens na interface Web 75
5.7.7.3 Detecção de modo offline 75
5.8 Dificuldades 75
5.9 Conclusões 76

6 Conclusões 77

Referências 79

Lista de Figuras

Figura 1: Classificação dos diversos tipos de RIA..... 6
Figura 2: Google Docs..... 17
Figura 3: Ecrã do sistema eyeos..... 19
Figura 4: Flash Player no YouTube..... 24
Figura 5: Exemplo de jogo desenvolvido em Flash..... 25
Figura 6: Ambiente de desenvolvimento de uma aplicação em Silverlight.....27
Figura 7: Ambiente de desenvolvimento de JavaFx..... 29
Figura 8: Ambiente de desenvolvimento da tecnologia VisualWebGui.....30
Figura 9: Exemplo de aplicação desenvolvida em VisualWebGui..... 31
Figura 10: Aplicação desenvolvida em OpenLaszlo executada na versão DHTML.....32
Figura 11: eBay Desktop, Exemplo de Aplicação desenvolvida em AIR a correr no Desktop…35
Figura 12: Ebay Desktop..... 45
Figura 13: Ebay Web..... 45
Figura 14: Microsoft Silverlight 4 Client for Facebook ecrã..... 46
Figura 15: Ecrã da versão Web da aplicação Facebook..... 47
Figura 16: Ecrã de Microsoft Expression Blend..... 48
Figura 17: Ecrã da aplicação iTunes..... 52
Figura 18: Ecrã da aplicação iTunes..... 53
Figura 19: Arquitectura lógica da aplicação Escolinhas Desktop..... 63
Figura 20: Exemplo de sequência de notificação da aplicação por parte de um evento na interface Web..... 64
Figura 21: Comunicação entre a aplicação e o motor de Javascript existente no Browser integrado representado sobre a a forma de diagrama de sequencia..... 65
Figura 22: Arquitectura Tecnológica do Escolinhas Desktop..... 66
Figura 23: Arquitectura Física do Escolinhas Desktop..... 67
Figura 24: Ecrã de Escolinhas Desktop: Drag and Drop de ficheiros..... 69
Figura 25: Ecrã Escolinhas Desktop: Criação de trabalho usando Microsoft Word.....70
Figura 26: Edição de um trabalho no escolinhas do tipo Word.....71
Figura 27: Submissão de um ficheiro em segundo plano enquanto se pinta no escolinhas.....72
Figura 28: Notificação de que o Escolinhas Desktop entrou em modo offline..... 73
Figura 29: Formulário de autenticação do Escolinhas Desktop..... 74

Lista de Tabelas

Tabela 1: Classificação de aplicações RIA de acordo com a distribuição do processamento.....7
Tabela 2: Principais lacunas da Web segundo Jeremy Allaire [ALL02].....16
Tabela 3: Sumário da tecnologia AJAX..... 23
Tabela 4: Sumário da tecnologia Flash..... 25
Tabela 5: Sumário da tecnologia Flex..... 26
Tabela 6: Sumário da tecnologia Silverlight..... 28
Tabela 7: Sumário da tecnologia JavaFX..... 29
Tabela 8: Sumário da tecnologia VisualWebGui..... 31
Tabela 9: Sumário da tecnologia OpenLaszlo..... 33
Tabela 10: Sumário da tecnologia Google Gears..... 34
Tabela 11: Sumário da tecnologia Adobe Air..... 35
Tabela 12: Sumário das vantagens da utilização do motor WebKit para a construção de uma RIA..... 36
Tabela 13: Sumário do uso de add-ons para Browsers..... 37
Tabela 14: Sumário da tecnologia WPF..... 37
Tabela 15: Comparação entre tecnologias Parte 1..... 38
Tabela 16: Comparação entre tecnologias Parte 2..... 38

Abreviaturas e Símbolos

AD	Aplicações Desktop
AIR	Adobe Integrated Runtime
AJAX	Asynchronous JavaScript and XML
API	Application Programming Interface
ASP	Active Server Pages
COM	Component Object Model
CSS	Cascate Style Sheet
DHTML	Dynamic HTML
DOM	Document Object Model
FEUP	Faculdade de Engenharia da Universidade do Porto
FTP	File Transfer Protocol
G.ho.st	Global hosted operating system
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
INESC	Instituto de Engenharia de Sistemas e Computadores
JPEG	Joint Photographic Experts Group
NCSA	National Center for Supercomputer Applications
NPAPI	Netscape Plugin Application Programming Interface
OO	Object Oriented
OOB	Out Of the Browser
PDA	Personal Digital Assistant
REST	Representational State Transfer
RIA	Rich Internet Application
RPC	Remote Procedure Call
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
SP2	Service Pack 2
SSL	Secure Sockets Layer
SVG	Scalable Vector Graphics
URI	Uniform Resource Identifier
VLC	VideoLAN Client
W3C	World Wide Web Consortio
WPF	Windows Presentation Foundation
VLC	VideoLAN Client
WPF	Windows Presentation Foundation
WSDL	Web Services Description Language
WYSIWYG	What You See Is What You Get
XHTML	eXtensible Hypertext Markup Language
XML	eXtensible Markup Language
XNA	XNA's Not Acronymed

1 Introdução

1.1 Contexto/Enquadramento

Este trabalho surge no âmbito do Mestrado Integrado em Engenharia Informática e Computação pela Faculdade de Engenharia da Universidade do Porto no ano lectivo de 2009 e 2010. Com esta dissertação tive a possibilidade de abordar dois grandes mundos da informática (Desktop e Web), procurando soluções para os aproximar.

Esta dissertação foi proposta pelo Professor Doutor Ademar Manuel Teixeira de Aguiar, e efectuada em ambiente empresarial no contexto da empresa Tecla Colorida [ESC10]. A empresa em questão é uma recente e pequena empresa de software que tem um produto inovador chamado "Escolinhas" [ESC10]. O "Escolinhas" é uma plataforma educativa orientada a crianças do ensino básico e já foi várias vezes considerado como um produto inovador e não só no mercado nacional.

A empresa Tecla Colorida pretende adicionar ao produto escolinhas.pt [ESC10] formas de interacção mais ricas à aplicação Web, formas de interacção actualmente ainda não possíveis com tecnologias "standard" dentro do Browser, e algumas em particular não possíveis dentro do Browser com nenhuma tecnologia actual. No capítulo 2 é apresentado em pormenor quais as novas interacções que são procuradas na Web e quais as tecnologias passíveis de as implementar.

1.2 Projecto

Este trabalho contempla a implementação de uma aplicação Desktop do produto "Escolinhas" com as principais funcionalidades de funcionamento offline da plataforma, integração com aplicações instaladas localmente e uma interação de *drag and drop* entre o sistema de ficheiros local e a aplicação.

1.3 Motivação

A simples motivação deste trabalho consiste em compreender as possibilidades tecnológicas que podem de alguma forma ir ao encontro das necessidades dos utilizadores, nas interacções com as aplicações Web actuais, de forma a ter a capacidade de responder de forma concreta com novas interacções de Desktop em aplicações Web, mas sempre ponderando o custo de desenvolvimento do lado do servidor e do cliente.

A análise do que a Web actualmente permite ou não fazer, dentro e fora do Browser, traz uma mais valia para os desenvolvedores de aplicações Web e uma vantagem para a evolução da Web como plataforma de aplicações.

1.4 Objectivos

A evolução recente da Web como plataforma de aplicações deveu-se em parte ao paradigma AJAX, como forma de aumentar as funcionalidades de uma Web estática e pouco ou nada interactiva. O AJAX é um conjunto de tecnologias que juntas permitiu superar as limitações do protocolo HTTP e adicionar características de interactividade até então só existentes nas aplicações Desktop. Esta capacidade da Web praticamente desconhecida até ao surgimento das aplicações Gmail e Google Maps da empresa Google veio revolucionar a Web e a interacção com o utilizador, criando uma web mais interactiva e rica em termos de interface com o utilizador também denominada Web 2.0.

Do ponto de vista tecnológico o AJAX e a Web 2.0 já eram possíveis antes de estas surgirem, pois praticamente todos os Browsers aquando do surgimento das primeiras aplicações Web já tinham implementado o objecto XMLHttpRequest [XREQ09], assim como uma linguagem de cliente Javascript.

Neste sentido pode-se dizer que a Web 2.0 adicionou características à Web inesperadas na altura, ou seja a tecnologia superava o que se pretendia da Web.

Actualmente e praticamente desde o surgimento da Web 2.0 a tendência tem sido para utilizar a tecnologia existente, por vezes de forma muito astuta, ou então quando a tecnologia standard não permite utilizar tecnologias de RIA não standard, sempre com o intuito de adicionar mais padrões de interacção característicos de Desktop à Web. Podemos assim concluir que as limitações da Web actualmente são tecnológicas, e tal pode ser comprovado pelo grande numero de tecnologias não standard de RIA actualmente existente que surgem para complementar.

O objectivo assim será o estudo das tecnologias e do processo de reengenharia necessário à implementação de funcionalidades Desktop ainda pouco utilizadas na Web mas de grande valor para o utilizador e consequentemente para produtos web.

Este trabalho utiliza como referencia bibliográfica fundamental o trabalho da tese de Licínio Mano [LIC09] e o livro “Ajax Design patterns” [AJXPTR]. Estas referencias bibliográficas foram fundamental como base para compreender o fenómeno da Web 2.0 e da inserção de interacções Desktop na Web.

1.5 Estrutura da Dissertação

Para além da introdução, esta dissertação contém mais 5 capítulos.

No capítulo 2, é descrito o estado da arte e são apresentadas de forma resumida as tecnologias de RIA existentes actualmente. No capítulo 3 são apresentadas as tecnologias candidatas à implementação das funcionalidades necessárias ao projecto. No capítulo 4 é feita a distinção entre duas abordagens encontradas em aplicações Web Standalone, assim como apresentado o processo de desenvolvimento da aplicação cliente e da reengenharia da aplicação Web. No capitulo 5 é apresentado em detalhe o projecto desenvolvido para a plataforma escolinhas.pt. Por fim no capitulo 6 é feita uma conclusão da dissertação.

2 Estado da Arte

Neste capítulo é descrito o estado da arte e são apresentados trabalhos relacionados de forma a mostrar o que existe no mesmo domínio e quais os problemas encontrados.

2.1 Introdução

As aplicações ricas para a internet ou *Rich Internet Application* ou simplesmente RIA são um misto entre aplicações Web e aplicações Desktop, podendo ser mais de uma ou mais de outra, mas mantendo sempre uma parte de ambos os mundos.

As aplicações que tipicamente envolvem um servidor Web e um cliente Desktop podem-se distinguir de acordo com a quantidade de processamento existente no cliente e no servidor. Como extremos teremos assim as aplicações unicamente a serem processadas pelo cliente (Aplicações Desktop) e as aplicações unicamente processadas pelo Servidor (Páginas e sítios Web tradicionais). Da mistura destes extremos surge então o conceito de RIA.

Desta forma podemos dividir as RIA em vários grupos, classificando-as de acordo com a quantidade de Desktop e de Web que estas têm. Aplicações Desktop (AD), Web-enabled App, Smart Client, Aplicação Web 2.0, Thin Client, Páginas e Sítios Web são tudo termos usados para descrever sucintamente vários tipos de aplicações ricas, classificando-as de acordo com a quantidade de processamento realizada no servidor e no cliente.

A figura seguinte pretende organizar os diferentes tipos de RIA [Moritz]. No eixo vertical temos a quantidade de processamento realizada pelo lado do cliente e pelo lado do servidor. O eixo horizontal apresenta a classificação das aplicações de acordo com o paradigma principal que se encontra na base do seu funcionamento.

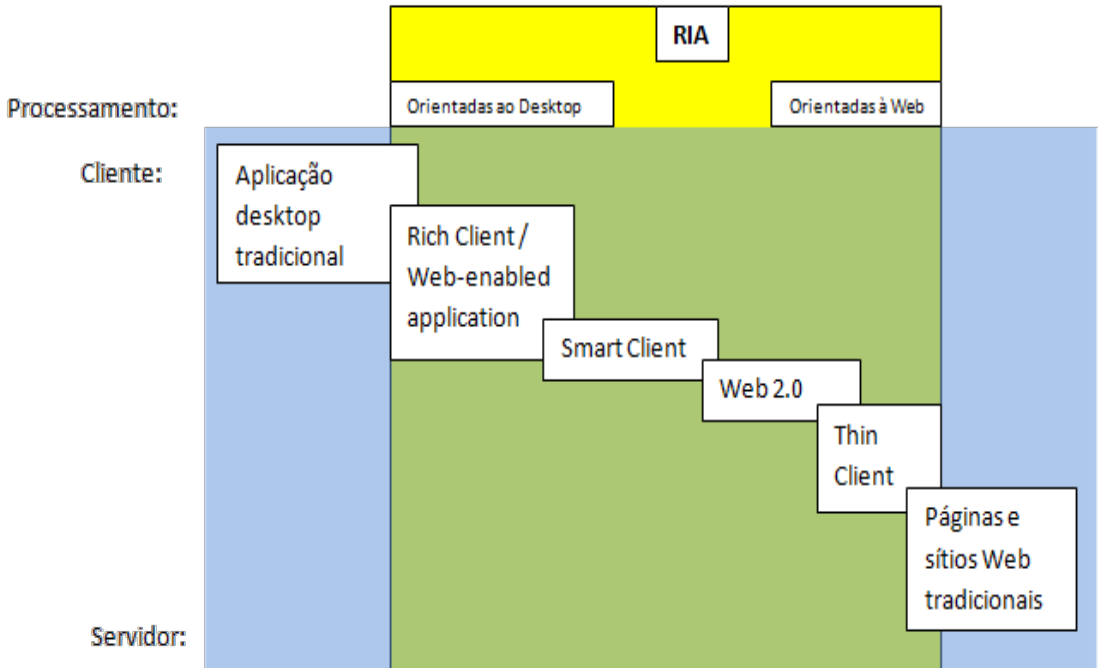

Figura 1: Classificação dos diversos tipos de RIA

Em termos práticos para classificar uma RIA, existe a necessidade de distinguir de forma clara entre uma RIA orientada à Web, que corre num Web Browser (e.g. Ajax) e uma RIA orientada ao Desktop (e.g. AIR). Com a Figura 1 torna-se mais fácil distinguir qual a categoria a que pertence uma aplicação.

Uma RIA surge então como uma mistura de dois mundos e tem como objectivo garantir a mais-valia dos dois.

A tabela seguinte clarifica melhor com exemplos de vários tipos de RIA.

Tabela 1: Classificação de aplicações RIA de acordo com a distribuição do processamento

Categoria	Descrição
Aplicação Desktop Tradicional	Consiste numa aplicação típica de Desktop que não utiliza a Web para nenhuma das suas funcionalidades principais. E.g.: Microsoft Word, 3d Studio Max
Rich Client / Web Enabled Application	Aplicação Desktop que utiliza a Web, mas funciona igualmente offline ou com conectividade reduzida. E.g.: Microsoft Outlook, Net Beans
Smart Client	Aplicação Desktop que tem como base de algumas funcionalidades principais a conectividade à Web. E.g.: iTunes, Microsoft Windows Live Messenger
Aplicação Web 2.0	Aplicação Web com muitos padrões de interacção presentes em Desktop. Podem recorrer unicamente a tecnologias standard (e.g. Ajax) e/ou tecnologias não standard (e.g. Flash, Flex, …). E.g.: Facebook, YouTube, Google Docs.
Thin Client	Aplicação baseada num Browser, instalado na máquina do cliente. Necessita de ligação à Web para funcionar. E.g.: Google toolbar, MBNet-Sidebar
Páginas e sítios Web Tradicionais	Páginas Web que apenas necessitam de um Web Browser para as mostrar. Não requerem qualquer instalação na máquina cliente. E.g.: www.fe.up.pt, www.amazon.com

As RIA orientadas para a Web distinguem-se por correrem num Web Browser, apresentando uma experiencia e facilidade de interacção mais rica e próxima das aplicações Desktop Por sua vez as RIA Desktop necessitam de uma instalação por parte do cliente, e são consideradas RIA por utilizarem a Web em determinadas funcionalidades. O iTunes da empresa Apple que é um exemplo de uma aplicação de Desktop que necessita de ser instalada. É o gestor dos conteúdos do Apple iPod ou Apple iPhone, conteúdo que se pode encontrar na máquina local, ou por sua vez recorrendo a um Web Browser integrado na aplicação, comprar na Web estes mesmos conteúdos. O Apple iTunes pode ser assim considerado por um lado como uma aplicação Web de compra de música, vídeo ou jogos, e por outro lado como uma aplicação de gestão do conteúdo de um periférico. Este exemplo torna clara a necessidade de uma RIA orientada ao Desktop, não só para melhorar a interacção da aplicação Web, mas para permitir o acesso a periféricos que de outra forma não seria possível numa aplicação Web tradicional baseada num Web Browser.

2.2 Evolução dos Browsers e da Web

O *Browser*, ou *Web Browser*, ou em português, navegador, é a aplicação responsável pela apresentação do conteúdo da Web. Desde o surgimento do primeiro Browser Web com o nome inicial de WorldWideWeb até a um Browser actual como o Mozilla Firefox 3.6 ou Google

Chrome muito existiu em termos evolutivos, desde maior capacidade de construir conteúdos visuais como a capacidade de construir tabelas, estilos Cascading Style Sheets (Css) [CSS], etc. e por outro lado funcionalidades que permitiram utilizar o Browser a outro nível, como a integração de um motor de Javascript no Browser e a capacidade de alterar o documento no lado do cliente.

Os Browsers Web foram de certa forma evoluindo com a Web e a Web com eles, assim sendo a análise das características destes ao longo do tempo permite analisar como evoluiu a Web e tirar algumas conclusões sobre o que virá a seguir.

Seguidamente são apresentados os principais Browsers que mais influenciaram a Web.

2.2.1 WorldWideWeb

Este é considerado por muitos o primeiro Browser Web. Foi lançado pela primeira vez em 1991 e era na altura a única forma de visualizar a Web. As suas únicas capacidades eram de mostrar o conteúdo básico. Mais tarde foi renomeado para “Nexus“ por motivos de confusão que o nome poderia causar [NEX91].

2.2.2 Erwise

Lançado em 1992 foi o primeiro Browser com a capacidade de navegar usando o rato para apontar para os links e navegar para este clicando nele. Para além da interacção do rato o Browser tinha como inovador a capacidade de carregar várias páginas ao mesmo tempo e a capacidade de encontrar palavras numa página Web. Foi desenvolvido para X-Window na universidade de Helsinkia por alunos finalistas, e apesar das suas características inovadoras nunca chegou a ser comercializado [ERW92].

2.2.3 MidasWWW

Mais um Browser desenvolvido para o sistema X-Window, lançado em Novembro de 1992. Uma das suas inovações era o facto de que os links mudavam de cor quando se clicavam neles, algo que ainda hoje faz parte de qualquer Browser

2.2.4 Mosaic

Erwise pode ser considerado o primeiro Browser gráfico com navegação com clique pelo rato, mas foi Mosaic [MOS93] que ganhou o seu lugar na história como o primeiro navegador gráfico largamente conhecido, tendo influencia no avanço da Web.

Mais uma vez este era um Browser desenvolvido para o sistema X-Windows para Unix e foi desenvolvido no NCSA (*National Center for Supercomputer Applications*) [NCSA10] com uma primeira versão lançada em 1993. A primeira versão ainda não implementava a especificação HTML2, mas a segunda versão do Mosaic lançado em Outubro de 1995 foi o primeiro grande Browser a suportar formulários HTML, tabelas de HTML3 e muitos estilos de texto da especificação HTML3. Como desvantagem deste Browser comparativamente com os outros era a sua menos conseguida performance.

2.2.5 Arena

Com uma primeira edição lançada em 1993, este Browser e como ponto marcante na história o facto de ter sido um Browser desenvolvido para demonstrar as ideias por trás da especificação HTML+. Foi então o primeiro Browser a suportar imagens de fundo, tabelas, controlo do texto à volta de imagens e expressões matemáticas definidas dentro do documento HTML. Foi um Browser usado durante algum tempo pelo W3C (World Wide Web Consortio) para como standard para efectuar testes à especificação HTML3 e Css até a utilização pelo W3C do Browser Amaya que ainda é hoje usado [ARN94].

2.2.6 Netscape Navigator

O único Browser até hoje que já teve mais cota de mercado que o Microsoft Internet Explorer foi o Netscape Navigator, chegando a atingir uma cota de mercado de praticamente 80% [NETSCP]. Na primeira versão lançada em 1994 oferecia suporte para HTML2 e já implementava algum do standard HTML3.

Foi neste Browser que se deu uma enorme evolução na história dos Browsers, o surgimento de uma linguagem interpretada por parte do cliente, o Javascript por volta de 1995 [NETSCP]. O facto de existir agora uma linguagem que era interpretada no lado do cliente abria uma porta enorme no que a padrões de interacção de Desktop diz respeito. Com o processamento a ser efectuado no lado do cliente é possível validar formulários sem ser necessário esperar pela resposta do servidor, melhorando em muito a interactividade com o utilizador, principalmente pela capacidade da página responder sem estar dependente da latência da rede. Esta linguagem em conjunto com a manipulação da página permitiu uma interacção com o utilizador de forma interactiva, sendo este o principio que levou ao surgimento das aplicações AJAX e da consequente Web 2.0.

Com o surgimento e seguinte domínio do Microsoft Internet Explorer o Netscape foi descontinuado.

2.2.7 Microsoft Internet Explorer

Até à pouco tempo o Browser mais popular da Web, foi o Browser da Microsoft, a partir da versão 2 que embora sendo um Browser bastante básico já tinha suporte para Javascript Na versão 3 já incluído no sistema operativo da Microsoft o Windows 95OSR2 foi um dos primeiros grandes Browsers a suportar CSS, pelo menos de forma parcial. A partir do IE3 existia o suporte para controlos ActiveX e Java Applets. A preocupação por aumentar as capacidades do Browser é desde logo presente a partir da versão 3 do Internet Explorer. Tanto Java Applets e controlos ActiveX são das primeiras tecnologias de RIA que surgiram, dando a possibilidade ao utilizador de interagir com o Browser e com a aplicação Web de uma forma muito mais rica que numa pagina Web tradicional da altura com tecnologia standard.

O ActiveX difere de outras tecnologias de RIA como o Java Applets da altura ou o Flash ou Silverlight actuais pelo facto de que um componente ActiveX ter os mesmo privilégios que qualquer outra aplicação a correr no sistema operativo Windows, como o acesso incondicional ao sistema de ficheiros por exemplo. Esta é uma das razões porque o Internet Explorer era considerado pouco seguro.

A versão 5 do IE oferecia melhorias quanto ao suporte para CSS e mais uma vez vinha embutido no Sistema Operativo da Microsoft, neste caso o Microsoft Windows 98 SE. A versão 6 com melhorias de privacidade e segurança lançada em 2001 com uma superioridade de mercado e já sem a ameaça do concorrente Netscape, pouco evoluiu durante alguns anos.

Com o Firefox em crescimento, um Browser opensource com características inovadoras, a Microsoft responde em 2006 com o IE7. Uma das evoluções desde a versão relativamente à anterior foi o suporte para navegação por *tabs* já presente em alguns Browsers anteriormente. A versão actual, IE8, é bastante semelhante em termos visuais ao IE7 mas com melhoramentos em termos de conformidade com os standards. Outra grande preocupação, existente em todos os Browsers actuais é a performance a executar Javascript que foi bastante melhorada na versão 8.

2.2.8 Opera

Lançado pela primeira vez em 1996 [OPERA], foi um Browser criado de raiz e independente de qualquer projecto ou Browser anterior. Desde do início até à actualidade a imagem de marca desde Browser passa pelo cumprimento dos standards e pela sua performance. A versão de 1997 (Opera 3) foi a primeira versão do opera a surgir com suporte para Javascript

A versão 4 do Opera lançado em Junho de 2000 trazia como grande evolução a navegação por *tabs* sendo o primeiro dos grandes Browsers a suporta-la. Actualmente a navegação por *tabs* encontra-se presente em todos os grandes Browsers Para além disso a versão 4 suporta SSL2, SSL3, CSS1, CSS2, XML e HTML4.

Uma das maiores influências do Opera quanto ao que evolução dos Browsers diz respeito foi possivelmente a navegação por “gestos” do rato, que ainda não suportados por todos os grandes Browsers nativamente é possível o suporte usando extensões. A versão 9 que existe

desde 2006, foi na altura um dos primeiros Browsers a passar o teste Acid2 [ACID2] de standards.

Actualmente a versão 10 autoproclama-se o Browser mais rápido do mundo, existindo testes efectuados por entidades externas que o confirmam, como o Peacekeeper da Futurmak [PEAKEE]. Apesar de o Opera actualmente ser o primeiro, o título de Browser mais rápido tem variado ultimamente entre o Opera e o Google Chrome.

2.2.9 Amaya

Criado pelo W3C em 1996 este é mais um editor Web que um Browser Web. O intuito principal é o de servir como ferramenta de testes de standards, assim como para experimentar novas tecnologias.

Ainda é usado hoje em dia, e o seu desenvolvimento continua a ser mantido[AMYA].

2.2.10 Safari

Browser da Apple surgiu pela primeira vez em 2003[SAFARI], e a partir de 2007 foi criada uma versão para Windows. Tendo por base o motor WebKit o Safari é conhecido pela sua boa performance e pelo boa implementação dos standards.

2.2.11 Firefox

Um Browser opensource, baseado no projecto Mozilla, e actualmente o Browser com mais utilizadores no mundo, lançado em 2004, ficou logo considerado como sendo um Browser mais seguro que o Microsoft Internet Explorer em grande parte por não suportar VBScript ou controlos ActiveX.

A base do sucesso do Firefox é a sua capacidade de suportar extensões, como por exemplo o Firebug[FBUG]. O Firebug[FBUG] é uma ferramenta de desenvolvimento considerada a mais poderosa e popular ferramenta de apoio ao desenvolvimento Web[FBUG]. Como este exemplo existem muitos mais extensões que por si só fazem do Firefox uma escolha imediata como Browser principal.

Actualmente na ultima versão é um dos poucos Browsers que já implementam algumas das funcionalidades do *draft* HTML5.

2.2.12 Chrome

O Browser mais recente do mercado dentro dos grande Browsers, lançado em finais de 2008, o Google Chrome[GCHRM] tem por base o motor WebKit, o mesmo que o Safari. É considerado um Browser muito seguro e robusto e com uma interface simples e agradável. Na

versão 3 começou a existir suporte para conteúdo HTML5 assim como uns declarados 25% de melhoramento em termos de performance[WCHRM]. A versão 4 lançada a 25 de janeiro de 2010 tem como grande melhoramento a possibilidade de suportar extensões tal como o Firefox . Actualmente na versão 5, a sua grande preocupação tem sido no suporte para novos conteúdos HTML5 assim como a performance.

2.2.13 Versões da Web

Actualmente recorre-se ao termo Web 2.0 para definir uma Web com mais e melhor interactividade e experiência de utilização mais rica. O termo Web 2.0 surgiu pela primeira vez por Tim O'Reilly [OREILLY], que a definiu como uma revolução em que a Web passaria a ser uma plataforma de aplicações. O termo Web 1.0 surgiu ao mesmo tempo que o tempo que a Web 2.0, para nomear a Web anterior a esta.

A Web 1.0 apresenta assim as seguintes características como exemplo [WEB1]:

- Páginas de conteúdo estático e falta de interactividade.
- Utilização de tags proprietárias introduzidas com a guerra dos Browsers
- Animações implementadas com imagens, tipicamente GIF's[1] animados.

As aplicações Web 2.0 apresentam assim as seguintes características diferenciadoras da Web anterior a esta [WEB2]:

- Aplicações disponibilizadas como um serviço e não como um pacote de software. O facto de as aplicações existirem na Web diminui o esforço em termos de crescimento e escalabilidade.
- Bases de dados de informação e conhecimento gerada directamente pelos utilizadores. (e.g Facebook, Wikipedia, etc.)
- Contribuição por parte dos utilizadores na construção e desenvolvimento de aplicações (e.g. Sourcefourge.net)
- A aplicação é controlada pela própria comunidade que a utilizam. (e.g Wikipedia)
- Aplicações disponíveis em muitas plataformas (e.g. Web, Desktop, telemóveis, PDA's, iPad)
- Interfaces interactivas (e.g Gmail, Google Maps)

A plataforma tecnológica que está na base da grande interactividade e da melhor experiencia de utilização possível na Web 2.0 é sem duvida o AJAX (*Asynchronous JavaScript and XML*). Não se tratando de uma tecnologia mas sim de um conjunto de tecnologias associadas a um paradigma o AJAX baseia-se em mecanismos já existentes nos Browsers desde cerca de 2001, mas foi apenas em 2004 que começaram a aparecer as primeiras aplicações

1 Graphics Interchange Format

usando este paradigma. O termo AJAX foi introduzido por Jesse James Garret em 2005 no seu artigo "**Ajax: A New Approach to Web Applications**" [JESJAM].

Fundamentalmente uma aplicação AJAX implementa os seguintes conceitos [JESJAM]:

- Interfaces com utilizador desenvolvidas através da utilização de XHTML e CSS;
- Interacção
- Transferência e manipulação de informação no formato XML. Outra forma é usando o formato Json[2] de dados;
- Processamento assíncrono com base em XMLHttpRequest;
- Programação em JavaScript do lado do Browser para combinar as referidas tecnologias.

Todo o conjunto de tecnologias em que se baseia o AJAX encontram-se implementadas pelos Browsers, não sendo necessária a instalação de software do lado do cliente, mas por outro lado a aplicação apresenta algumas diferenças de Browser para Browser, pois depende sempre de como o Browser implementa as referidas tecnologias.

Como forma de resolver o problema das diferenças de compatibilidade entre Browsers e para adicionar funcionalidades impossíveis com AJAX, como a leitura de multimédia, recorre-se normalmente ao uso de plugins para o Browser Assim sendo podemos falar de uma Web 2.0 + Plugins [PLUGIN].

Os plugins são tecnologias de RIA não standard que têm como objectivo possibilitar o seguinte:

- Transmissão de vídeo e áudio (e.g. YouTube e Last.fm);
- Animações sofisticadas;
- Interactividade avançada (e.g. jogos online);
- Capacidade de armazenamento de dados da parte do cliente;
- Capacidade de processamento superior ao Javascript;
- Independência do Browser em que são executados;

Alguns dos tópicos anteriores são actualmente impossíveis em HTML4 como a transmissão de vídeo e áudio, mas outros são possíveis de serem desenvolvidos unicamente em AJAX como as animações sofisticadas e os jogos online, apesar do seu desenvolvimento ser bastante menos facilitado. Como exemplo de tecnologias de RIA não standard temos o Adobe Flash, Silverlight ou o mais recente JavaFX. No ponto 2.8.2 encontram-se abordadas com mais detalhe os principais plug-ins para o Browser

Apesar de muito disseminados como o Flash (99% [FLSHSTAT]) estes plug-ins não conseguiram derrotar a utilização de HTML, CSS, etc., como fonte de informação e forma de desenhar uma aplicação Web. Os plugins normalmente só são usados nas seguintes situações:

2 http://www.json.org/

- Impossibilidade de implementação por parte de HTML e AJAX (como por exemplo conteúdos multimédia como áudio ou vídeo)
- Necessidade de fechar o código a executar ou o conteúdo a mostrar no lado do cliente normalmente por questões comerciais ou de segurança.

O uso de plugins para além das situações anteriormente enunciadas é praticamente inexistente, pois o uso destas tecnologias proprietárias e fechadas barra o seu conteúdo a motores de busca. A pesquisa de um site normalmente passa pelos motores de busca que desta forma não conseguiriam ler o texto da aplicação Web tirando visibilidade ao conteúdo dessa aplicação.

Recentemente com a evolução destes plug-ins, surgiram por motivos tecnológicos de limitação do Browser, e consequentemente pela necessidade também, a possibilidade de executar aplicações fora do Browser como se de uma aplicação Desktop se tratasse. Este tema irá ser abordado em mais detalhe no capítulo adiante.

2.3 Capacidades de aplicações Desktop impossíveis na Web

O uso de uma tecnologia Web Based (assente no Browser Web) ou Desktop Based (aplicação de Desktop) depende principalmente dos requisitos de interacção necessários à RIA. Apesar de cada vez mais as Frameworks Web Based para RIA serem mais completas e repletas de funcionalidades Desktop existe ainda um conjunto de interacções que se encontram limitadas pelo facto da aplicação se encontrar a correr num Web Browser.

O padrão de interacção Drag-and-Drop comum em aplicações Desktop apesar de já ser possível de aplicar numa RIA Web Based, actualmente, apenas se encontra disponível nos Browsers Google Chorme e Firefox. Outra restrição existente é a interacção com outras aplicações. O acesso a mecanismos de comunicação entre aplicações; seja memória partilhada, fifos ou pipes, etc. não é possível a uma aplicação Web dentro de um Web Browser por motivos de limitação deste. Da mesma forma se a comunicação com o sistema operativo for necessária para controlar periféricos específicos ou simplesmente por motivos de performance tal é igualmente impossível dentro de um Web Browser.

A escolha entre uma RIA Web ou uma RIA Desktop depende sempre do que se pretende obter em termos de interacção, necessidades de comunicação a mais baixo nível, performance necessária e capacidade ou não de interagir com outras aplicações e a que nível.

2.4 Surgimento das Rich Internet Applications

A Web lançada em Dezembro de 1990 [WEB90] tinha inicialmente o propósito de servir de plataforma para publicação de conhecimento. A Web utiliza como base o protocolo HTTP (Hypertext Transfer Protocol) que foi baseado no protocolo FTP (File Transfer Protocol). Este protocolo tem como paradigma o pedido-resposta entre um cliente e um servidor. O protocolo

HTTP para a transferência de texto é o mais indicado, mas com o evoluir da Web este tornou-se limitador.

A evolução natural da Web até aos dias de hoje foi a de evoluir no sentido de proporcionar interacções e características de aplicações Desktop. Uma das principais funcionalidades que apareceu nos browsers que permitiu uma grande evolução ao nível de interacção com a Web foi o surgimento do Javascript no Browser Netscape por volta de 1996.

Javascript é uma linguagem de programação que é interpretada por parte do cliente, que desta forma permitiu uma revolução na interacção possível e na experiencia de utilização de aplicações Web. O facto de ser executada por parte do cliente permitiu que se pudessem executar simples acções, ricas em termos de usabilidade, que não eram possíveis antes. Estamos a falar de modificação e validação de formulários no cliente, ajuda interactiva no preenchimento destes, efeitos de hover (modificação da cor, tamanho da letra, etc. quando é colocado o rato por cima) sobre botões sem ser necessário obter novamente a totalidade da página do servidor e unicamente aquando da execução de um click num hiperlink. O grande incremento proporcionado pelo Javascript deve-se ao facto de este poder aceder e alterar a estrutura do documento Web (Document Object Model – DOM), do lado do cliente, permitindo assim aplicações mais interactivas. O Javascript passou a ser um standard da Web e hoje é suportado por praticamente todos os browsers Web.

O Javascript foi então tão bem recebido, pelo facto de aumentar as funcionalidades de um Web Browser, sendo o início para uma Web como plataforma de aplicações e não apenas uma Web estática e nada interactiva.

No caso da Web pode-se dizer que esta evoluía mais rapidamente que os standards, pois com a evolução da Web na tentativa de ganharem mercado aos rivais os browsers passaram a integrar novas tecnologias, diferentes entre eles, cada uma com diferentes capacidades. O desenvolvedor Web tinha assim que escolher entre compatibilidade com todos os browsers ou então uma aplicação mais rica mas que apenas garantia a compatibilidade com um Browser

Actualmente ainda existem diferenças significativas entre tecnologias suportadas pelos browsers, criando problemas aos desenvolvedores Web na criação de aplicações que sejam universalmente compatíveis. Uma das formas de colmatar as diferenças entre os browsers é a utilização de tecnologias não standard para implementar certas funcionalidades. Traz a desvantagem de que é necessário por vezes a instalação de software por parte do cliente, mas cada vez mais essas tecnologias não standard se encontram disseminadas. Como exemplo temos o Adobe Flash Player que se encontra instalada em cerca de 99% dos computadores. Com estas tecnologias para além de trazerem mais capacidade de interacção eliminam os problemas de compatibilidade entre browsers.

O termo RIA foi utilizado pela primeira vez por Jeremy Allaire, no seu trabalho de 2002 "*Macromedia Flash MX - A next-generation rich client*" [ALL02]. O objecto do seu trabalho era promover o Flash, mas ao mostrar os pontos fortes e vantagens do Flash mostrou também por um lado as lacunas da Web. O quadro seguinte [ALL02] explora em concreto as lacunas da Web segundo esse autor.

Tabela 2: Principais lacunas da Web segundo Jeremy Allaire [ALL02]

Lacunas da Web tradicional	Mais-valias das RIA
Impossibilidade de processamento de informação (lado do cliente), dependência do paradigma request-response para actualização e apresentação das páginas e impossibilidade de armazenamento de dados (lado do cliente)	Possibilitar um ambiente eficiente de execução de código, tratamento e disponibilização de conteúdo e realização de comunicações
Necessidade de diferentes tecnologias, sendo que a interacção e troca de informação entre elas não é standard (texto, vídeo e áudio, animação, comunicações em tempo real)	Contexto integrado para conteúdos, comunicações e interfaces com aplicação
Falta de Object Oriented Design (OOD) para o desenvolvimento de aplicações de grande dimensão.	Possibilitar o desenvolvimento de interfaces com o utilizador, de acordo com modelos de objectos que podem ser estendidos pelos programadores.
Não existem (ou não existiam na altura) componentes e bibliotecas de componentes para facilitar e acelerar o desenvolvimento de aplicações mais complexas.	Possibilitar a utilização de componentes previamente desenvolvidos e reutilização de código para acelerar o processo de desenvolvimento de aplicações.
Não apresenta uma separação clara entre: - Lógica de apresentação - Interface com o utilizador - Lógica da aplicação	Possibilitar a utilização de serviços e informação na Web, através de serviços disponibilizados por servidores aplicacionais
Só funcionam desde que exista uma ligação à Web. Quando essa ligação é de má qualidade, a experiencia de utilização diminui significativamente.	Abranger utilizadores com e sem ligação à Web.

O quadro anterior tenta mostrar as principais lacunas da Web e como as RIA surgem assim no sentido de as colmatar.

O termo RIA nasce assim da evolução da Web e das novas necessidades em transformar este mundo numa plataforma de aplicações mais completas e com maior capacidade de interacção.

2.5 Migração de Desktop para a Web

Com o fenómeno de expansão da Web, e com o evoluir das tecnologias e dos browsers começou-se a ter uma melhor capacidade de interacção com os sítios Web. Um dos grandes impulsionadores da Web como plataforma de aplicações foi a empresa Google com o surgimento das aplicações Gmail e Google Maps [SWARTA], usando apenas tecnologias já disponíveis em grande parte dos Browsers. A Google inovou com o surgimento de aplicações disponíveis na Web e gratuitas que pretendiam competir com grandes aplicações Desktop. Um dos exemplos é o Google Docs. Esta aplicação Web com uma interface em tudo semelhante a uma aplicação Desktop tem como objectivo competir com aplicações como OpenOffice ou

Microsoft Office. Como grande vantagem desta aplicação temos o facto de não ser necessária qualquer instalação na máquina cliente, pois usa como tecnologia de RIA o Ajax, disponível em praticamente todos os Browsers.

Na Figura 2 pode-se ver o quanto semelhante a uma aplicação Desktop é o Google Docs. Este é um bom exemplo do uso da Web como plataforma de aplicações.

Figura 2: Google Docs

Para além das funcionalidades da aplicação, semelhantes às presentes em Desktop, existem ainda funcionalidades que surgem com a Web. Como exemplo temos a partilha de documentos e a edição simultânea por vários utilizadores. A integração com a aplicação Gmail da Google permite a visualização de documentos em anexo nos emails directamente no Browser, sem ser necessário o download do documento em questão, nem a existência de software para além do Browser para o abrir.

Outros exemplos existem de aplicações que pretendem concorrer com versões típicas de Desktop como o Zoho Writter[3], gOffice[4], iNetWord[5] ou o ThinkFree[6] [FREEONL].

O facto de estas aplicações se encontrarem disponíveis na Web traz algumas vantagens para os utilizadores, dai esta procura em fornecer aplicações em ambiente Web. Seguidamente são apresentadas as vantagens e motivos do uso de aplicações na plataforma Web [LIC09].

- Os utilizadores actualmente usam diversos computadores e dispositivos no seu dia-a-dia (em casa, no emprego, no telemóvel, no estabelecimento de ensino, etc.) e o facto de a aplicação estar disponível em todos eles, assim como a sua informação, é uma mais-valia para este tipo de utilizador.

3 http://writer.zoho.com
4 http://www.goffice.com
5 http://www.inetword.com
6 http://www.thinkfree.com

- Utilizadores sem computador pessoal necessitam de ter acesso às suas informações em computadores públicos ou emprestados, assim como não têm forma de instalar aplicações. A alternativa viável para este grupo de utilizadores será a Web.
- O facto de se ter de instalar uma aplicação Desktop pode intimidar utilizadores. Uma aplicação Web por sua vez não necessita de qualquer instalação.
- Actualmente a capacidade dos servidores assim como a evolução das infra-estruturas físicas de comunicação disponíveis permitem um acesso a serviços de forma eficaz.
- O Ajax assim como outras tecnologias permitem obter níveis cada vez mais elevados de performance e interactividade das aplicações. A performance das aplicações Ajax têm vindo a evoluir cada vez mais em grande parte pelo suporte e preocupação cada vez maior dos Browsers na eficiência do código a correr do lado do cliente. A interactividade e mais em concreto a facilidade de implementação desta interactividade AJAX aparece também em grande parte pelo aparecimento de bibliotecas de Javascript cada vez mais poderosas e com grandes funcionalidades, como por exemplo jQuery[7] ou Prototype[8].
- A opção de uma empresa de lançar um produto de software na Web traz vantagens económicas óbvias, pois elimina a necessidade de produto físico. Outra vantagem é o facto de não existirem preocupações com a compatibilidade com plataformas (Windows, Linux, Mac) existindo assim um único código fonte. Apesar de apenas existir um código fonte existem ainda questões de compatibilidade entre browsers, mesmo em tecnologias standard. É eliminada a questão de compatibilidade entre versões de software, visto a actualização ser ao nível do servidor Web, estando o cliente sempre a executar a versão mais recente.
- Cada vez mais as tecnologias e ferramentas permitem um desenvolvimento facilitado de aplicações Web. O uso de IDE (Integrated Development Environment) mais poderosos orientados à Web, e inclusivamente capacidade de debugging (Funcionalidade de detectar e corrigir erros de software) presente em Browsers. Bibliotecas e Frameworks de software que simplificam o desenvolvimento e garantem a compatibilidade entre vários browsers, como por exemplo jQuery[7], Prototype[8], etc. [AJAXLIB] são um avanço no desenvolvimento para a Web.
- As aplicações Web permitem que exista um controlo muito maior sobre os utilizadores que as usam, podendo obter informações sobre acessos, acções efectuadas, etc. A cópia ou uso ilegal de software torna-se muito mais complicada ou mesmo impraticável. A cópia de software torna-se inviável pois a máquina do cliente não tem acesso ao código do servidor.
- O facto de ser Web e usar tecnologias presentes em vários tipos de plataformas torna a aplicação acessível em diversos dispositivos (PDA's, telemóveis, computadores) e em diversos sistemas operativos.

7 http://jquery.com/

8 http://www.prototypejs.org/

- As aplicações Web têm a vantagem de facilitar a colaboração e a partilha entre utilizadores separados geograficamente. Um exemplo é o Google Docs que permite a edição em simultâneo por vários utilizadores de um mesmo documento.

Com os argumentos anteriores pode-se assumir que a migração para a Web traz mais-valia para a indústria de Software. Apesar de todas as vantagens anunciadas a escolha entre Web e Desktop dependem sempre dos requisitos da aplicação em concreto e de uma análise das capacidades actuais da Web.

Um exemplo interessante é o eyeos[9]. É um projecto Open Source e aumenta a ideia de aplicações na Web para Desktops na Web. O Project é uma espécie de sistema operativo que corre num servidor e é acessível através de um Browser Web. Tal como num sistema operativo comum é possível executar e instalar aplicações.

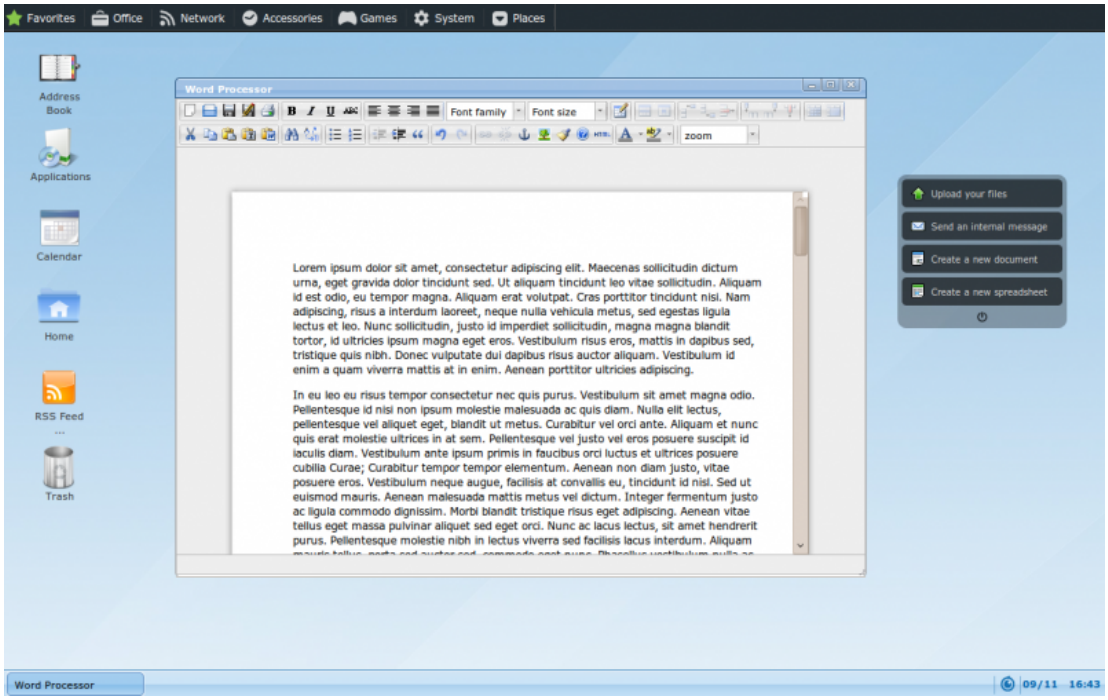

Figura 3: Ecrã do sistema eyeos

Na Figura 3 mostra-se um ecrã do programa eyeDocs a correr em ambiente eyeos que por usa vez corre dentro de um Web Browser. Este conceito tem o nome de WebOS (Web operating system) e mostra até que ponto é possível a fusão entre o Desktop e a Web. Para além do eyeos existem mais exemplos semelhantes como o G.ho.st (Global hosted operating system)[10]. Ambos estes sistemas são compatíveis com várias plataformas e periféricos estando acessíveis tanto em computadores pessoais como em dispositivos móveis. Este tipo de sistemas para além das vantagens enunciadas trazidas pelas aplicações Web permite-nos ter não uma aplicação mas todo um Desktop em múltiplos lugares e dispositivos.

9 http://eyeos.org
10 http://ghost.cc/

2.6 Migração da Web para o Desktop

A evolução da Web e sucessivamente o melhoramento das capacidades dos Browsers e tecnologias permitiu enriquecer as aplicações Web juntando-lhes cada vez mais características de Desktop. Com a proliferação do Ajax e de outras tecnologias como Flash ou Silverlight permitiu ainda mais aproximar a Web do Desktop.

Apesar de toda a evolução e capacidades das aplicações Web estas encontram-se ainda com algumas lacunas relativamente as capacidades das aplicações Desktop. É exactamente por este motivo que aplicações que inicialmente eram Web e que o seu surgimento tinha unicamente sentido na Web, evoluíram de forma a complementar as suas funcionalidades com um cliente Desktop. Como exemplo temos o eBay Desktop. O eBay é um sítio na Web de leilões, mas para adicionar mais funcionalidades e melhor usabilidade decidiu criar uma versão Desktop como evolução da sua aplicação.

O surgimento de versões Desktop para aplicações Web é fruto de necessidades não atingíveis actualmente dentro de um Browser Seguidamente são apresentadas as principais lacunas actuais das aplicações Web baseadas no Browser.

- Incapacidade de interacção com o sistema operativo de forma mais natural, tal como o efectuar Drag-and-Drop entre o sistema de ficheiros e a aplicação. Esta interacção já pode ser realizada nos Browsers Firefox 3.6 e Google Chrome, utilizando como tecnologia o HTML5, mas ainda se encontra limitado a apenas dois browsers.
- A aplicação Web a correr dentro de um Browser encontra-se sempre limitada e fechada no que ao sistema operativo diz respeito. O acesso a este não é possível, limitando funcionalidades como acesso a periféricos ou comunicação com outras aplicações (memoria partilhada, fifos, sockets). Actualmente os únicos periféricos que é possível obter dentro do Browser são a câmara web e o microfone, utilizando como tecnologia Microsoft Silverlight 4 ou Adobe Flash.
- O código que executa no ambiente do Browser como o Javascript ou ActionScript do Flash não têm a mesma eficiência que o de uma aplicação Desktop, e apenas em poucas tecnologias é possível programar em multithread, o que pode ser bastante limitador do ponto de vista de desenvolvimento de uma aplicação.
- Incapacidade de utilizar Frameworks ou bibliotecas de software disponível para aplicações Desktop, tais como XNA da Microsoft ou o acesso a API's gráficas como OpenGL.
- O acesso a hardware gráfico apesar de possível em ambiente Browser ainda se encontra bastante limitado relativamente ao mundo Desktop.
- A capacidade de aceder offline à aplicação, como a necessidade de guardar informação local, é praticamente como um impedimento ao mundo do Browser. Existem ferramentas que acrescentam essas funcionalidades ao Browser como o Gears da Google (actualmente descontinuado) mas que se encontram limitadas a apenas alguns browsers perdendo a grande vantagem de multi-plataforma da Web.

Analisando os pontos anteriores nota-se que ainda existe uma necessidade de evoluir as tecnologias até se conseguir obter as capacidades de Desktop dentro de uma aplicação Web

baseada no Browser. Uma grande evolução neste sentido pretende ser o HTML5 que quando implementado cobrirá algumas das lacunas apresentadas, e como standard diminuir a necessidade de plug-in's proprietários para desenvolver a aplicação Web (Flash, Silverlight, JavaFX, etc.) .

2.7 HTML5

A grande revisão seguinte planeada ao standard do HTML é o HTML5 [HTML5]. Actualmente na versão HTML 4.0.1, a nova versão permite colmatar grande parte das lacunas existentes no HTML. Uma das alcunhas dadas ao HTML 5 é o de "Flash-Killer", pois o objectivo principal é de tornar standard uma especificação, que torne obsoleta tecnologia não standard como Flash, JavaFX ou Silverlight, que só existem actualmente para corresponder as necessidades de interacção que os utilizadores exigem da Web.

Na especificação do HTML5 serão avançadas novas API's destacando-se as seguintes [H5WIKI]:

- Existência de uma API de canvas 2D. Desta forma recorrendo a uma API de desenho 2D vai ser possível criar gráficos mais completos e de forma mais eficiente do que utilizado tabelas ou secções rectangulares (tag) para a construção de objectos gráficos. A utilização de uma API 2D permite acima de tudo a criação de aplicações interactivas a um nível muito mais facilitado para o desenvolvedor, já não necessitando em parte de usar tecnologia como Flash ou Silverlight para obter o mesmo efeito.
- Leitura de conteúdos multimédia. Com o HTML5 não será obrigatório como agora ter uma tecnologia não standard para colocar um vídeo ou música embutida numa página Web. Actualmente a aplicação Web da Google YouTube, já tem uma versão experimental em HTML5, não necessitando portanto de nenhum software para além de um Browser compatível. Actualmente na versão de HTML 4 o leitor de vídeo está implementado em Flash.
- Base de dados local para armazenamento de informação offline. Com uma base de dados local é possível guardar conteúdo de forma a este estar disponível offline do lado do cliente. Esta funcionalidade é muito interessante e actualmente só possível com tecnologia não standard. Como grande vantagem tem o facto de permitir uma experiência mais rica de interacção com um utilizador que se encontre com uma ligação apenas temporária ou inconstante à Web.
- Capacidade de *Drag and Drop*. Com o HTML5 será possível arrastar documentos do sistema de ficheiros local para a aplicação Web tal como se de uma aplicação Desktop se trata-se. Este tipo de interacção é de grande valor para os utilizadores.«

Algumas das funcionalidades estáveis do HTML5 já se encontram implementadas por alguns browsers como o caso do Mozilla Firefox e do Google Chrome. O HTML5 ainda se

encontra numa versão draft e de acordo com Ian Hickson a norma atingirá o estatuto de recomendada a candidata em 2012.

Um exemplo de problemas no HTML5 é o facto de quais os formatos de vídeo que devem ser suportados na tag <video> do HTML5. Na especificação actual cada Browser suporta os formatos de vídeo que entender. Isto pode-se tornar um problema para a adopção do standard, no sentido que alguns formatos de vídeo, por questões de propriedade, podem apenas ser permitida a sua descodificação em alguns browsers, logo voltando ao problema de diferenças entre browsers.

2.8 Tecnologias disponíveis

O conjunto de tecnologias disponíveis actualmente é bastante vasto ao nível de Frameworks apesar de maior parte delas não serem standard.

Os tópicos seguintes tentam abordar as Frameworks e outras tecnologias mais utilizadas na construção de uma RIA, assim como descrever as principais vantagens destas. As Frameworks seguintes apresentam-se divididas em dois grupos; as tecnologias Web Based (Que correm dentro de um Web Browser) e as tecnologias Desktop Based (Aplicação Desktop que é instalada na maquina cliente).

2.8.1 Ajax

A tecnologia standard principal para o desenvolvimento de uma RIA é o Ajax (asynchronous Javascript and XML). Em concreto Ajax não é uma tecnologia mas sim um grupo de tecnologias que todas juntas formam o paradigma Ajax. As tecnologias usadas não necessitam de ser obrigatoriamente Javascript ou XML (nem de ser necessariamente assíncrona). Ajax é assim um conjunto de técnicas e tecnologias usadas em conjunto no desenvolvimento de páginas Web de forma a aumentar o dinamismo destas e melhoram a interacção com o utilizador.

O padrão Ajax consiste em do lado do cliente de forma normalmente assíncrona alterar dinamicamente o conteúdo da página em questão por Javascript, e obter dados do servidor sem necessidade de actualizar toda a página. A este conceito também se dá o nome de DHTML (Dynamic HTML). A obtenção de dados por parte do servidor Web de forma assíncrona combinada com o processamento desta por parte do cliente permite uma usabilidade muito maior em aplicações Web, sustentando assim o desenvolvimento de um conjunto enorme de padrões de interacção provenientes de aplicações Desktop. A alteração do conteúdo das páginas Web é feita por Javascript acedendo ao DOM, e a obtenção de dados de forma assíncrona é utilizando a API XMLHttpRequest que existem em praticamente todos os browsers. A utilidade deste conjunto de tecnologias tornou-se evidente quando a Google criou as aplicações Gmail e Google Maps [SWARTA], que usavam largamente o paradigma Ajax, existindo a partir dessa data um grande exemplo das capacidades do Ajax. O Gmail por exemplo periodicamente obtém do servidor assincronamente informação sobre novos emails, e caso existam, actualiza a página.

No Google Maps é possível arrastar os mapas com o rato e quando vamos para novas posições do mapa em "background" e assincronamente é feito o download das partes novas do mapa e continuamente actualizadas na página sem ser necessário o carregamento completo desta.

Comparada com tecnologias não standard peca principalmente pela dificuldade acrescida de desenvolvimento e pela performance limitada de execução de código em Javascript.

Tabela 3: Sumário da tecnologia AJAX

Ajax	
Vantagens	- Totalmente standard - Capacidades elevadas de interacção
Desvantagens	- Desenvolvimento não muito facilitado - Sem capacidades multimédia - Para certos casos não tem a performance necessária

2.8.2 Web Based

Dentro das aplicações RIA baseadas na Web podemos falar de tecnologias standard, tecnologias suportadas por maior parte dos Browsers Web, ou de tecnologias não standard, ou seja, tecnologias apenas suportadas nativamente por parte ou nenhum dos principais browsers, sendo necessária a instalação de algum software na parte do cliente.

2.8.2.1 Adobe Flash

Adobe Flash ou anteriormente Macromedia Flash, surgiu em 1996, e tornou-se bastante popular por permitir a adição de animação e interactividade em páginas Web. Como se trata de uma tecnologia não standard, apesar de muito comum, necessita da instalação na máquina cliente (como plugin para o Browser) de forma a permitir a execução do seu conteúdo no Browser De acordo com os dados da própria Adobe o Flash Player encontra-se instalado em 99% de todos os computadores em mercados ditos maduros e num conjunto largo de dispositivos [FLSHSTAT], mostrando assim a grande adesão que existe a esta tecnologia.

Uma das maiores aplicações Web conhecidas que utilizam o Flash como tecnologia principal é o YouTube da Google. Este utiliza o Flash como tecnologia para o streaming de vídeos na Web, visto não haver tecnologia standard disponível para isso.

Figura 4: Flash Player no YouTube

A Figura 4 é um exemplo de uma página Web que utiliza Flash como tecnologia principal. De notar o leitor de vídeo semelhante em interface a uma aplicação de multimédia para desktop tipo Microsoft Windows Media Player ou VLC Media Player. A experiencia de utilização comparada com a de uma página Web tradicional é muito melhor.

Em aplicações Flash após o carregamento total da página é possível que esta funcione sem ligação à Web, se esta não necessitar de dados da Web constantemente como no caso do streaming de vídeo ou áudio.

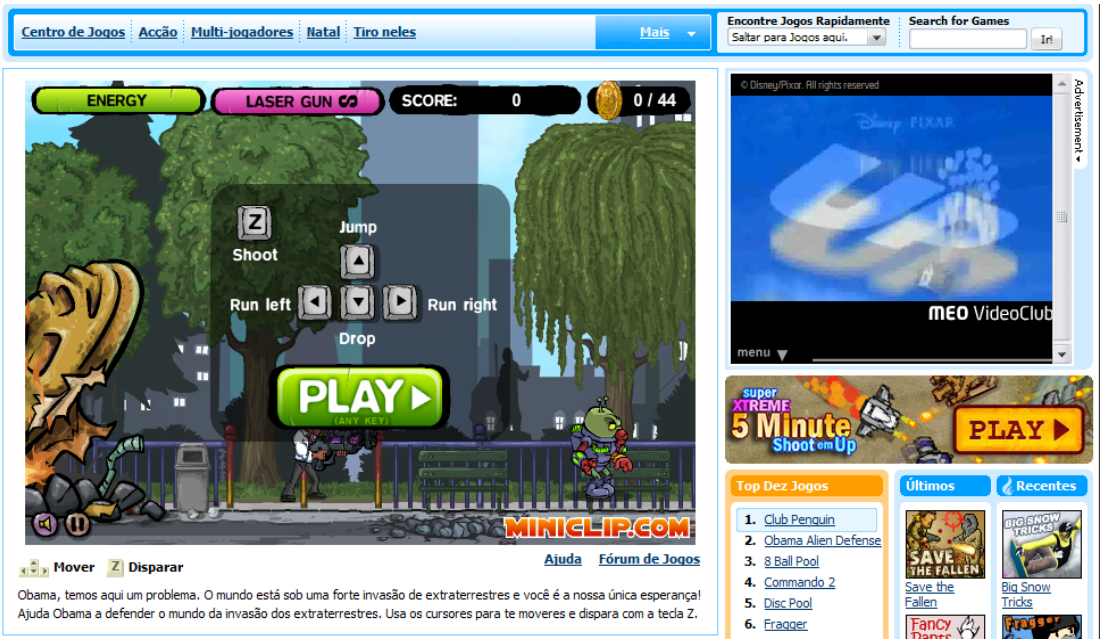

Figura 5: Exemplo de jogo desenvolvido em Flash

No exemplo da Figura 5 retirada da página Miniclip[11] a aplicação existente em Flash após ser carregada inicialmente com o resto da página, passa a executar totalmente na máquina cliente não necessitando de acesso à Web para executar o jogo. Assim, mesmo com uma ligação à Web inconstante ou lenta, consegue-se obter uma experiência rica em interacção, podendo em casos que não necessitem de ligação permanente à Web (como o exemplo de um jogo não multi-jogador) funcionar sem ligação à mesma.

A capacidade de integração com motores de Jogo é possível mas unicamente em 2D.

O quadro seguinte tem como objectivo apresentar resumidamente vantagens e desvantagens desta tecnologia.

Tabela 4: Sumário da tecnologia Flash

Flash	
Vantagens	- Grande índice de permeabilidade no mercado (cerca de 99% [FLSHSTAT]) - Grande capacidade de efectuar animações - Tecnologia madura (desde 1996) - Facilidade de construir aplicações complexas usando uma linguagem de programação OO (Object Oriented) ActionScript
Desvantagens	- ActionScript 3 tem algumas limitações em termos de linguagem OO [RIAF] - Não tem suporte para processamento multithread

11 http://www.miniclip.com

2.8.2.2 Adobe Flex

Flex da Adobe é uma tecnologia que surgiu em 2004, que tem como grande vantagem o facto de usar Flash como leitor, sendo esta a principal vantagem desta tecnologia. Flex surge como uma evolução ao Flash, não em termos de possibilidades na Web, mas sim em termos de facilidade e quantidade de componentes pré-definidos no desenho de interfaces e aplicações. Flex é assim uma evolução no que ao desenvolvimento diz respeito, permitindo fazer o mesmo mais rápido e com menos código.

Quanto ao que a permeabilidade no mercado diz respeito, esta é idêntica ao do Flash possibilitando a sua utilização, sem necessidade de instalação de qualquer aplicação na maioria dos casos.

No quadro seguinte encontram-se as principais vantagens desta tecnologia.

Tabela 5: Sumário da tecnologia Flex

Flex	
Vantagens	- Utiliza Flash Player como leitor que existe disponível em 99% dos computadores do mercado [FLSHSTAT] - Maior quantidade de componentes e possibilidades disponíveis no SDK relativamente ao Flash
Desvantagens	- Não tem suporte para processamento multithread

2.8.2.3 Microsoft Silverlight

O Silverlight da Microsoft surge como uma alternativa as tecnologias da Adobe, e do enorme peso que esta tem nos mercados de transmissão de multimédia. Como grande vantagem desta tecnologia temos o facto de esta se englobar dentro da plataforma .NET. Isto significa que utilizamos como ferramenta de desenvolvimento o Microsoft Visual Studio e que podemos programar linguagem NET C# o que torna o Silverlight extremamente poderoso para o desenvolvedor. Com o Visual Studio como ferramenta de desenvolvimento assegura-se um grande auxílio ao programador com uma ferramenta muito madura e avançada em termos de IDE (*Integrated Development Environment*) de programação.

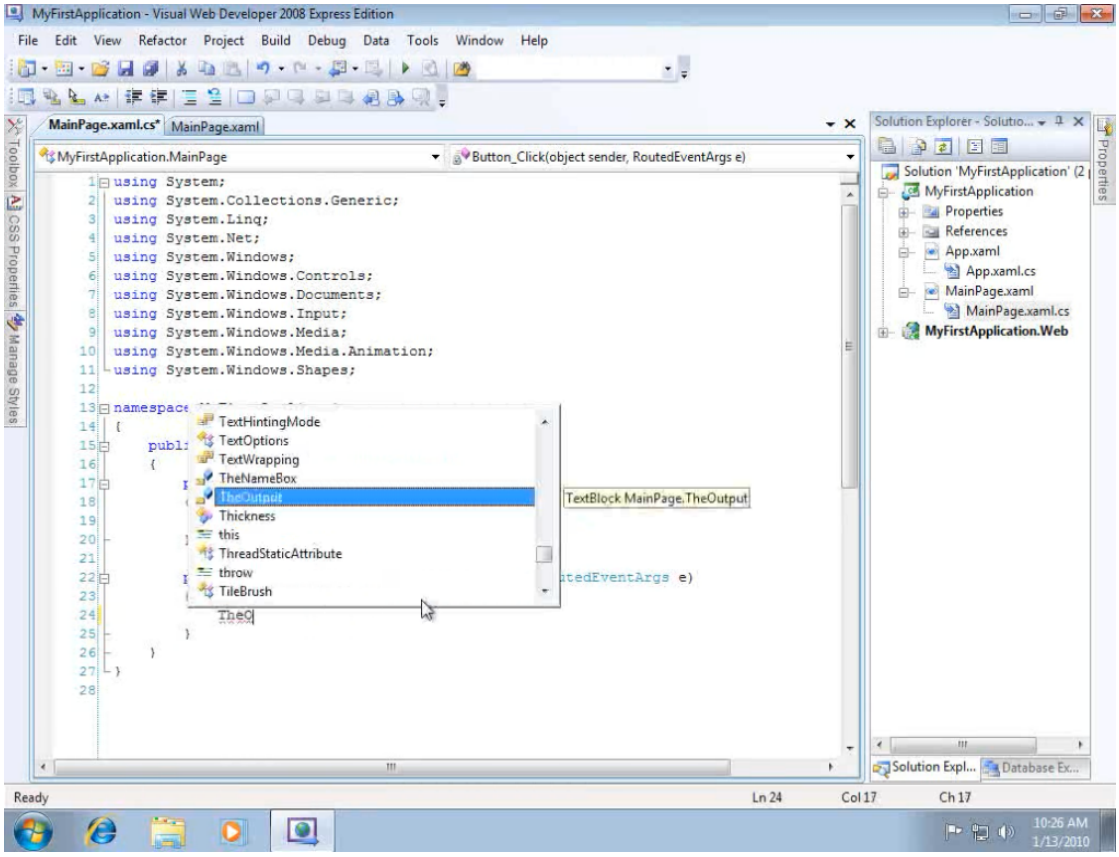

Figura 6: Ambiente de desenvolvimento de uma aplicação em Silverlight

A Figura 6 mostra um exemplo do desenvolvimento de uma aplicação em Silverlight usando a linguagem C#. Pode-se observar o acesso a funcionalidades disponíveis no desenvolvimento de aplicações Desktop em Visual Studio como o IntelliSense (sistema de ajuda por *autocomplete*) da Microsoft.

A compatibilidade com o Windows e múltiplos Browsers é garantida, e actualmente é possível executar aplicações Silverlight 3 em Linux e FreeBSD dentro do Browser apenas recorrendo ao plug-in para Firefox Moonlight da Novel. A compatibilidade com Mac é garantida pela própria Microsoft sempre para a versão mais recente.

Existe a capacidade de guardar dados locais de duas formas. A primeira recorrendo ao chamado Isolated Storage que é um sistema de ficheiros virtual acessível a uma aplicação Silverlight. Outra forma é acedendo directamente ao sistema de ficheiros local mas unicamente à pasta "Os Meus Documentos" ou equivalente se não estivermos em ambiente Windows. A utilização com base de dados relacional ou objecto relacional apenas é possível utilizando projectos externos ou alguma solução proprietária não gratuita .

A partir da versão 3 do Silverlight é possível exportar para o Desktop esta aplicação, sendo a instalação extremamente fácil tanto para o utilizador como para o desenvolvedor [SLVRLDSK]. Do ponto de vista de funcionalidades a versão Desktop a partir da versão 4, que é a versão actual, permite uma comunicação com outras aplicações utilizando COM (*Component Object Model*) [COM] que permite a interacção com programas da Microsoft tais

como o Microsoft Word, Microsoft Excel ou outro tipo de aplicações compatíveis com esta tecnologia.

O quadro seguinte pretende apresentar de forma resumida as principais vantagens e desvantagens do Silverlight.

Tabela 6: Sumário da tecnologia Silverlight

Silverlight	
Vantagens	- Programação em C# - Desenvolvimento integrado no Visual Studio - Alguma compatibilidade com sistemas Linux, FreeBSD, etc. através do Moonlight - Possibilidade de integração com outras tecnologias Microsoft como XNA, mas apensa gráficos 2D usando a biblioteca Silversprite
Desvantagens	– Comparado com as outras grandes tecnologias semelhantes à data actual é a que tem menor índice de permeabilidade do mercado [RIASTAT] – Menos madura que o Flash – A versão mais recente ainda não é suportada em Linux

2.8.2.4 Sun JavaFX

JavaFX da Sun é uma das mais recentes tecnologias a emergir no mundo das RIA. O seu ponto mais forte é a integração com a linguagem Java também da Sun e todo o mundo à volta do Java. Permite o uso de IDE bastante conhecidos e gratuitos como o Eclipse e o NetBeans. A grande capacidade de multi-plataforma do java permite que esta seja uma plataforma de desenvolvimento de RIA não só para Desktop mas igualmente para o mercado dos dispositivos móveis. O JavaFX peca principalmente pela falta de maturidade [RIAF].

Esta tecnologia permite a criação de RIAs desktop, ainda que para desktop apenas seja compatível com Windows e Mac. A versão de desktop não incorpora um Web Browser.

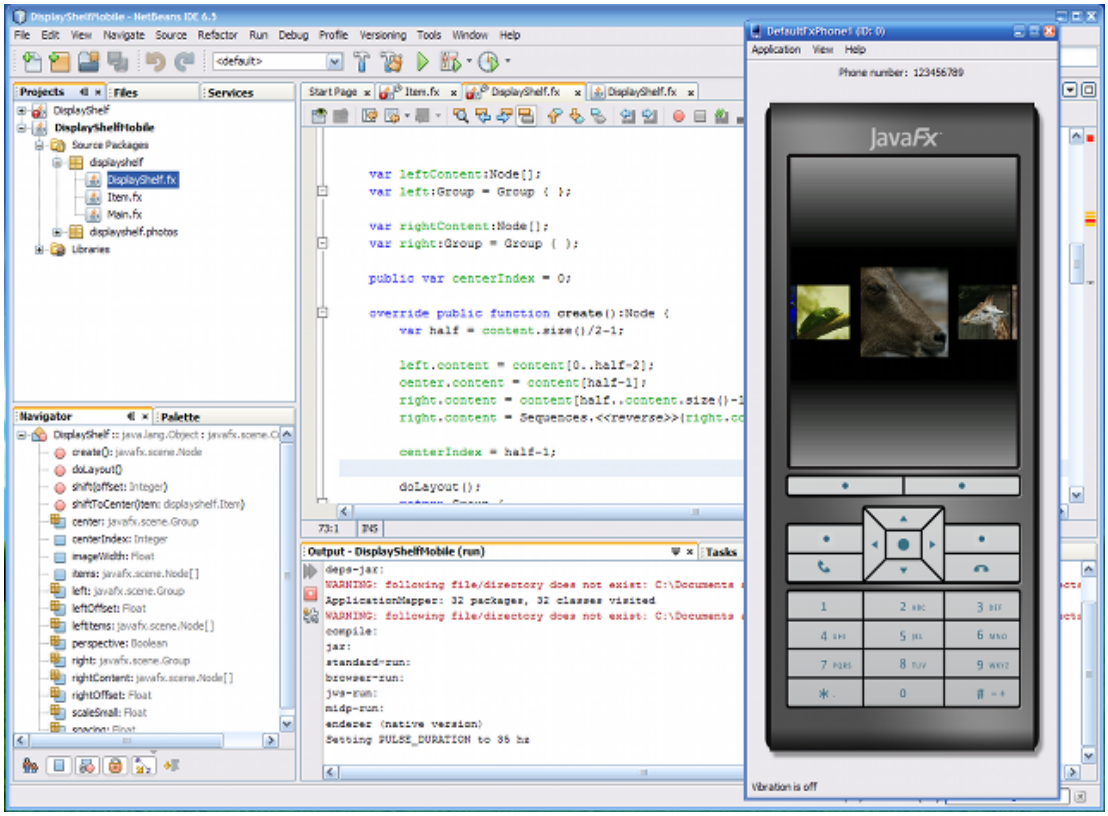

Figura 7: Ambiente de desenvolvimento de JavaFx

Na Figura 7 pode-se ver o desenvolvimento de uma aplicação em JavaFX no IDE NetBeans para um dispositivo móvel. De notar a facilidade de desenvolvimento em dispositivos móveis, recorrendo a emuladores.

Tabela 7: Sumário da tecnologia JavaFX

JavaFX	
Vantagens	- Tem por base a linguagem Java - Multiplataforma - Maior permeabilidade no mercado que o Silverlight da Microsoft [RIASTAT]
Desvantagens	– Falta de maturidade da tecnologia no que diz respeito ao desenvolvimento de RIA – Inexistência de grandes e conhecidas aplicações com esta tecnologia

2.8.2.5 Visual WebGui

A ferramenta Visual WebGui[12] distingue-se das anteriores pelo seu objectivo como ferramenta construtora de RIA. Esta ferramenta ao contrário de Flex ou Silverlight não tem por objectivo a animação e a multimédia, mas sim tentar ao máximo obter a semelhança entre a aplicação desenvolvida e uma aplicação Desktop tradicional. Uma das vantagens desta

12 http://www.visualwebgui.com

aplicação passa inicialmente pelo desenvolvimento. O desenvolvimento é efectuado em ambiente Visual Studio, mas contrariamente ao ASP.NET (tecnologia server side de desenvolvimento Web) da Microsoft a aplicação utiliza as mesmas ferramentas que são utilizadas no desenvolvimento em Visual Studio mas desta vez em aplicações Desktop. Nesta ferramenta o desenvolvedor abstrai-se completamente de questões normais na Web como desenhar a página em HTML, CSS, escrever o código em Javascript, ter em atenção sobre que informação tem que ser retornada em XML, etc., inclusivamente não existe a noção de código do cliente e de código do servidor.

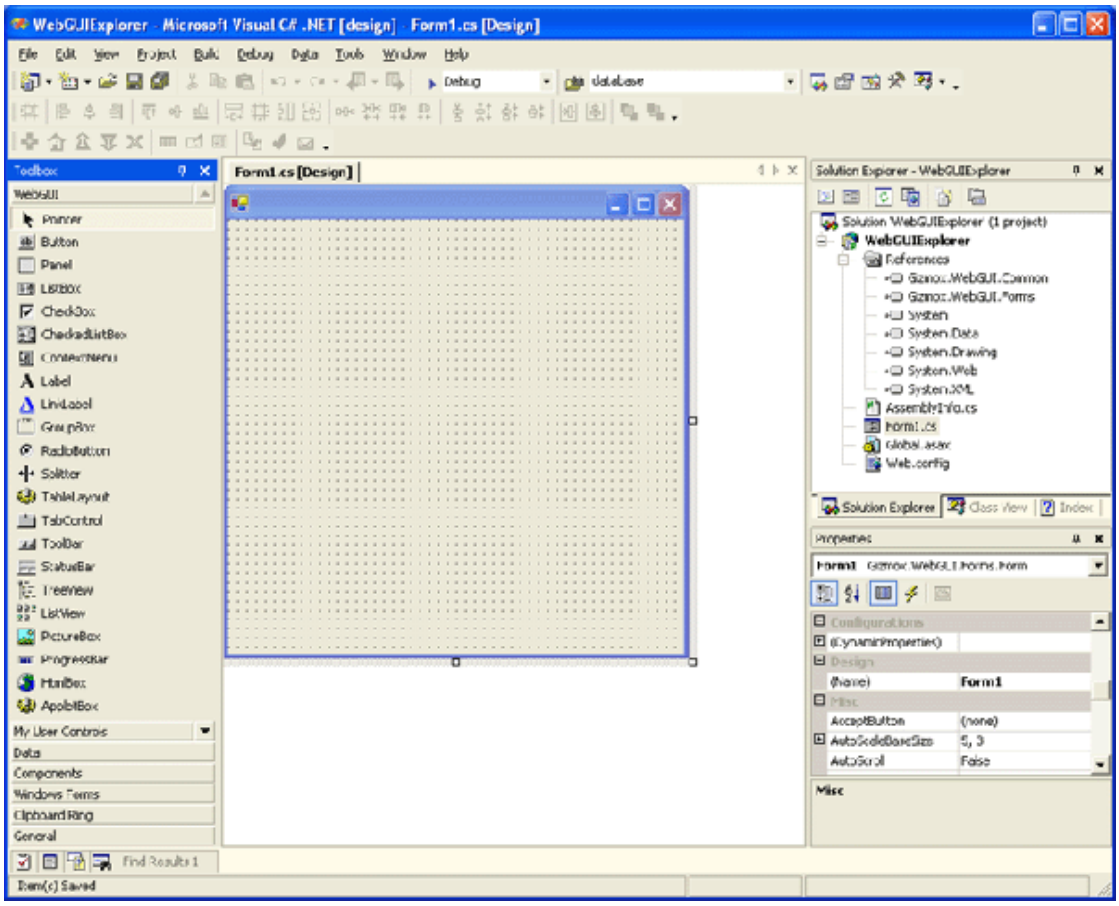

Figura 8: Ambiente de desenvolvimento da tecnologia VisualWebGui

Na Figura 8 pode-se observar o desenvolvimento da aplicação Web como se de uma aplicação Desktop se tratasse em Visual Studio. De notar o facto de que a aplicação apesar de ser Web é construída mesmo assim sobre uma janela do Windows.

Uma das grandes vantagens apontadas pela empresa é a migração fácil de uma aplicação Desktop, como por exemplo uma aplicação em Visual C++, para a Web, permitindo inclusive que seja mantido o seu código fonte ou objectos binários [VISWGID].

A ferramenta inclui dois modos de exportação para RIA. Uma forma é a exportação para DHTML (Dynamic HTML, normalmente sinónimo de Ajax), ou então para Silverlight.

A imagem seguinte mostra um exemplo de aplicação construída em Visual WebGui.

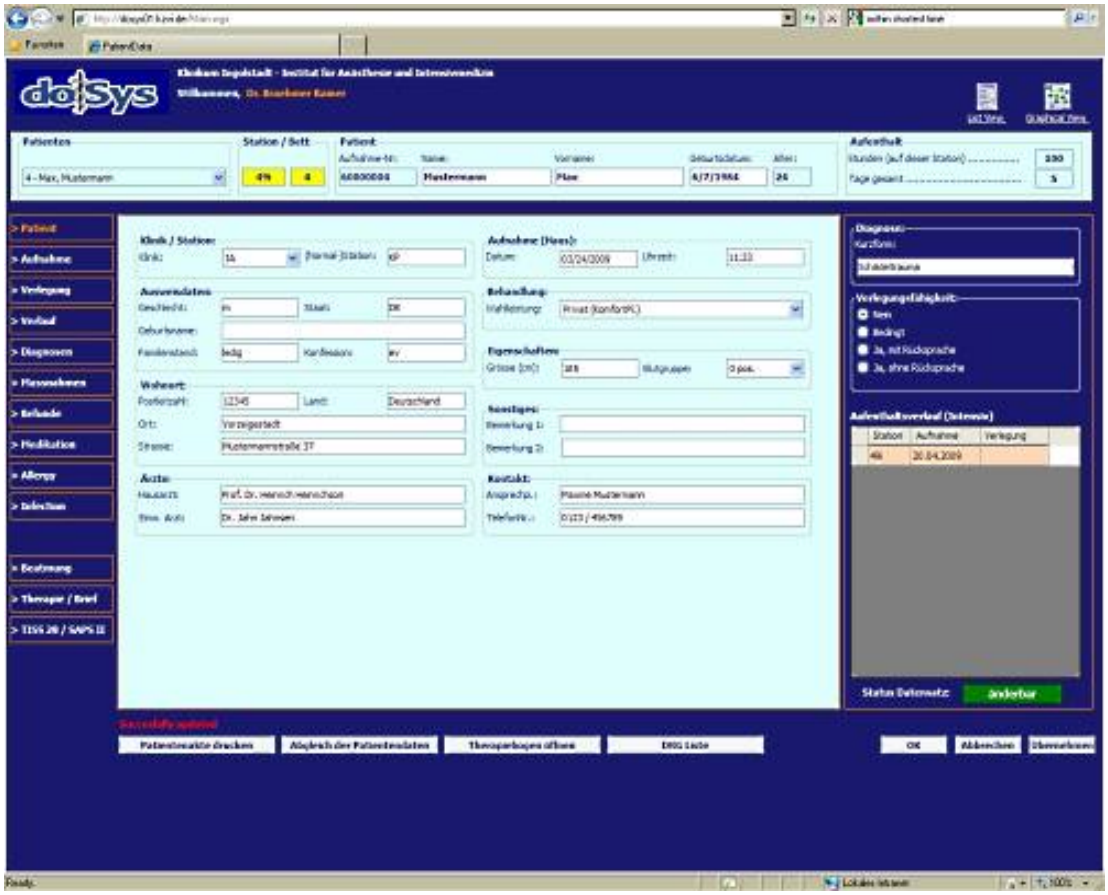

Figura 9: Exemplo de aplicação desenvolvida em VisualWebGui

A aplicação anterior chamada de doSys foi desenvolvida em Visual WebGui pelo facto de a equipa não ter experiência a desenvolver aplicações Web utilizando por exemplo ASP.NET, e da dificuldade maior de desenvolvimento em ASP.NET relativamente a uma aplicação Desktop normal em Visual Studio [MAR09].

Uma das grandes desvantagens a apontar à ferramenta passa pelo facto de esta não ser gratuita. O quadro seguinte expõe assim as principais vantagens e desvantagens deste sistema.

Tabela 8: Sumário da tecnologia VisualWebGui

Visual WebGui	
Vantagens	- Desenvolvimento como se de uma aplicação Desktop se trata-se - Não é necessário ter experiência da Web para desenvolver aplicações - Capacidade de converter aplicações Desktop tradicionais - Capacidade de exportação para tecnologia standard (DHTML)
Desvantagens	- Pouco útil para gerar conteúdos multimédia ou animações - Ferramenta não gratuita

2.8.2.6 OpenLaszlo

Esta tecnologia, mais orientada à multimédia e animação que o Visual WebGui, tem como mesma vantagem que este anterior o facto de permitir exportar a aplicação para DHTML ou Flash. Esta é uma mais-valia relativamente a Flex, Silverlight, etc. pois é possível desenvolver uma aplicação que possa ser compilada para os dois tipos de formatos de RIA que são os mais compatíveis actualmente [RIASTAT].

O projecto OpenLaszlo é Open Source e pertence à empresa Laszlo Systems[13]. A linguagem de programação usada é o LZX. É uma linguagem que tem como objectivo permitir uma familiarização com desenvolvedores habituados a tecnologias Web como HTML ou Javascript.

Do ponto de vista funcional e visual as diferenças entre uma aplicação desenvolvida em OpenLaszlo exportadas para DHTML ou para Flash são mínimas e imperceptíveis.

Figura 10: Aplicação desenvolvida em OpenLaszlo executada na versão DHTML

Na Figura 10 está um exemplo de aplicação desenvolvida em OpenLaszlo, em DHTML. Existe uma versão Flash desta mesma aplicação sem diferenças aparentes em termos de usabilidade e funcionalidades[14].

No quadro seguinte são apresentadas as principais vantagens e desvantagens desta solução.

13 http://www.laszlosystems.com

14 http://www.openlaszlo.org/showcase

Tabela 9: Sumário da tecnologia OpenLaszlo

OpenLaszlo	
Vantagens	- Capacidade de compilação para DHTML e Flash - Open Source e gratuito
Desvantagens	- Não apresenta as mesmas facilidades de desenvolvimento como Flex ou Silverlight

2.8.2.7 Google Gears

Uma solução para aumentar as capacidades de um Browser Web poderia ter por base a pouco tempo a tecnologia do Google com o nome Gears. Esta tecnologia recentemente descontinuada foi bastante utilizada no desenvolvimento de aplicações Web, permitindo efectuar interacções mais ricas com a aplicação.

O Google Gears passa pela instalação por parte do cliente deste software, aumentando as características do Browser Web. Torna-se um projecto diferente que os plug-ins para Browser como Flash ou Silverlight pelo facto de em vez de trazer novos tipos de conteúdo ao Browser adiciona funcionalidades a este.

Como principais funcionalidades inovadoras do Gears temos:

- Armazenamento de dados por parte da aplicação Web no lado do cliente utilizando uma base de dados local;
- Interacções mais ricas com o desktop como a possibilidade de utilizarmos *drag and drop* entre um ficheiro do sistema de ficheiros local e página Web.
- Capacidade de executar código Javascript em background;

Não tendo como objectivo a multimédia ou a interactividade da aplicação o Google Gears inova mais por adicionar um conjunto de funcionalidades ao Browser por meio de uma API interessantes para aplicações Web. Existindo versões para os principais sistemas operativos, peca por apenas ser compatível com alguns browsers. A causa apontada para a descontinuação do projecto Gears é o surgimento do HTML5 e a intenção de adicionar as principais funcionalidades deste ao novo standard.

Tabela 10: Sumário da tecnologia Google Gears

Google Gears		
Vantagens	–	API com funcionalidades interessantes
	–	Capacidade de funcionamento offline
	–	Possibilidade de d*rag and drop* de ficheiros para a aplicação
Desvantagens	–	Descontinuado
	–	Não tem funcionalidades de multimédia

2.8.3 Desktop Based

2.8.3.1 Adobe AIR

A tecnologia AIR da Adobe vem em seguimento da tecnologia Flex, adicionando a esta a funcionalidade de exportar esta tecnologia para o Desktop. Foi a primeira tecnologia deste género, surgindo à pouco mais de dois anos. Uma das grandes vantagens de correr fora do Browser é a capacidade de funcionamento offline da RIA. Como se trata de uma aplicação Desktop a persistência dos dados no sistema de ficheiros local não foi deixada de parte, o AIR para além de permitir o acesso ao sistema de ficheiros local tem também integrado o motor de base de dados SQLite [TUCDAV08].

Uma das mais-valias desta solução é a capacidade de gerar aplicações desktop que correm praticamente da mesma forma nos principais sistemas operativos, permitindo assim criar aplicações sem restrições de plataforma. Como exemplos de vantagem da aplicação fora do Browser relativamente ao Flash é a possibilidade de correr a aplicação mesmo sem conectividade à rede e o facto de ter integrado um Web Browser.

Como exemplo de uma aplicação feita em AIR temos o eBay Desktop. Este em tudo se assemelha ao sítio Web do eBay, mas com todo um conjunto de funcionalidades e interacções mais ricas em termos de interacção com o utilizador.

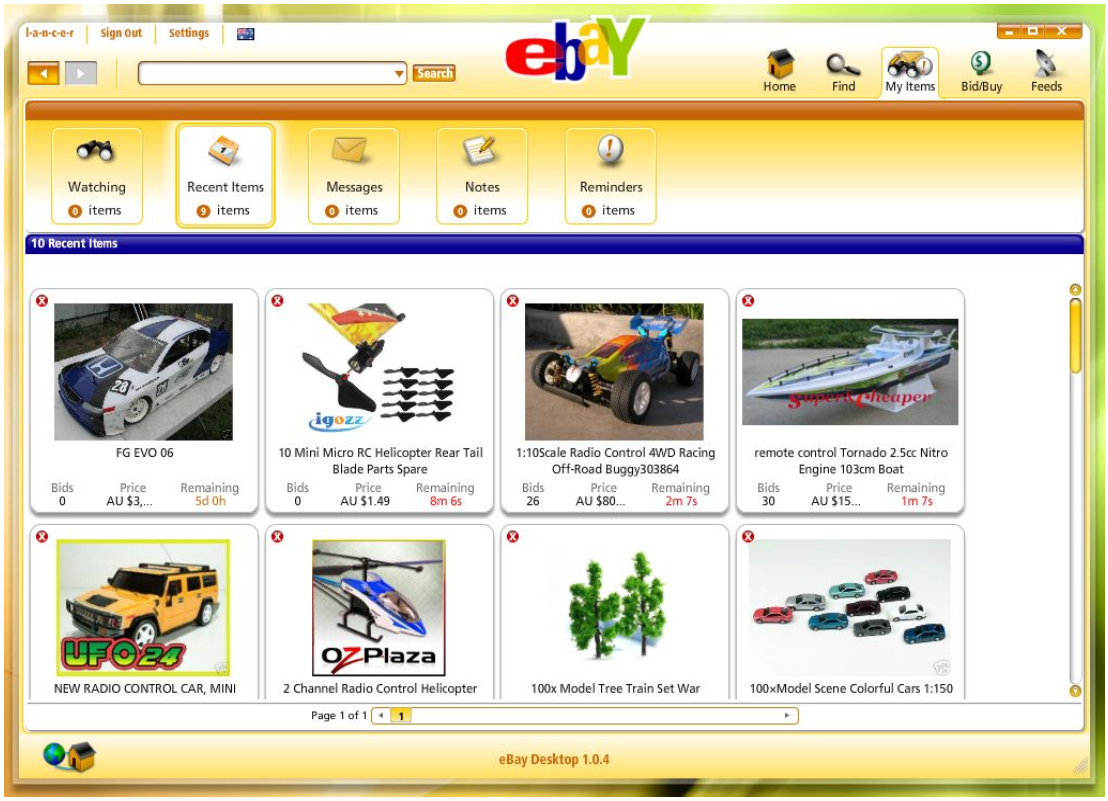

Figura 11: eBay Desktop, Exemplo de Aplicação desenvolvida em AIR a correr no Desktop

Na Figura 11 pode-se ver a aplicação eBay Desktop. Trata-se na realidade de uma aplicação Desktop com as funcionalidades do sítio Web.

Tabela 11: Sumário da tecnologia Adobe Air

AIR	
Vantagens	- Desktop based - Compatível com os principais sistema operativos (Linux, Windows e Mac) - Facilidades de desenvolvimento de aplicações Web tal como Flex
Desvantagens	- Liguagem ActionScript não tem suporte para processamento multithread

2.8.3.2 WebKit

O Webkit diferencia-se das tecnologias anteriores principalmente por não ser uma Framework de desenvolvimento de RIA, mas sim um motor de Browser Open Source.

Faz sentido falar desta tecnologia pois uma possibilidade de construção de uma RIA é a criação de uma aplicação Desktop com este motor de Browser integrado, podendo assim adicionar novas funcionalidades a uma aplicação Web já existente com todo o poder do Desktop. Neste tipo de solução teríamos a vantagem de não ter limitações de uma Framework e pode-se justificar caso não exista mais nenhuma tecnologia que permita realizar certos tipos de

requisitos necessários à RIA. Webkit é desenvolvido na linguagem C++, o que possibilita juntar numa mesma aplicação um motor de Browser com tecnologias 3D como OpenGL ou Microsoft DirectX. Permite o acesso directo aos serviços do sistema operativo podendo ser o mais optimizado possível caso a performance seja importante. Caso a aplicação necessite de aceder a hardware diferente da câmara Web ou da placa de som, tais como periféricos específicos é necessária uma solução deste género. Como exemplo de RIA a este nível temos o iTunes da Apple.

O Webkit como motor de Browser, permite a geração de uma janela com a página Web, com facilidades de acesso e manipulação da página por acesso ao DOM por parte da aplicação. Se o objectivo da RIA for a criação de uma aplicação multimédia ou com maior animação, o desenvolvimento desta será muito mais complicado do que usando uma tecnologia das anteriores. O facto de ser uma aplicação C++ de raiz, leva a que não disponhamos de ferramentas ou utensílios já concebidos para esses fins usuais na Web e em tecnologias para RIA.

A compatibilidade com vários sistemas operativos pode ser conseguida mas nunca sem pequenas alterações na aplicação, o que leva a que o desenvolvimento seja mais demorado do que em Frameworks multiplataforma.

Resumindo, o quadro seguinte pretende mostrar as principais vantagens e desvantagens do uso deste método para desenvolver uma RIA.

Tabela 12: Sumário das vantagens da utilização do motor WebKit para a construção de uma RIA

Webkit	
Vantagens	- Sem limitações de uma Framework - Capacidade de integração com tecnologias poderosas não disponíveis em Frameworks RIA (OpenGL, DirectX) - Capacidade de acesso a periféricos - Eficiência
Desvantagens	- Pouco adequada a animações ou multimédia - Compatibilidade com múltiplos sistemas operativos não garantida

2.8.3.3 Firefox, Chrome, Safari e Opera

O Mozilla Firefox (Browser Web Open Source com grande cota de mercado) surge aqui como uma base para novas funcionalidades possíveis numa aplicação Web, através da expansão do Browser específico pelo uso de add-ons.

O Mozilla Firefox permite expandir o Browser de duas formas diferentes. A expansão através de "*Extensions*" mais não é que criar uma interface em HTML com possibilidade de utilizar Javascript como linguagem de programação. A "Extension" tem acesso à Web e ao DOM das páginas do Browser. Esta solução apresenta-se claramente limitada e pouco ou nada traz de novo relativamente a uma RIA Ajax, estando a sua vantagem na interface do Browser e não tanto na interface da aplicação Web.

Uma segunda forma de adicionar funcionalidades a um Browser é através de plugins, Plugins neste caso são pedaços de software compilado que o Browser usa para executar

conteúdo que não conhece. Os plugins para o Mozilla Firefox, Google Chrome, Safari da Apple ou Opera são escritos em NPAPI (Netscape Plugin Application Programming Interface) que é implementado por vários browsers com a grande excepção do Microsoft Internet Explorer. Plugins servem assim o propósito de adicionar funcionalidades ao Browser mas sempre de uma forma limitada. É possível criar animações e gráficos simples usando as classes SVG (Scalable Vector Graphics) e Canvas.

Em comparação com Flash, Flex, JavaFX ou Silverlight não se vê vantagem nenhuma nesta solução excepto em situações pontuais em que as tecnologias enumeradas não dêem resposta.

Analisando esta solução podemos ter o seguinte quadro resumo.

Tabela 13: Sumário do uso de add-ons para Browsers

Firefox, Chrome, Safari e Opera	
Vantagens	- Fácil de implementar caso se pretenda algo como um Thin Client - Compatível com vários Sistemas Operativos
Desvantagens	- Multimédia e animações possíveis mas de baixo nível - Necessita instalação do lado do cliente - Na generalidade não traz vantagens relativamente a Frameworks Web Based

2.8.3.4 WPF (Windows Presentation Foundation)

Esta solução tecnológica da Microsoft aparece aqui como uma alternativa ao WebKit, Silverlight ou Adobe Air. WPF é um subsistema gráfico presente em .NET, que permite a criação de aplicações gráficas ricas para sistemas Windows a partir da versão XP SP2. Silverlight consiste num subsistema de WPF logo sendo mais limitado que este.

A opção por WPF fará sentido caso o Silverlight não responda em termos de características ao que é necessário implementar. A grande desvantagem é o facto de apenas ser compatível com sistemas Windows.

Tabela 14: Sumário da tecnologia WPF

WPF	
Vantagens	- Grandes possibilidades de desenvolvimento, e grande capacidade da tecnologia
Desvantagens	- Não ser multiplataforma - Obrigatório a instalação na maquina do cliente

2.8.4 Comparação entre tecnologias

Como resumo das tecnologias apresentadas o quadro seguinte exprime o essencial. Tenta avaliar cada tecnologia do ponto de vista de várias métricas todas de grande importância para o projecto. Para simplificar são utilizados os valores: inexistente, fraco, médio, elevado e máximo. A estes podem ser adicionados os símbolos + e – significando que se encontram entre dois

patamares. É considerado que uma classificação de médio ou superior garante os requisitos necessários para este projecto em concreto. A métrica do desenvolvimento relaciona-se com a facilidade de desenvolver nessa plataforma. Padrões de interacção identificam a capacidade de interacção rica possível. Performance é relativa à performance de execução da aplicação rica. Capacidades multimédia são relativas às capacidades de mostrar animações, vídeo ou áudio. Capacidades de interligação com a base de dados relacionam a possibilidade de funcionamento sem ligação permanente à Web e facilidade de armazenamento de dados numa base de dados relacional. A interligação com motores de jogo é relativa à capacidade de incluir um motor de jogo na aplicação. A métrica da multiplataforma corresponde à compatibilidade da tecnologia com vários sistemas operativos, ou então com vários browsers.

Tabela 15: Comparação entre tecnologias Parte 1

Tecnologia	Desenvolvimento	Padrões de interacção	Performance
Ajax	Médio	Médio	Médio
Flash	Elevado	Elevado+	Elevado-
Flex	Elevado+	Elevado+	Elevado-
Silverlight	Máximo	Elevado+	Elevado
JavaFX	Médio+	Elevado	Médio+
Visual WebGui	Máximo	Médio	Médio
OpenLaszlo	Médio+	Elevado+	Elevado-
Google Gears	Médio	Fraco	Médio
AIR	Elevado+	Máximo	Elevado
Webkit	Fraco-	Médio	Máximo
Plugin NPAPI	Inexistente	Fraco	Elevado
WPF	Máximo	Máximo	Máximo

Tabela 16: Comparação entre tecnologias Parte 2

Tecnologia	Multimédia	Interligação com Base de Dados local	Interligação com Game Engines	Muti-plataforma
Ajax	Inexistente	Inexistente	Inexistente	Máximo
Flash	Máximo	Inexistente	Fraco	Máximo
Flex	Máximo	Inexistente	Fraco	Máximo
Silverlight	Máximo	Médio-	Máximo-	Médio
JavaFX	Máximo	Médio-	Fraco	Médio+
Visual WebGui	Inexistente	Fraco	Inexistente	Máximo
OpenLaszlo	Médio+	Fraco	Inexistente	Máximo
Google Gears	Inexistente	Máximo	Inexistente	Médio+
AIR	Máximo	Máximo	Fraco	Máximo
Webkit	Inexistente	Fraco	Máximo	Médio-
Plugin NPAPI	Inexistente	Inexistente	Inexistente	Médio-
WPF	Máximo	Médio+	Maximo-	Fraco

3 Revisão de tecnologias

3.1 Necessidades tecnológicas para este projecto

O projecto a implementar apresenta como requisitos principais:

- Capacidade de *drag and drop* entre ficheiros do sistema e a aplicação;
- Criação de trabalhos recorrendo a aplicações instaladas no sistema como Microsoft Word, Microsoft Power Point e Microsoft Excel;
- Capacidade de funcionamento offline da submissão de ficheiros e criação e edição de trabalhos offline;
- Possibilidade de correr a aplicação com acesso à internet ou não;
- Versão pretendida deve ser dentro do possível multiplataforma;
- A aplicação deve correr fora do Browser, com a instalação de um ícone no *ambiente de trabalho*.
- Deve envolver o mínimo de alterações a ser efectuadas no lado do servidor;
- Versão Desktop deve ter todas as funcionalidades de aplicação Web;

Tendo em conta as características anteriores é apresentado seguidamente uma análise de viabilidade de tecnologias. O último ponto da lista apresenta uma decisão importante quanto à RIA Desktop a desenvolver. Por forma a manter todas as funcionalidades actuais da aplicação Web foi decidido construir uma RIA Desktop do tipo Browser dedicado. Este tipo de abordagem encontra-se discutida em mais detalhe no ponto 4.2. Pelo facto de não conter um

Browser Web integrado JavaFx não foi incluído na lista de tecnologias candidatas a este projecto.

3.2 Adobe Air

Esta tecnologia da Adobe é uma candidata à implementação da solução. Contem um Web Browser integrado o que possibilita a criação de uma aplicação do tipo “Browser Dedicado”. Flash que é a base do AIR, é a tecnologia de RIA mais madura no mercado existindo actualmente vários exemplos de aplicações efectuadas em AIR[AIRGAL].

A funcionalidade de integração com aplicações locais não é possível em AIR, limitando a escolha desta tecnologia.

Uma aplicação do tipo “Browser dedicado” exige mais do que a possibilidade de mostrar conteúdo HTML estático, é necessário ter um conhecimento mais pormenorizado das funcionalidades do Browser do Adobe Air para concluir se é viável a implementação do projecto. Uma análise deste tecnologia sobre o ponto de vista de pormenores de implementação, em função dos testes efectuados, encontra-se presente no ponto 4.1.3 e 4.2.3.

3.3 Microsoft Silverlight

Actualmente na versão 4, que foi lançada em versão final durante o decorrer deste trabalho, apresenta-se como um concorrente ao Adobe Flash e ao Adobe Air, pois permite efectuar aplicações ricas tanto como plug-in para o Browser como aplicações de desktop.

Tal como o Adobe Air, vem com um Web Browser incorporado, o que torna possível em princípio a implementação de uma aplicação Web do tipo “Browser dedicado”. Mais uma vez a capacidade para o efectuar depende sempre das características da ferramenta WebBrowser do Silverlight.

Como grande vantagem é uma tecnologia que permite de forma facilitada a interacção com aplicações instaladas, indo ao encontro a um dos requisitos do projecto. Com Silverlight é possível interagir com todas as aplicações do Microsoft Office utilizando como tecnologia COM(Component Object Model) da Microsoft.

Mais detalhes sobre a tecnologia podem ser encontrados no ponto 4.1.3, 4.2.3 e no ponto Erro: Origem da referência não encontrada onde é apresentada a solução.

3.4 Web Kit

A mais valia desta tecnologia passa pelo conhecimento à partida da viabilidade de implementação de todas as funcionalidades do projecto. A grande desvantagem passa pela incapacidade de estimar o tempo necessário à implementação.

Uma aplicação RIA tendo por base unicamente a o motor de Browser WebKit apresenta-se como sendo um projecto de grande envergadura principalmente pelos seguintes motivos:

- Comunicação utilizado WebServices[WSERVC] SOAP[SOAP], REST[REST] ou de outra forma com a aplicação tem que ser suportada ou por projecto ou código externo ou então é necessário implementar utilizando a comunicação por sockets.
- Apesar de não ser requisito explicito para este projecto, mas com o intuito de o adicionar futuramente, a capacidade de multimédia é em muito inferior as capacidades já embutidas em tecnologias como AIR ou Silverlight. Alguns dos conteúdos presentes na Web tais como Windows Media Vídeo[WMVWIKI] ou Real Vídeo[RVIDEO], entre outros, necessitam de código complexo e por vezes fechado para o descodificar.
- Suporte de armazenamento em base de dados se necessário terá que ser suportado por código ou projecto externo que seria necessário adicionar à aplicação.
- Componentes como editores WYSIWYG, álbuns de fotos, etc. já existentes em AIR ou Silverlight teriam de ser totalmente implementados nesta aplicação.
- Compatibilidade com vários sistemas operativos ou plataformas teria que ser suportado pelo projecto, assim como compatibilidade com novas plataformas futuras.

Analisando os tópicos anteriores fica bem claro que a opção por esta solução é a que claramente necessita de maior esforço de implementação.

3.5 WPF

Windows Presentation Foundation apresenta um desenvolvimento em tudo semelhante com Silverlight, sendo este um subsistema de WPF. A existência de um controlo WebBrowser ainda com mais capacidades que o Silverlight aponta para esta solução como viável para a implementação do projecto com sucesso. A grande desvantagem é a falta de compatibilidade com vários sistemas operativos sendo apenas possível compilar a aplicação para Windows.

3.6 Conclusões

Analisando as necessidades do projecto e as tecnologias passíveis de o implementar dentro do tempo atribuído, foi escolhida a tecnologia Silverlight. Foi necessário trabalhar com as duas tecnologias de forma a escolher a que mais se adequava ou a que permitisse implementar o projecto proposto.

A grande dificuldade na escolha da tecnologia passa pelo facto de não existirem actualmente disponíveis bons exemplos exemplos de RIAs desktop implementadas utilizando

Air ou Silverlight do tipo “Browser dedicado”, mas apenas exemplos de RIAs do tipo cliente desktop. A definição e comparação entre estes dois tipos de RIA são abordadas no capítulo 4.

4 Abordagens *Out Of the Browser (OOB)*

Neste capítulo vão ser discutidos dois métodos de implementar uma versão standalone de uma aplicação Web. Os dois tipos serão definidos assim como apresentados exemplos de aplicações actuais desse tipo. O processo de desenvolvimento e o de reengenharia da aplicação Web é descrito de forma sucinta, aliado às necessidades tecnológicas que cada abordagem acarreta.

4.1 RIA OOB tipo cliente desktop

Este tipo de RIA é actualmente a que existe em maior escala pela Web, e a que mais exemplos existe no que a RIA Desktop diz respeito. Seguidamente é feita uma caracterização deste tipo de aplicação.

4.1.1 Definição

Uma RIA, que denominei do tipo cliente desktop, caracteriza-se por ser uma aplicação desktop que possibilita numa aplicação, em tudo típica de desktop, aceder a funcionalidades da aplicação Web.

Uma RIA deste tipo apresenta então como características principais:

- RIA semelhante em tudo a uma aplicação desktop;

- Não serve de substituto à aplicação Web, apenas adiciona certas funcionalidades ou interacções mais ricas, não implementando todas as funcionalidades da aplicação original.
- Normalmente é comum a interface da RIA desktop diferir bastante da interface da aplicação Web o que pode prejudicar a adopção por parte dos utilizadores.

Os grandes exemplos actualmente têm particularmente o propósito de dar a conhecer a tecnologia em que se encontra implementado, de forma a mostrar as potencialidades da mesma.

4.1.2 Exemplos

Uma lista não muito extensa de exemplos pode ser encontrado nas páginas das principais tecnologias, Microsoft Silverlight[15] e Adobe Air[16].

4.1.2.1 Ebay Desktop

O primeiro grande exemplo de uma aplicação RIA desktop feita em AIR é o eBay Desktop. Esta aplicação contempla uma grande parte das funcionalidades do Ebay, mas mais uma vez como na generalidade deste tipo de aplicações esta não pretende substituir a aplicação Web Ebay na sua versão de Browser

15 http://www.silverlight.net/showcase/

16 http://www.adobe.com/products/flashplatformruntimes/gallery/?promoid=GRLQA

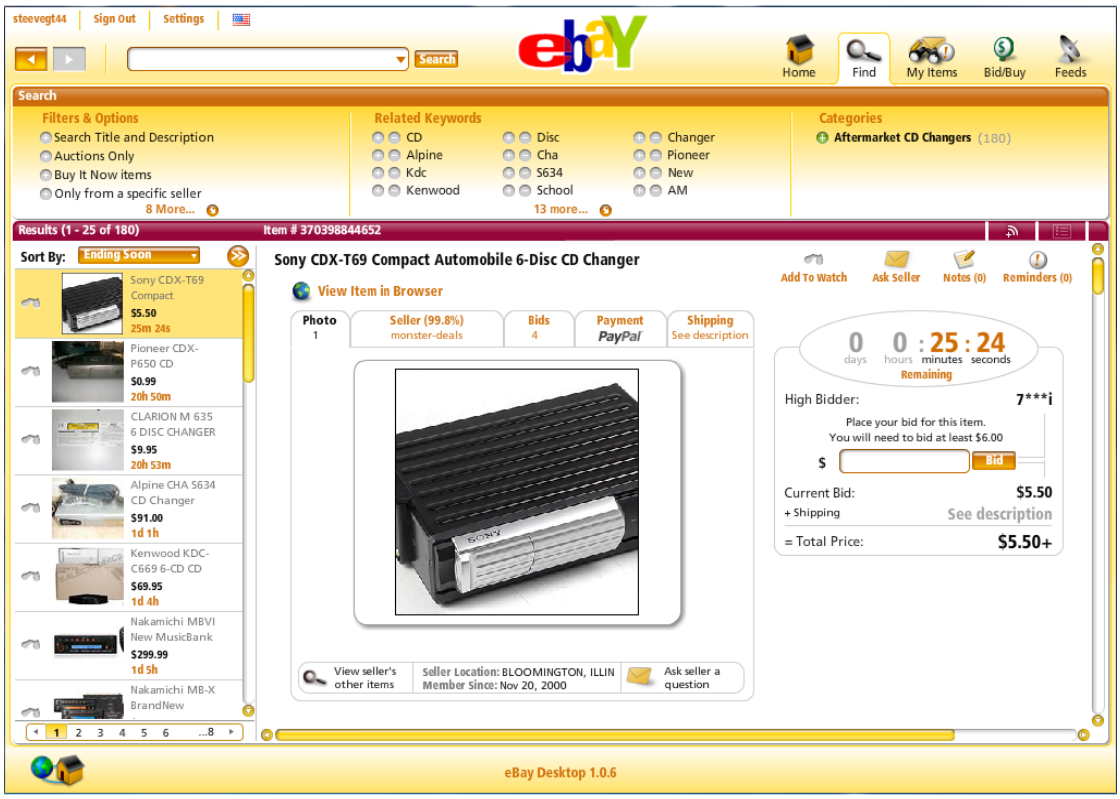

Figura 12: Ebay Desktop

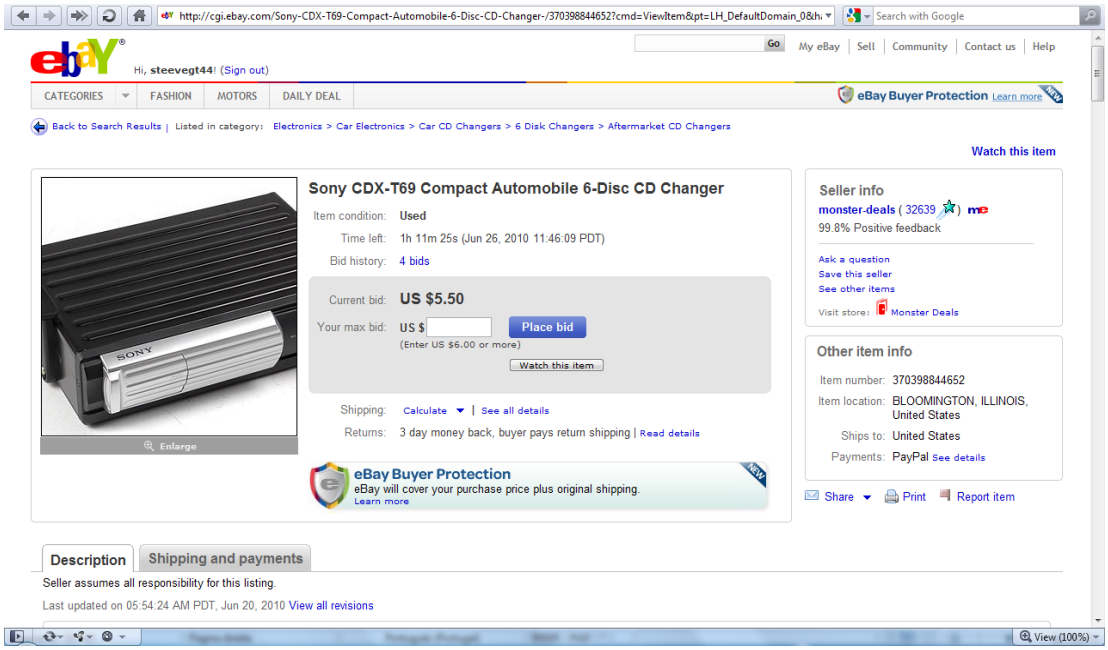

Figura 13: Ebay Web

Comparando a versão Desktop com a versão Web da aplicação Ebay nota-se claramente a diferença de interface existente. Apesar de existirem semelhanças verifica-se que as interfaces são bastante diferentes. O botão de colocar a aposta na versão Web tem o texto “Place bid” enquanto que na versão desktop o texto apenas é “Bid”. Outro exemplo de diferenças de interface entre as duas versões da aplicação é o facto de que na versão Web as informações relativas ao vendedor, envios, etc, encontram-se ao lado direito ou por baixo da imagem do produto, enquanto que na versão Desktop essa mesma informação encontra-se dividida por tabs encontrando-se a imagem do produto igualmente numa tab.

4.1.2.2 Microsoft Silverlight 4 Client for Facebook

Este exemplo desta vez implementado em Silverlight 4 foi desenvolvido ainda na versão 4 beta do Silverlight. É o primeiro e dos mais conhecidos exemplos de uma RIA Desktop para uma aplicação Web. Tal como o exemplo do Ebay esta aplicação Desktop é para uma grande e conhecida aplicação Web, o Facebook neste caso, e o grande objectivo desta aplicação é a promoção da tecnologia e um exemplo das capacidades desta.

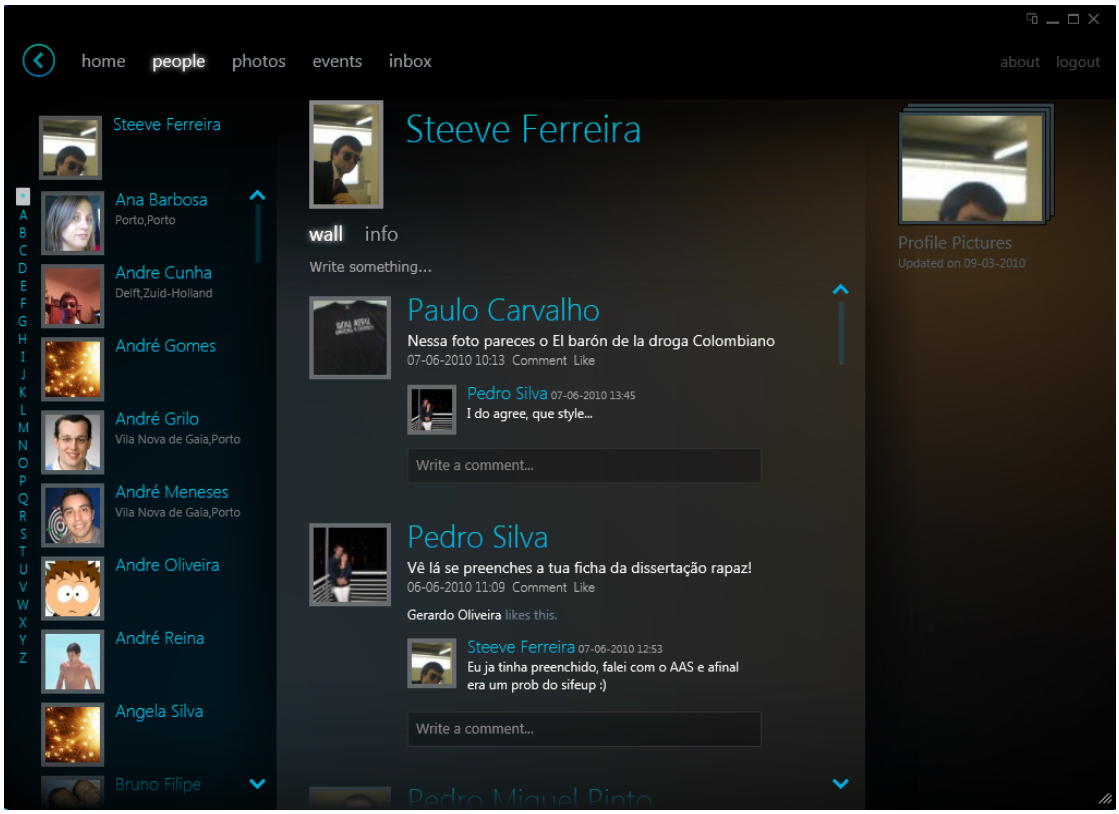

Figura 14: Microsoft Silverlight 4 Client for Facebook ecrã

Figura 15: Ecrã da versão Web da aplicação Facebook

Comparando as interfaces notam-se claras diferenças entre ambas como no exemplo anterior da aplicação Ebay Desktop. Em termos de funcionalidades a versão Desktop possibilita ao utilizador submeter imagens por drag and drop do sistema de ficheiros, assim como tirar fotografias directamente da câmara Web. Mais uma vez esta aplicação não pretende nem pode ser um substituto da aplicação Web visto que não é possível correr as aplicações do Facebook como a muito conhecida farmville, entre outras funcionalidades que não estão disponíveis.

4.1.3 Processo de desenvolvimento da RIA

O desenvolvimento de uma RIA deste tipo passa pela implementação da interface utilizando controlos disponíveis pela tecnologia a usar. Neste tipo de RIA em particular é necessário desenvolver de raiz toda a interface necessária para a aplicação Desktop. O processo é equivalente ao da construção de uma aplicação Desktop tradicional.

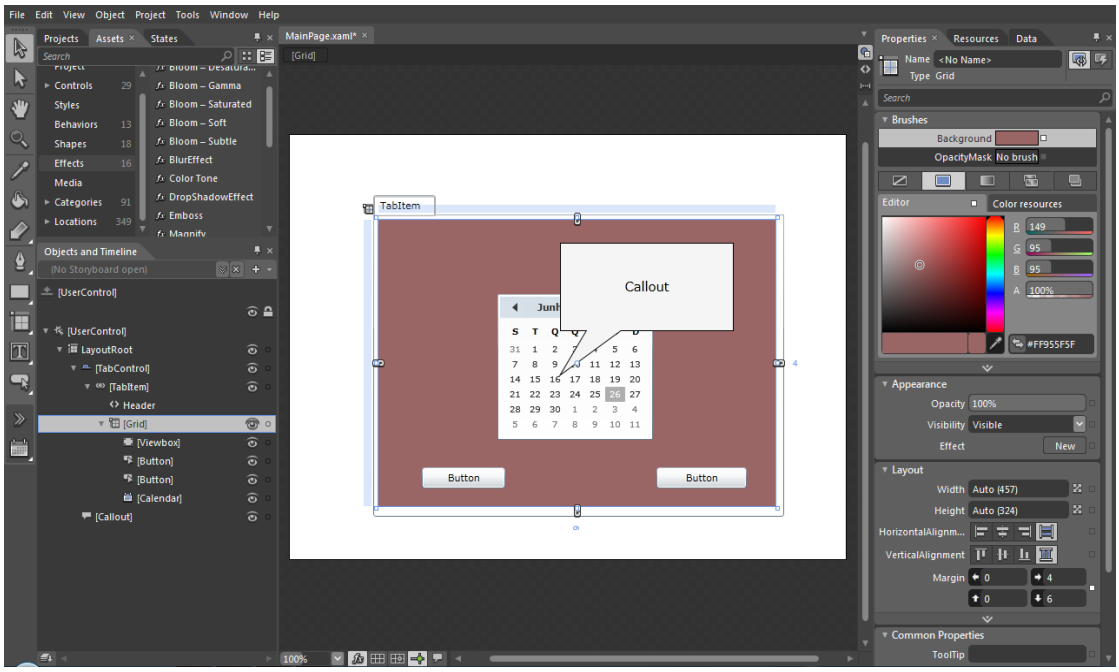

Figura 16: Ecrã de Microsoft Expression Blend

4.1.3.1 Ambiente de desenvolvimento

Na figura 16 é possível observar o ambiente de desenvolvimento de uma interface para uma RIA Silverlight utilizando a ferramenta da Microsoft Expression Blend. Esta ferramenta para além de trazer os componentes habituais de uma interface desktop (caixas de texto, botões, etc.), incorpora também um conjunto alargado de comportamentos e de animações a aplicar aos objectos da interface.

Para a tecnologia rival da Adobe, o AIR, o desenvolvimento de interfaces é igualmente facilitado, existindo ferramentas da Adobe para o seu desenvolvimento como o Flash Professional CS5 ou ferramentas open-source como o Aptana Studio da empresa Aptana. Em AIR para além do desenvolvimento em Flash ou Flex é possível desenvolver interfaces e a lógica da aplicação com tecnologias Web (HTML, Css e Javascript). Esta última solução pode ser vantajosa para desenvolvedores já com alguma experiência em tecnologias Web.

4.1.3.2 Persistência

O armazenamento da informação pode ser feito recorrendo a armazenamento em ficheiros de dados, visto que ambas as tecnologias permitem o acesso ou total ou em parte ao sistema de ficheiros local. Outra solução é o armazenamento utilizando uma base de dados relacional.

Adobe AIR inclui uma implementação do motor de base de dados SQLite, sendo de grande utilidade quando se pretende armazenar informação relacional, grandes volumes de dados e em que se pretenda tratar muita informação localmente. De qualquer das formas o uso de uma base

de dados relacional pode ser uma opção por mera comodidade de implementação, não sendo necessário processar ficheiros com dados binários ou de texto.

Em Silverlight não existe a possibilidade de utilizar uma base de dados local sem a utilização de código externo que o implemente. Existem algumas soluções gratuitas como o silverdb[17] e proprietárias como Siaqodb[18]. Outra forma de armazenar informação estruturada e sem a necessidade de processamento binário ou de texto é utilizando as capacidades de processamento de dados XML do Silverlight.

4.1.3.3 Comunicação Web

Para além da interface e da persistência locais é necessário comunicar com a aplicação Web de forma a sincronizar informação, obter e enviar dados. A forma especifica como se comunica com a aplicação depende sempre da tecnologia em que a aplicação Web se encontra implementada, mas normalmente a comunicação é efectuada ao nível da lógica de negócio utilizando os chamados Web Services ou serviços Web. A comunicação ao nível da base de dados não é claramente aconselhada por motivos de segurança.

Os Web Services mais não são que uma API da aplicação Web, podendo estes ser ou Big Web Services[WSERVC] ou RESTful services[REST].

- Big Web Services: utilizam como forma de comunicação o protocolo standard SOAP (Simple Object Access Protocol)[SOAP] que utiliza Xml como formato de mensagem na chamada ao Web Service. De forma simplificada um Web Service é uma chamada RPC(Remote Procedure Call) ao servidor Web em que os argumentos e retorno do procedimento são encapsulados numa mensagem XML. É comum o servidor fornecer uma descrição sobre os serviços existentes escrita em WSDL (Web Services Description Language). Uma das desvantagens deste tipo de arquitectura é o facto de não utilizar funcionalidades inerentes ao protocolo HTTP como a autenticação e cache. Como exemplo numa procedimento getUsers() é necessário enviar na chamada as credenciais de acesso do utilizador que se encontra autenticado na aplicação, pois o protocolo não contempla a autenticação já efectuada em HTTP ou em sessão na aplicação Web.
- RESTful Web Services: são serviços Web que se encontram de acordo com o estilo REST(Representational State Transfer). Este estilo de arquitectura, cada vez mais em moda nas aplicações Web 2.0, difere do SOAP RPC pelo facto de que este utiliza largamente as funcionalidades do HTTP e os métodos suportados em HTTP. Nesta arquitectura não é obrigatório utilizar XML como forma de escrever a mensagem.

17 http://silverdb.codeplex.com/
18 http://siaqodb.com/

Em Adobe Air existe a possibilidade de comunicar com uma aplicação Web utilizando os dois tipos de Web Service. A programação é um tudo semelhante a RPC, em que se cria um objecto proxy onde são executados os métodos remotos em ActionScript da mesma forma como se de um método local se trata-se. Em Silverlight o processo é o mesmo, existindo facilidade de comunicação com Big Web Services implementados em tecnologia Microsoft.

A comunicação com RESTful Web Services é mais complicada que a anterior em alguns dos casos. Como se baseia unicamente no protocolo HTTP as chamadas a estes serviços são feitas utilizando directamente chamadas HTTP ao servidor. A forma de transmitir as mensagem podem ser utilizando XML, Json, simples texto ou outras. Adobe Air tem compatibilidade com todos os Verbs(Métodos) do protocolo HTTP não existindo limitações de comunicação. Em Silverlight a classe responsável pela comunicação HTTP (HTTPWebRequest) apenas é compatível com os métodos GET e POST do protocolo.

De forma a contornar as limitações anteriores do Silverlight podemos optar pela seguinte opção:

- Dependendo do servidor Web utilizado este pode conter mecanismos de substituição do método HTTP. Um exemplo é a utilização do Header HTTP “X-HTTP-Method-Override” com o método pretendido. Outra forma de efectuar a substituição do método enviado pelo pretendido é utilizando sempre o método HTTP POST e codificar o método pretendido num parâmetro “_method”. Esta solução é apenas possível em aplicações Ruby On Rails.

4.1.4 Reengenharia da aplicação Web

A necessidade de reengenharia da aplicação Web para a construção de uma RIA do tipo cliente desktop é normalmente mínima devido ao facto de que uma aplicação Web 2.0 actual normalmente já fornecer uma API de comunicação, utilizando Web Services.

No caso de aplicações com tecnologia Microsoft, nomeadamente ASP.NET, a adição de Big Web Services à aplicação e o seu desenvolvimento é bastante facilitado, sendo esta uma boa solução pelo facto de poder poupar problemas de integração com a aplicação cliente, dependendo da tecnologia escolhida para a aplicação cliente como é obvio.

Qual o tipo de Web Services a implementar na aplicação Web é uma ponderação que envolve o que já se encontra implementado na aplicação em termos de Web Services, e em qual a tecnologia da aplicação Web se adequa melhor.

Aplicações implementadas utilizando largamente o paradigma AJAX, implica que já exista um conjunto de pontos de abertura da aplicação (normalmente REST) que funciona como a API do nível de lógica de negócio da aplicação. Aplicações nestas condições podem sofrer poucas ou nenhumas alterações para suportar os clientes Desktop.

4.1.5 Necessidades/opções tecnológicas

Do ponto de vista tecnológico as melhores tecnologias actuais, tendo como prova os exemplos existentes neste tipo de aplicações, são o AIR da Adobe e o Silverlight da Microsoft.

Caso a multi-plataforma e a gestão automática de actualizações da aplicação não seja uma questão importante WPF é igualmente uma boa escolha.

4.1.6 Conclusões da abordagem

Neste tipo de abordagem em sumário temos as vantagens das RIA desktop, mas igualmente um conjunto de desvantagens que é necessário ponderar:

- Interface difere da interface da aplicação Web podendo-se tornar uma barreira à utilização. Efectuar uma interface semelhante é aconselhável é possível mas é necessário manter os dois projectos em paralelo, e actualizar a interface da aplicação desktop aquando da actualização da aplicação Web.
- Implementação de todas as funcionalidades da aplicação Web por vezes pode não ser viável, especialmente em aplicação Web médias e grandes, o que implica que a aplicação Desktop não serve de substituto para a aplicação Web, levando a que os utilizadores optem pela totalmente funcional aplicação Web.
- Novas funcionalidades na aplicação Web necessitam de ser implementadas na versão Desktop para se tornarem acessíveis na aplicação cliente.

4.2 Ria OOB tipo “Browser Dedicado”

Não sendo uma RIA com muitos exemplos, este tipo de aplicação rica tem como principal diferença a capacidade de substituição na totalidade a aplicação Web, pois a implementação passa por incluir um Web Browser dentro da RIA. Esta capacidade como se vai ver a seguir comporta maior problemas de implementação do ponto de vista tecnológico.

4.2.1 Definição

Possivelmente o primeiro grande exemplo de RIA, o Apple iTunes, é uma RIA do tipo Browser dedicado, onde existe uma fusão total entre a aplicação Web e a aplicação Desktop.

Partindo do exemplo do iTunes e generalizando podemos definir uma RIA do tipo Browser dedicado através das seguintes características:

- Interface da aplicação desktop encontra-se fundida com a interface Web.

- Implementa todas as funcionalidades da aplicação Web, pois incorpora um Web Browser. Este pode passar despercebido ao utilizador, pela fusão das interfaces Web e Desktop.
- Sem diferenças significativas entre a interface unicamente Web e a interface Desktop.

4.2.2 Exemplos

O grande exemplo neste tipo de RIA é a aplicação iTunes da Apple. O iTunes é uma aplicação de gestão de conteúdo do iPod e iPhone da Apple, um gestor e leitor multimédia, assim como uma aplicação Web de compra e download de conteúdo áudio-visual.

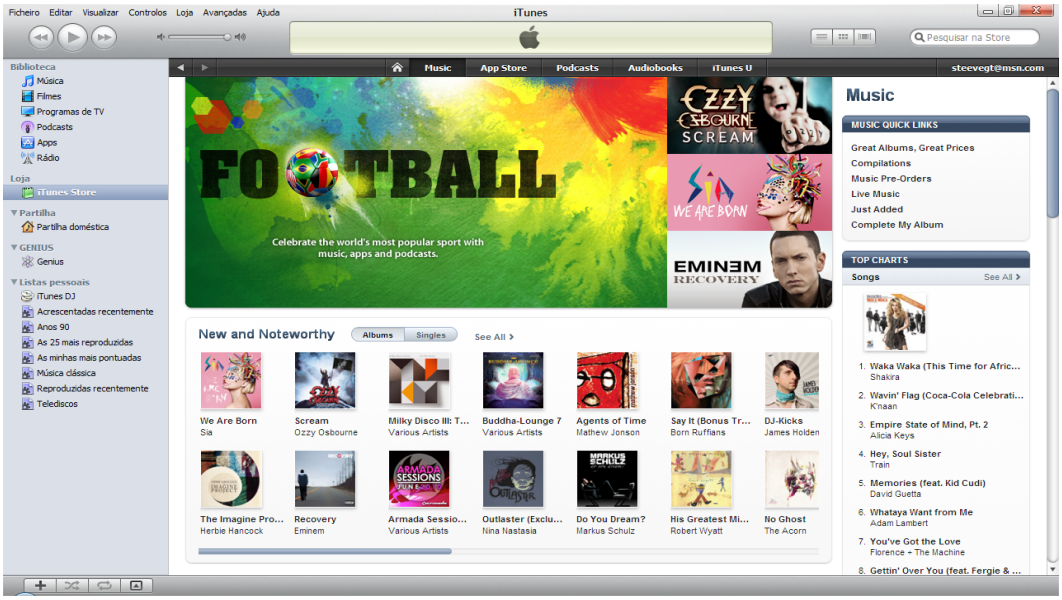

Figura 17: Ecrã da aplicação iTunes

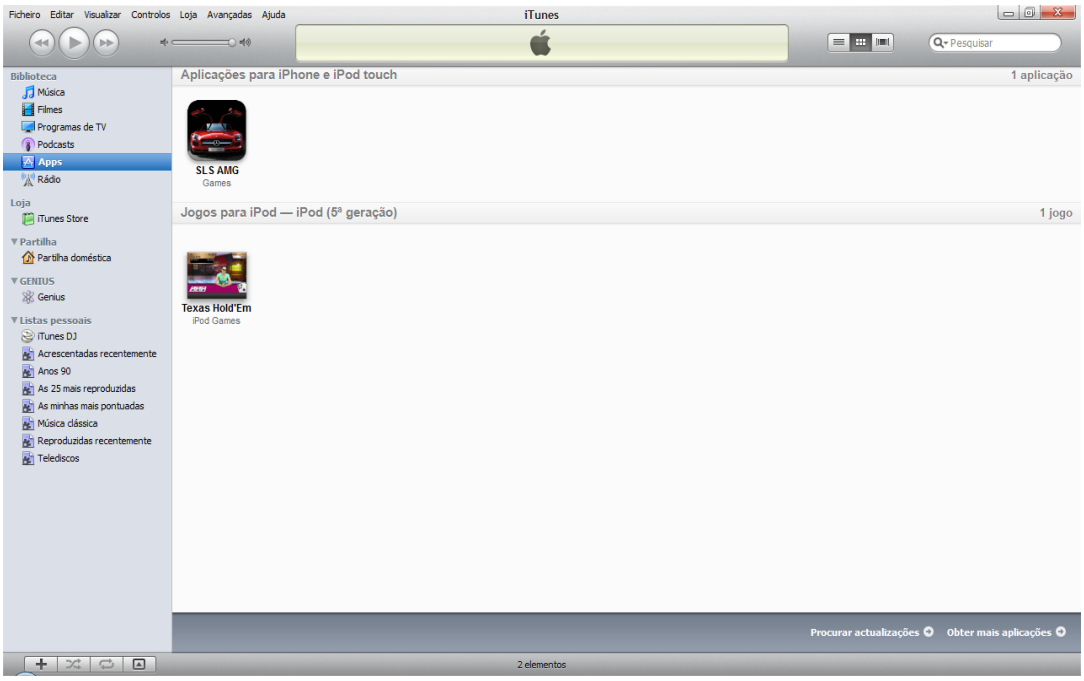

Figura 18: Ecrã da aplicação iTunes

Na Figura 18 é possível observar o Browser integrado na aplicação em que a aplicação Web se encontra totalmente integrada na aplicação Desktop. A integração entre a aplicação Web e a aplicação Desktop verifica-se aquando da transferência do conteúdo, que adiciona automaticamente este à biblioteca de musica, ou ao conjunto de aplicações descarregadas.

4.2.3 Processo de desenvolvimento da RIA

4.2.3.1 Construção da interface

Contrariamente ao processo de desenvolvimento de uma RIA do tipo cliente desktop, a interface desta encontra-se dividida entre a interface da aplicação Web e a interface da aplicação Desktop. O desenvolvimento da interface Desktop é efectuado da mesma forma e com recurso às mesmas ferramentas discutidas no ponto 4.1.3. A interface Web poderá manter-se a mesma, ou então adicionados componentes de interface na aplicação Desktop.

Na implementação de novas funcionalidades a serem mostradas na interface Web existem duas alternativas de implementação:

- Implementar adicionando controlos HTML (botão, caixa de texto, etc.) na aplicação Web e capturar os eventos deste no código da aplicação Desktop;
- Utilizar controlos existentes na tecnologia de RIA (Silverlight, AIR, WPF) e desenha-los por cima da interface Web.

A expansão da interface usando HTML exige alterações na aplicação Web, podendo estas ser realizadas tanto do lado do servidor como do lado do cliente. A realização das alterações por parte do servidor é abordada no ponto 4.2.4. Efectuar as alterações por parte da aplicação cliente exige mudar o conteúdo HTML da página Web obtida. Estas alterações podem ser feitas utilizando mecanismos de DOM presentes em AIR para o Browser integrado, ou pela execução de Javascript no Browser activado pelo código da aplicação em Silverlight. Tanto em AIR como em Silverlight é preferível a alteração no lado do servidor Web, de forma a ter a interface Web apenas definida num sítio.

A inserção de controlos da tecnologia de RIA (e.g. leitor multimédia, algum de fotografias, etc.) tem como base a adição desse controlo em cima do Browser Web integrado. Para um alinhamento correcto é necessário saber qual a posição e dimensões do espaço onde se vai inserir esse controlo dentro do Browser Web integrado. É necessária a existência de um contentor (tag em HTML) como referência de alinhamento para esse conteúdo. Em Silverlight a posição pode ser obtida recorrendo à execução de Javascript e em AIR recorrendo às capacidade de DOM da tecnologia.

4.2.3.2 Offline

A funcionalidade que implica maiores problemas a esta solução é a capacidade de funcionamento offline da aplicação. Como parte da interface é obtida do servidor Web é necessário persistir a interface (código HTML) localmente, e posteriormente carregar esta interface quando o acesso à rede não se encontra disponível. Este processo actualmente envolve problemas com as tecnologias actuais, principalmente por questões de segurança, que envolvem utilizar o Web Browser integrado para abrir conteúdo local.

Em Adobe Air o Browser integrado apresenta algumas diferenças de implementação relativamente ao suposto equivalente Safari, notando-se diferenças particularmente na implementação de CSS. Para além desse problema o Browser integrado não possibilita a execução de código Javascript nem conteúdo de plug-ins como Flash ou outros limitando claramente a opção deste tipo de RIA com esta tecnologia. Assim a criação de uma RIA do tipo Browser dedicado utilizando AIR apenas é possível em aplicações Web que de nada têm de AJAX, plug-ins e nem Javascript, ou seja, páginas Web 1.0.

A tecnologia Silverlight da Microsoft apresenta igualmente limitações no que ao Browser integrado diz respeito. Contrariamente ao Adobe Air em que o Browser se encontra dentro da compilação da aplicação o Silverlight utiliza ou o Internet Explorer em Windows ou o Safari em Macintosh para apresentar conteúdo HTML. Desta forma é possível ver todo o tipo de conteúdo pelo Browser de uma aplicação Silverlight, suportando Javascript e plugins desde que instalados na máquina onde corre a aplicação.

Relativamente ao que conteúdo local diz respeito o Browser integrado do Silverlight abre ficheiros HTML locais, mas não permite que este inclua ficheiros Javascript (tag HTML

<script>) em que a fonte é um ficheiro local. Isto pode ser contornado colocando o conteúdo do ficheiro contendo código Javascript embutido no ficheiro HTML.

Outra limitação que existe é nas imagens de uma página HTML (tag <img> de HTML) em que a fonte não pode ser um ficheiro de imagem local. Para contornar esta situação pode-se adicionar a imagem *inline* com o ficheiro HTML guardando a imagem no atributo src da tag <img>. A este forma de armazenar conteúdo no ficheiro HTML chama-se de *data URI scheme*[URISCHM]. De acordo com os standards[URISCHM] o tamanho máximo de uma imagem armazenada desta forma deve ser de 1KB, mas felizmente os browsers suportam tamanhos maiores. No caso do Internet Explorer este suporta imagens inline com o HTML até 32KB, e no Safari não existe restrição de tamanho. Como em Windows o Browser integrado no Silverlight utiliza o Internet Explorer é necessário reduzir o tamanho das imagens que tenham mais que 32KB, comprimindo-as por exemplo para o formato JPEG (Joint Photographic Experts Group) que sendo um formato de compressão *lossy*(que origina percas) permite ajustar a compressão de forma a ser sempre possível obter um ficheiro representativo da imagem inicial mas que se encontre dentro da restrição de espaço existente.

Para além de armazenar a interface da aplicação em ficheiros HTML de forma a esta se encontrar disponível na versão *offline* da aplicação é necessário ainda não permitir efectuar os pedidos AJAX da aplicação Web caso esta os faça. Quando a aplicação se encontra offline não é possível comunicar com o servidor, como é obvio, logo em situações em que a interface necessite de ser carregada por chamadas assíncronas ao servidor utilizando AJAX a solução de armazenar a página em HTML torna-se insuficiente, pois a interface é mais que o código HTML obtido no primeiro pedido HTTP. Para resolver esta questão existem três opções:

- Redesenhar a aplicação Web de forma a que a interface não seja carregada por AJAX. Por vezes isto pode ter bastante custo para a aplicação Web não sendo por vezes uma solução viável;
- Implementar em Javascript a falha do pedido AJAX, de forma a que caso não seja possível aceder ao servidor a interface seja clara dessa situação. Como exemplo existe o Gmail da Google que quando existe um problema de conexão altera a interface de forma a esta não se tornar inconsistente.
- Implementar dentro da RIA um pequeno servidor Web que reencaminhe as chamadas AJAX e os pedidos do Browser para o servidor, quando online, e que responda de forma a não tornar a interface inconsistente quando em offline. Esta solução apenas é possível em AIR da Adobe, pois actualmente o Silverlight não tem suporte para *Server Sockets* [SERVSOCK].

A grande dificuldade das aplicações do tipo Browser dedicado relativamente às RIAs do tipo cliente desktop passa efectivamente pela divisão da interface, e dos problemas em trabalhar e juntar num só entre os dois componentes da interface: parte Web e parte Desktop.

A forma de implementar a comunicação entre a a parte Desktop e o servidor é a mesma abordada no ponto 4.1.3.

Usando esta abordagem é aconselhável utilizar o mecanismo de autenticação já existente na aplicação Web, podendo ser detectado se o utilizador se encontra autenticado ou não recorrendo mais uma vez a Web Services.

4.2.4 Reengenharia da aplicação Web

Para além das questões abordadas em 4.1.4 referentes à comunicação entre a aplicação Web e a aplicação Desktop utilizando Web Services existe a possibilidade de adicionar controlos Web à aplicação para serem mostrados unicamente no Browser da versão Desktop.

A forma de se detectar do lado do servidor que o acesso a esta foi de um Browser comum ou da aplicação Desktop depende particularmente da tecnologia usada. No caso de utilizarmos AIR da Adobe existe a possibilidade de enviarmos no cabeçalho dos pedidos HTTP um *User Agent* [USRAGNT] com informação identificadora da aplicação Desktop. Em Silverlight isso não é possível, visto que ele envia um *User Agent* igual ao do Internet Explorer caso nos encontramos em Windows ou então igual ao do Safari caso estejamos em Mac. Para contornar a situação, a solução para o problema consiste na detecção, no lado do cliente da aplicação Web, de que o Browser usado pertence à aplicação Desktop, e posteriormente enviar essa informação para o servidor por AJAX. O servidor com esses dados armazena em variáveis de sessão essa informação, tomando conhecimento o servidor a partir desse ponto que os pedidos HTTP feitos por aquele Browser pertencem à RIA Desktop. Em Javascript a forma de detectar que nos encontramos no Browser da RIA Desktop é analisando se existe ou não o método "window.external.notify" que apenas existe dentro do Browser integrado no Silverlight.

Com a informação de que nos encontramos no Browser da RIA Desktop ou num Browser comum permite assim desenvolver interfaces diferentes ou adicionar funcionalidades à aplicação. A comunicação com a aplicação Desktop a partir da interface Web é efectuada recorrendo a uma função Javascript existente no Browser da RIA que permite notificar a aplicação desktop, funcionando como ponte de comunicação entre ambos.

4.2.5 Necessidades/opções tecnológicas

A opção Silverlight ou WPF da Microsoft é a mais indicada para implementar este tipo de RIA em que se procura uma interligação entre a interface Web já existente e a interface Desktop da aplicação. Adobe AIR à partida não é a mais indicada para este tipo de RIA Desktop pelo facto de que o Web Browser existente ter limitações comprometedoras tais como a impossibilidade de executar Javascript, podendo limitar as funcionalidades da aplicação Web.

A criação de uma aplicação utilizando um motor de Browser como o WebKit é uma possibilidade a ter em conta. Pelas limitações encontradas tanto em Silverlight como em AIR no Browser integrado pode-se concluir que estas tecnologias não foram pensadas para este tipo de uso.

4.2.6 Conclusões da abordagem

Este tipo de abordagem apresenta vantagens relativamente ao tipo cliente de Desktop. A principal vantagem é o facto de que esta RIA serve em todos os aspectos como substituta da aplicação Web existente, implementando todas as funcionalidades da aplicação sem existir a necessidade de as implementar do lado da RIA. Outra vantagem é a possibilidade de actualizar funcionalidades à aplicação Web sem existir a necessidade de actualizar a aplicação Desktop.

As tecnologias de RIA Desktop existentes actualmente não são o mais indicado para este tipo de abordagem, principalmente pelas limitações e dificuldades existentes na utilização do Browser integrado. Acredito que com a evolução da tecnologia e possivelmente com o surgimento de mais exemplos de aplicações RIA deste tipo a tendencia seja para enriquecer em termos de funcionalidades e capacidades os browsers integrados de Silverlight e de AIR, e possivelmente até o surgimento de outras tecnologias.

5 Escolinhas Desktop

5.1 Contexto

Esta dissertação e projecto insere-se na unidade curricular Dissertação do Mestrado Integrado em Engenharia Informática e Computação. Esta dissertação contempla a criação de uma protótipo funcional do produto escolinhas.pt, da empresa Tecla Colorida.

A Tecla Colorida é um spin-off da FEUP e INESC Porto, fundada em Dezembro de 2008, especializada no desenvolvimento de plataformas web 2.0 e soluções para fins educativos. O principal produto desta empresa é a plataforma colaborativa e social para o ensino básico o escolinhas.pt [ESC10].

5.2 Requisitos Funcionais

Seguidamente são apresentados os requisitos funcionais da aplicação "Escolinhas Desktop" sobre a forma de casos de utilização.

5.2.1 Submissão offline de ficheiros para o escolinhas

Neste caso de utilização o utilizador da aplicação enquanto offline arrasta um ou mais ficheiros para a aplicação. Quando largados os ficheiros sobre a aplicação esta notifica o utilizador que estes foram adicionados à aplicação, mas que a sua submição para o servidor Web se encontra pendente pelo facto de que no momento não existe conecção à internet.

Este caso de utilização tem como requisito o utilizador se encontrar autenticado na aplicação e sem ligação à internet no momento.

Quando for restabelecida a ligação à internet na actual execução da aplicação, ou numa próxima execução, é apresentada na interface um notificação para os ficheiros em envio assim como um confirmação de que o envio para o servidor foi efectuado com sucesso.

É necessário que o utilizador se tenha autenticado previamente na aplicação.

5.2.2 Criação de trabalhos utilizando ferramentas do Microsoft Office

Este caso de utilização contempla a criação de trabalhos utilizando o Microsoft Office Word, Microsoft Office Excel e Microsoft Office Power Point.

O utilizador dentro da interface do escolinhas.pt clica em “Criar novo trabalho Word” ou “Criar novo trabalho Excel” ou então “Criar novo trabalho Power Point”. Após o clique aparece uma nova janela com uma área de texto onde o utilizador preenche o título do novo documento. Quando clica em “OK” abre a aplicação do Microsoft Office numa nova janela com um novo documento já aberto.

Quando o utilizador finalizar de editar o documento clica em guardar e este é notificado de que o documento foi guardado e que se encontra a ser submetido para o escolinhas. Quando o envio para o servidor terminar o utilizador é notificado do seu sucesso.

Um cenário alternativo contempla o caso em que a aplicação não se encontra com ligação à internet, sendo neste caso o utilizador notificado, aquando do documento guardado, de que o envio deste se encontra pendente. Quando a aplicação voltar a ter conecção à internet o utilizador é notificado dos trabalhos que se encontram a ser enviados e mais uma vez notificado quando estes forem enviados com sucesso para o servidor.

Outro cenário alternativo possibilita ao utilizador, após guardar o documento, continuar a sua edição. Desta forma o utilizador nas vezes seguintes que clica em guardar é notificado de inicialmente de que o seu trabalho se encontra a ser actualizado no servidor, e finalmente é notificado de que o seu trabalho foi actualizado aquando do fim do envio deste.

Como requisito de funcionamento deste caso de utilização o utilizador necessita de se encontra autenticado na plataforma assim como ter instalado na máquina local as aplicações do Microsoft Office referidas.

5.2.3 Edição de trabalhos do tipo Microsoft Office

O utilizador dentro da interface Web do escolinhas aquando da visualização de um trabalho do tipo Microsoft Office clica no botão de edição. Quando clica em editar abre a aplicação do Microsoft Office respectiva numa nova janela. O utilizador após efectuar a edição

do documento clica em guardar e é notificado de que o documento se encontra a ser enviado para o servidor. Quando o envio é concluído o utilizador é novamente notificado.

Alternativamente o utilizador pode alterar o documento após guardar, clicando novamente em guardar, sendo posteriormente notificado do inicio e final da operação de envio para o servidor.

A edição do documento poderá ser efectuada sem existir conecção à internet, ficando a submição para o servidor pendente quando o utilizador clica em guardar. O utilizador é notificado de que o documento irá ser enviado para o servidor quando existir novamente conecção à rede, sendo novamente notificado quando tal acontecer.

Como requisito deste caso de utilização o utilizador necessitou de se ter autenticado previamente na aplicação. Também é necessário ter instalado na sua máquina a aplicação Microsoft Office correspondente ao documento a editar.

5.2.4 Submissão de vários ficheiros em simultâneo e em segundo plano para o escolinhas por drag and drop

O utilizador arrasta para a aplicação Escolinhas Desktop um conjunto de ficheiros de forma simultânea. Seguidamente recebe uma notificação da aplicação de que os respectivos ficheiros se encontram a ser submetidos para o servidor. A partir do momento que o utilizador larga os ficheiros sobre a aplicação, pode utilizar todas as funcionalidades do escolinhas.pt e Escolinhas Desktop sem necessitar de aguardar pela conclusão da submição destes, ocorrendo o envio dos vários ficheiros em segundo plano de forma transparente para o utilizador. Quando o envio de um ficheiro estiver concluído a aplicação lança uma notificação avisando o utilizador de que a submição do respectivo ficheiro foi concluída com exito.

É necessário que o utilizador se encontre autenticado na aplicação para permitir submeter ficheiros.

5.2.5 Acesso à aplicação em offline

A aplicação quando em determinadas situações ficar sem ligação à internet, notifica o utilizador disso, e deve poder funcionar, dentro do possível, da mesma forma como se o utilizador se encontra-se em modo online. A aplicação quando passa para modo offline notifica o utilizador do mesmo, mas permite a este aceder à aplicação da mesma forma como em online. As funcionalidades que funcionam de forma transparente para o utilizador em modo offline encontram-se descritas nos casos de utilização anteriores.

O utilizador necessita de previamente já se ter autenticado na aplicação quando em online, podendo esta autenticação ter sido realizada numa execução prévia da aplicação.

5.2.6 Acesso a toda a interface Web

O utilizador através do Browser integrado no Escolinhas Desktop permite ter acesso a toda a interface Web da plataforma escolinhas.pt. O utilizador abre a aplicação Escolinhas Desktop e a interface é em tudo semelhante à interface da aplicação Web escolinhas.pt. O utilizador pode efectuar todas as acções possíveis na plataforma da mesma forma como se acede-se de um Web Browser normal.

5.3 Requisitos suplementares

Do ponto de vista de requisitos suplementares a aplicação necessita de possibilitar a sua expansibilidade de novas funcionalidades Desktop assim como permitir de forma praticamente imediata a adopção de novas funcionalidades implementadas na plataforma Web.

A aplicação deve ter uma interface rica em usabilidade, assim como o mais igual possível à interface da aplicação escolinhas.pt, pois dedica-se a utilizadores familiarizados com a plataforma.

É necessário garantir o minimo de alterações necessárias no servidor, passando a implementação o mais possível pelo lado do cliente.

5.4 Arquitectura Lógica

O digrama na Figura 19 mostra a arquitectura lógica da aplicação Escolinhas Desktop.

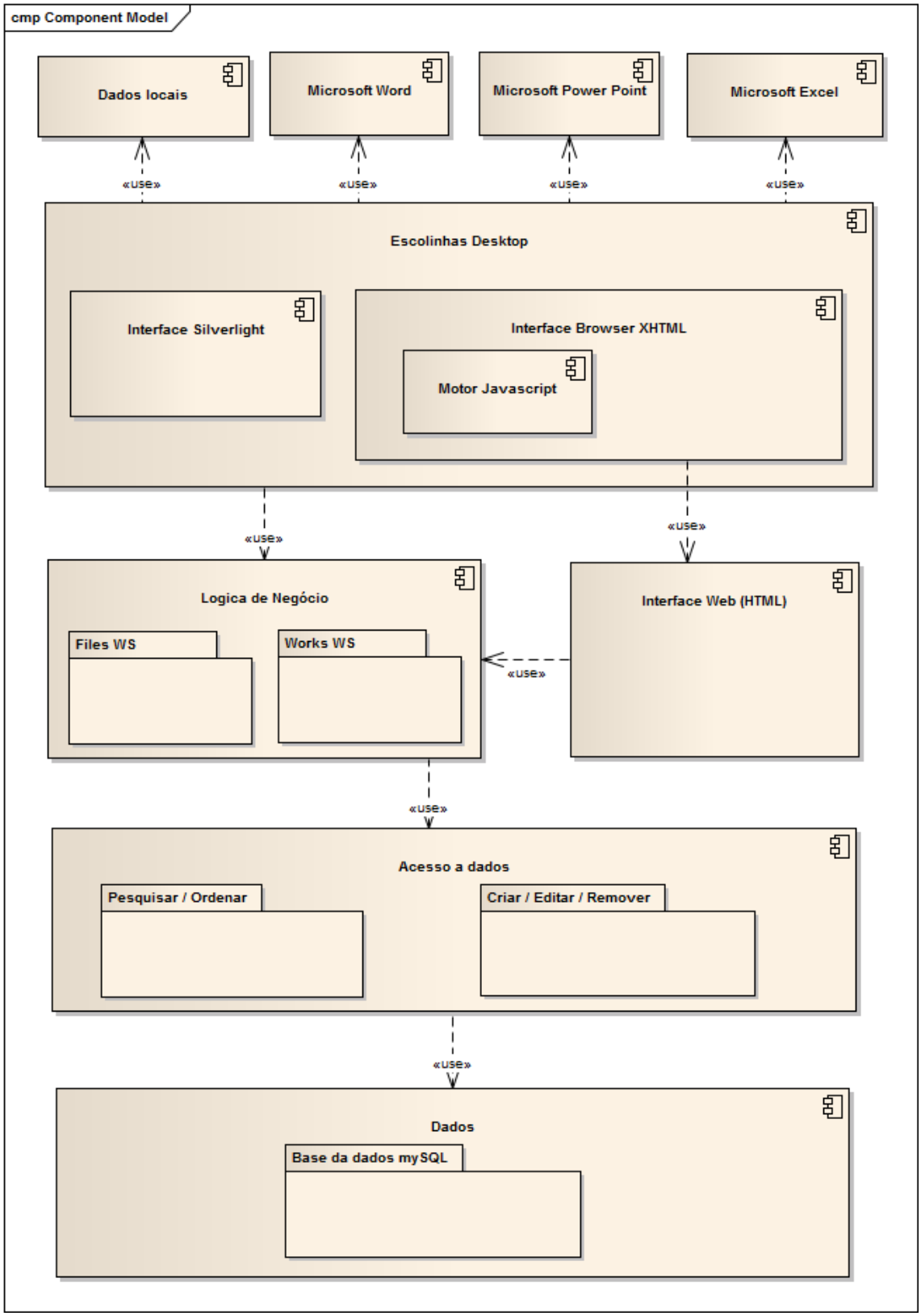

Figura 19: Arquitectura lógica da aplicação Escolinhas Desktop

Como é visível na Figura 19, a aplicação Desktop encontra-se com a interface separada em duas partes, a componente HTML e a componente Silverlight. A interface Silverlight para este protótipo apenas é constituída pelas janelas de notificação ao utilizador, como visível na Figura 27, e o rectângulo de Drop de ficheiros visível na Figura 24. A comunicação entre a interface Silverlight e a aplicação é efectuada utilizando o paradigma de programação orientada a eventos. A comunicação entre a aplicação e a interface Web existente no Browser XHTML por razões de limitação tecnológica necessita de ser efectuada recorrendo ao motor de Javascript existente no Browser. A comunicação entre estes dois componentes, apesar de serem parte da mesma aplicação, não é de todo facilitada, comunicando recorrendo unicamente a uma "String" com os dados. Os diagrama da Figura 20 mostra a sequencia da aplicação para eventos gerados na interface Web, e o diagrama da Figura 21 para a comunicação entre a aplicação Silverlight e a aplicação Desktop.

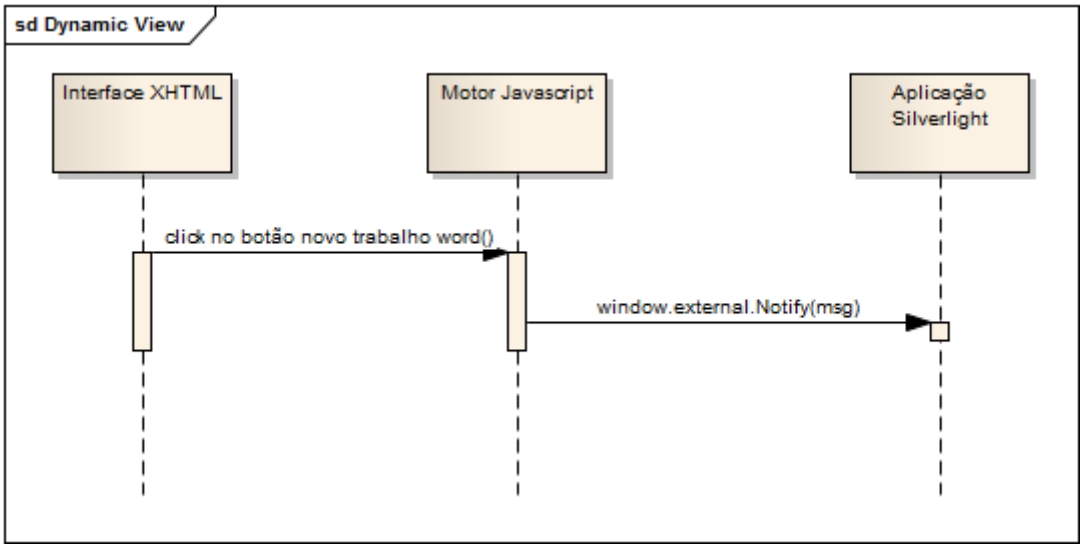

Figura 20: Exemplo de sequência de notificação da aplicação por parte de um evento na interface Web

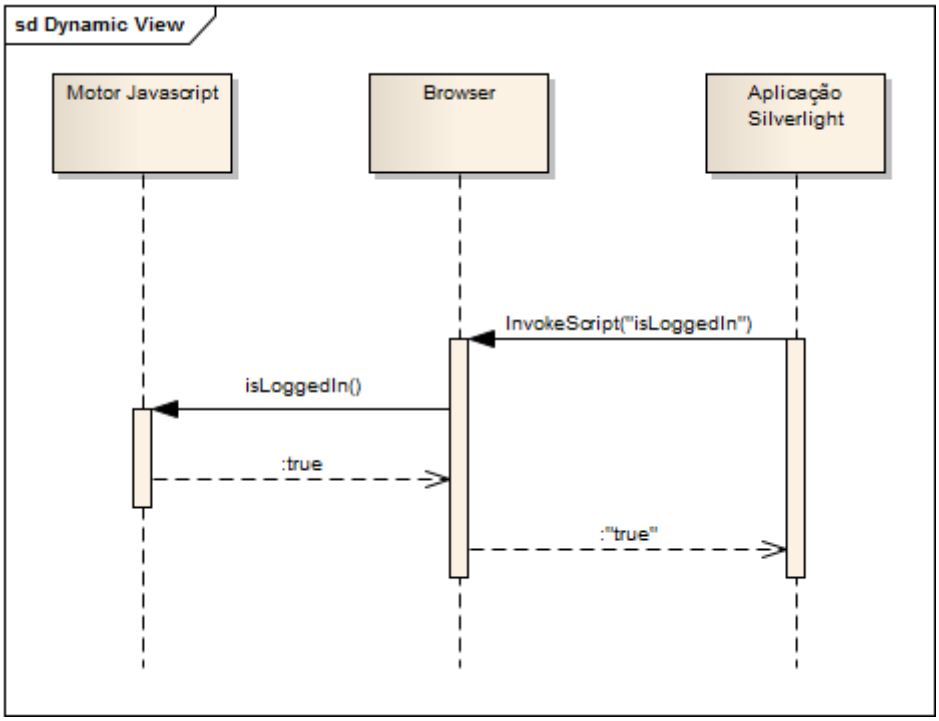

Figura 21: Comunicação entre a aplicação e o motor de Javascript existente no Browser integrado representado sobre a a forma de diagrama de sequencia

Apesar de possível, a comunicação entre a interface Web e a aplicação Silverlight não é muito facilitada. A única forma de comunicação contempla a passagem de informação tanto de um lado como para outro usando uma "String" como transmissão de dados. O método "InvokeScript" disponível no Browser do Silverlight apenas permite invocar métodos com atributos do tipo "String". Para contornar esta limitação pode-se utilizar a função de Javascript "eval" que interpreta a String recebida como se de código Javascript se tratasse. Este método utiliza assim a capacidade de reflexão[19] da linguagem Javascript.

5.5 Arquitectura Tecnológica

A Figura 22 tem o intuito de mostrar em forma de diagrama UML de componentes a arquitectura tecnológica da aplicação.

19 Capacidade que um programa tem de analisar e alterar a sua estrutura. Umas das funcionalidades possíveis em linguagem que suportem reflexão é a capacidade de analisar uma cadeia de caracteres (String), como se esta foce código fonte, em tempo de execução.

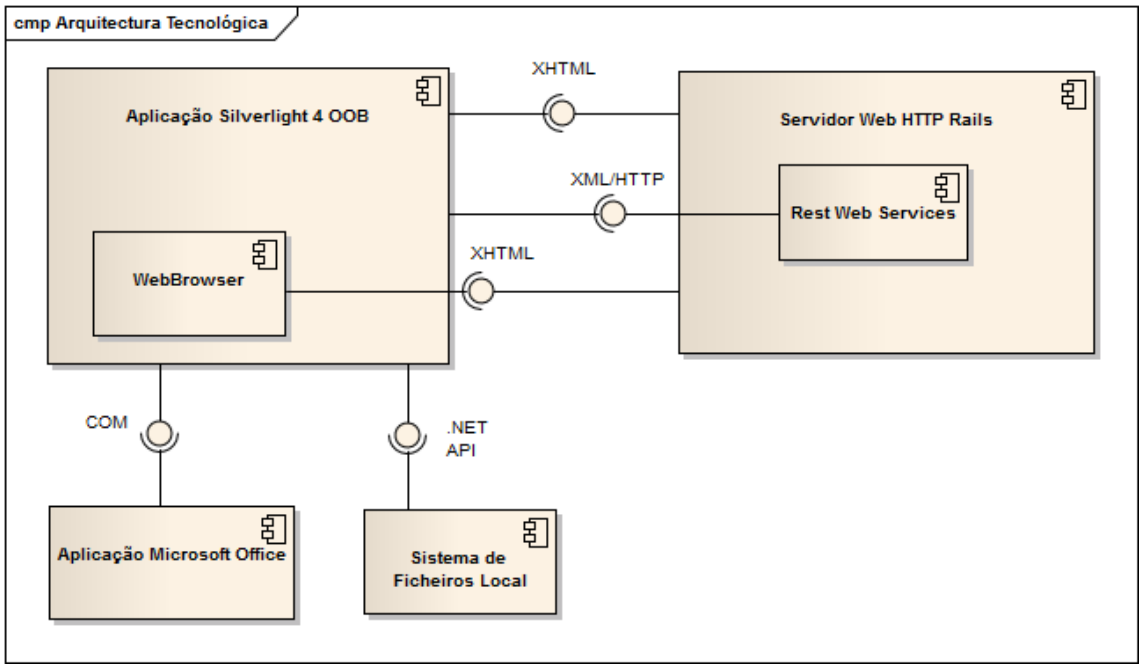

Figura 22: Arquitectura Tecnológica do Escolinhas Desktop

A aplicação Escolinhas Desktop encontra-se implementada em Silverlight 4, sendo apenas possível a sua execução após a instalação na máquina do cliente.

O componente principal de interface da aplicação consiste em um Web Browser, tendo este a tarefa de apresentar a interface da aplicação escolinhas.pt. O componente WebBrowser em Silverlight é na realidade uma janela do Internet Explorer se a aplicação estiver a correr num sistema operativo Windows ou então uma janela de Safari, caso estejamos a correr a aplicação em Macintosh da Apple. O facto de o Web Browser diferir de plataforma implica com que haja certas diferenças visuais da interface Web, como pode ser comprovado entre a Figura 24 e a Figura 27. Desta forma temos em ambos os sistemas operativos compatíveis com o Silverlight 4 da Microsoft um Browser com capacidade para executar todas as funcionalidades da plataforma escolinhas.pt.

O Web Browser apresenta código XHTML que obtém directamente do servidor do escolinhas.pt, assim como utiliza o motor de Javascript para interpretar este. As acções executadas sobre a interface Web são processadas da mesma forma que em qualquer Web Browser moderno como o seguimento de hiperligações com o clique, alterações ao DOM por parte do Javascript, etc.

A aplicação Silverlight para persistir a parte Web da interface comunica de igual forma como o Browser com o servidor e armazena no sistema de ficheiros local as páginas XHTML, assim como os ficheiros de código Javascript, folhas de estilo HTML, imagens e todo o tipo de objectos contidos na página respectiva. Desta forma quando em offline a aplicação em Silverlight “alimenta” o Browser com as páginas XHTML existentes no sistema de ficheiros local.

Para a submição de ficheiros ou trabalhos para o escolinhas.pt utiliza-se como forma de comunicação o protocolo HTTP simples, sendo necessário compor todo o corpo do pedido

HTTP. Para a comunicação com Web Services do tipo REST o Silverlight apenas dispõe de comunicação a baixo nível, mais concretamente ao nível do protocolo HTTP.

A comunicação com as aplicações do Microsoft Office utilizam o mecanismo de Component Object Model (COM). Com esta tecnologia que se encontra disponível pela primeira vez na versão 4 do Silverlight e unicamente nas versões fora do navegador, o Escolinhas Desktop permite comunicar e interagir com aplicações instaladas na máquina do cliente que utilizem este paradigma para abrirem a sua API. Em COM a interacção com outras aplicações faz-se utilizando uma API totalmente orientada a objectos.

A aplicação Escolinhas Desktop para persistir os dados utiliza uma API de acesso ao sistema de ficheiros igual à existente em .NET 4.

5.6 Arquitectura Física

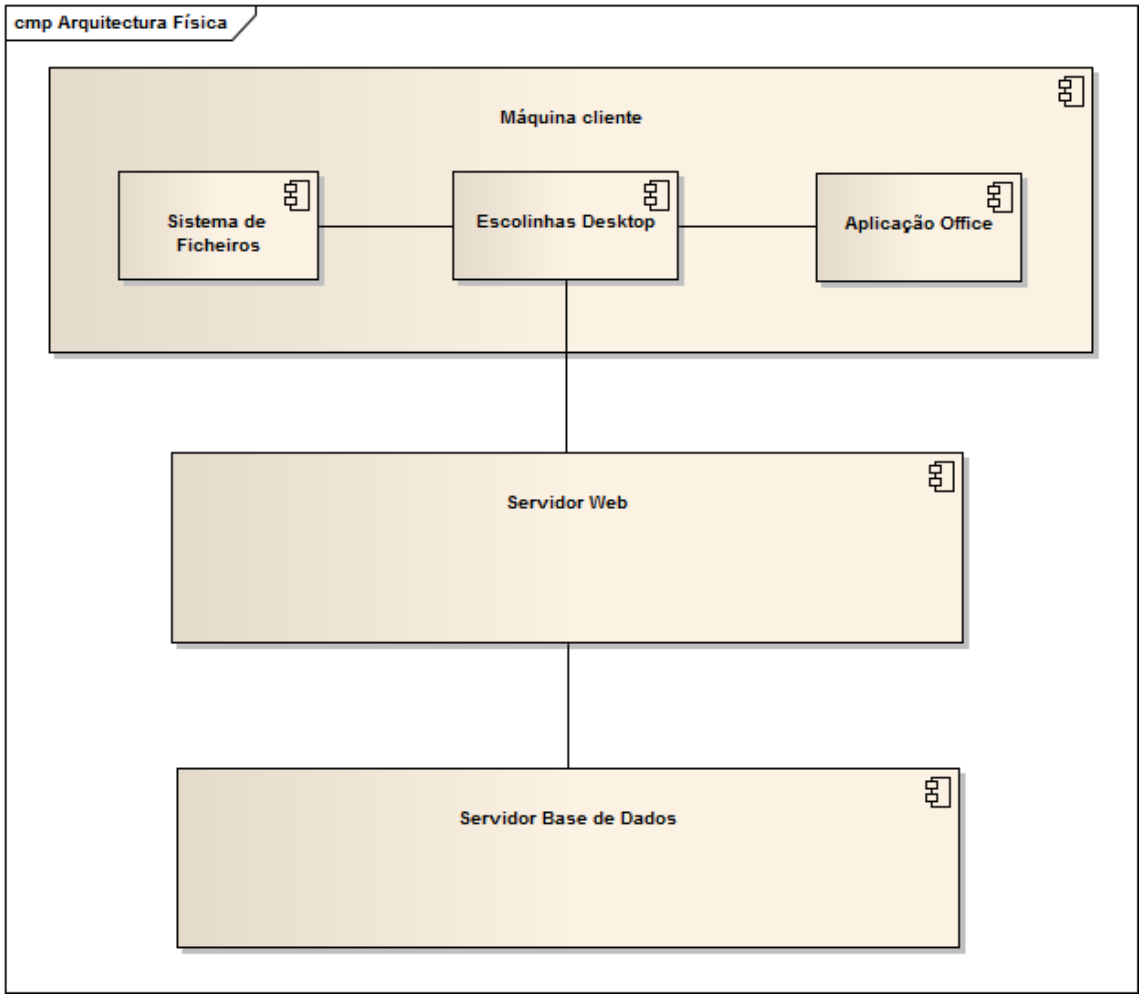

Figura 23: Arquitectura Física do Escolinhas Desktop

5.6.1 Máquina cliente

A máquina do cliente é um terminal físico que necessita de ser compatível em Hardware e Sistema Operativo com a versão 4 do Microsoft Silverlight. Actualmente existe a restrição de que o Sistema Operativo necessita de ser ou Microsoft Windows ou Macintosh, existindo progressos no suporte para Linux, estando o Silverlight a ser migrado para esta plataforma não pela Microsoft mas sim pela Novel.

Para ter acesso a todas as funcionalidade do escolinhas é necessário que o utilizador tenha instalado na sua máquina o Microsoft Word, Excel e Power Point de forma a poder realizar trabalhos usando estes editores.

5.6.2 Máquina Servidor Web

O servidor Web é actualmente um Virtual Private Server (VPS), que do ponto de vista deste diagrama de arquitectura é em tudo semelhante a um servidor físico. Este servidor aloja a aplicação Web implementada em Ruby On Rails (ROR), correndo uma distribuição de Linux como Sistema Operativo.

5.6.3 Máquina Servidor Base de Dados

A base de dados encontra-se num servidor ligado em rede com a VPS da aplicação Web. Esta máquina aloja a base de dados em mySql.

5.7 Protótipo

Nesta secção são apresentadas as interfaces correspondentes ao casos de utilização enumerados em 5.2, assim como as principais decisões de desenho relativas a esse caso de utilização. Por ultimo são enunciadas algumas decisões de desenho relativas a todo o projecto.

5.7.1 Submissão offline de ficheiros para o escolinhas

5.7.1.1 Interface

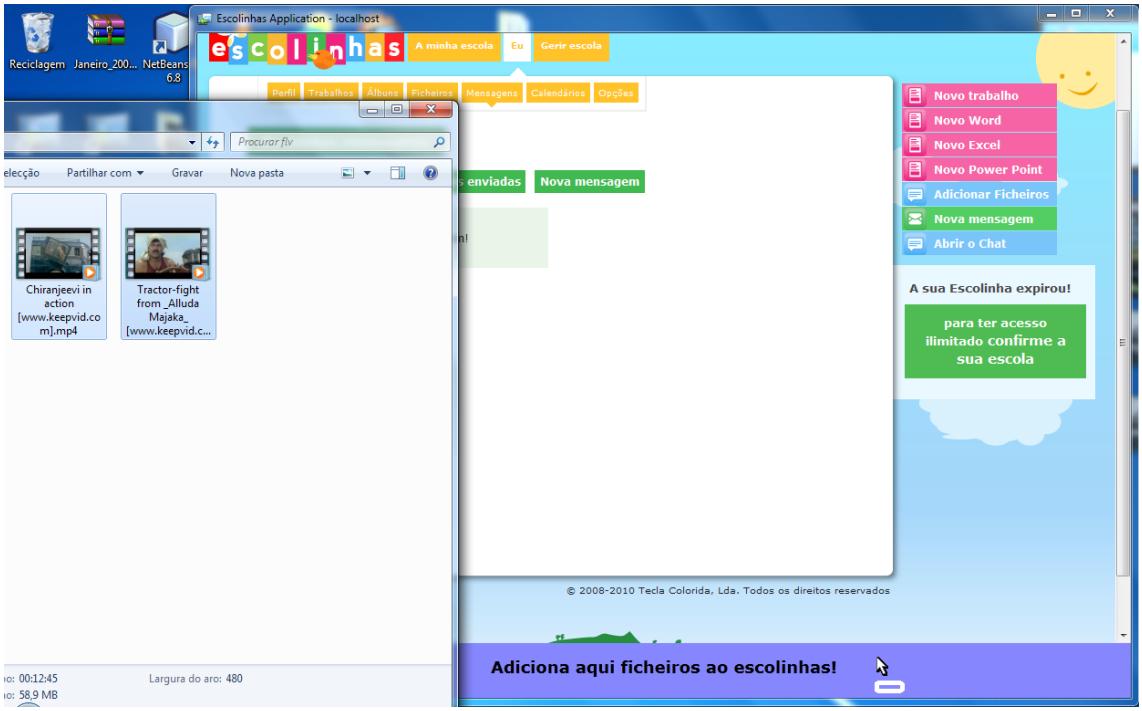

Figura 24: Ecrã de Escolinhas Desktop: Drag and Drop de ficheiros

5.7.1.2 Decisões de desenho

Relativamente à submição de ficheiros do escolinhas.pt, de maneira a ser possível submeter em modo offline foi decidido criar um Thread trabalhador, que de 10 em 10 segundos analisa o estado de conectividade da aplicação, e caso já exista ligação à rede é efectuado o respectivo envio dos ficheiros pendentes.

Uma das limitações por questões de segurança que o Silverlight apresenta, prende-se com o facto de que apenas é possível aceder ao sistema de ficheiros do directório correspondente à pasta “Meus Documentos” ou equivalente em sistemas Macintosh. Quando é efectuado o Drag and Drop para o Escolinhas Desktop apenas está acessível o nome do ficheiro assim como o seu conteúdo, não sendo possível ter o caminho completo no sistema de ficheiros para aceder a esse documento ou anexo. A informação e conteúdo do ficheiro largado sobre a aplicação apenas se encontra disponível durante aquela execução, no objecto enviado pelo evento de Drop.

Por esta razão quando o utilizador efectua o Drag and Drop de um ficheiro para o Escolinhas Desktop, é necessário armazenar o ficheiro numa pasta temporária criada dentro dos meus documentos, para posterior envio, tanto nesta como numa execução futura da aplicação Escolinhas Desktop.

5.7.2 Criação de trabalhos utilizando ferramentas do Microsoft Office

5.7.2.1 Interface

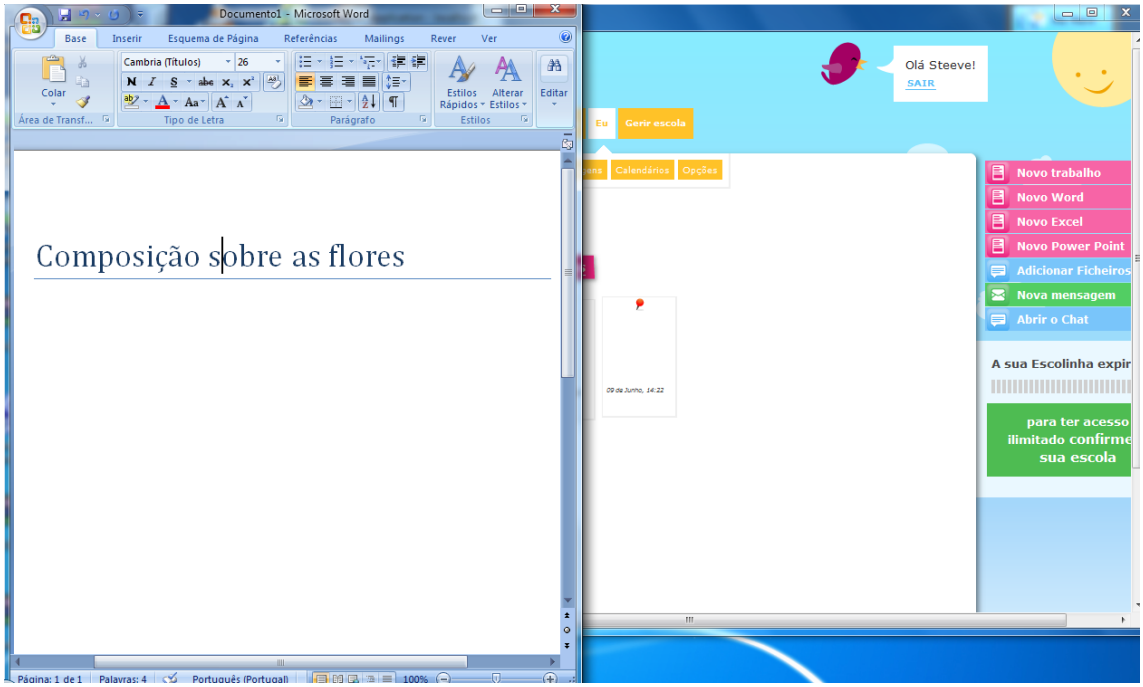

Figura 25: Ecrã Escolinhas Desktop: Criação de trabalho usando Microsoft Word

5.7.2.2 Decisões de desenho

Para criar trabalhos do tipo Word, Excel ou Power Point foram adicionados botões respectivos na interface Web. O mecanismo usado foi o descrito no ponto 4.2.4, de expansibilidade da aplicação Web mas unicamente visível para a aplicação Desktop.

Quando é criado um novo trabalho este é armazenado numa pasta temporária dentro do directório “Os meus documentos” em Windows ou na pasta equivalente em Macintosh, de forma a ser possível ler o ficheiro e envia-lo para o servidor.

5.7.3 Edição de trabalhos do tipo Microsoft Office

5.7.3.1 Interface

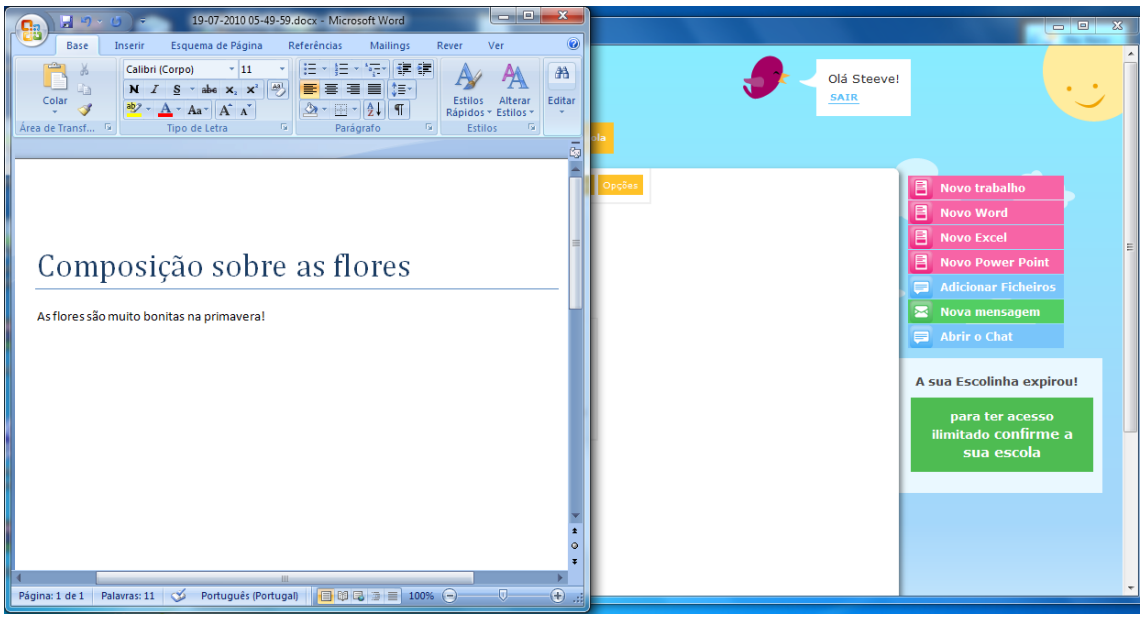

Figura 26: Edição de um trabalho no escolinhas do tipo Word

5.7.3.2 Decisões de desenho

Do ponto de vista de implementação foi decidido guardar localmente os trabalhos que o aluno consultou em modo online, para permitir uma posterior edição em offline.

A interface foi expandida de acordo com o referido em 4.2.4, aquando da visualização de um trabalho do tipo Office, em que o botão existente agora, de edição do trabalho, notifica a aplicação, que por sua vez abre a ferramenta de Office responsável pela edição daquele formato.

A edição de documentos do tipo Microsoft Office não é possível de ser feita na aplicação escolinhas.pt acedendo por um Web Browser. Apenas é possível a edição no Escolinhas Desktop.

5.7.4 Submissão de vários ficheiros em simultâneo e em segundo plano para o escolinhas por drag and drop

5.7.4.1 Interface

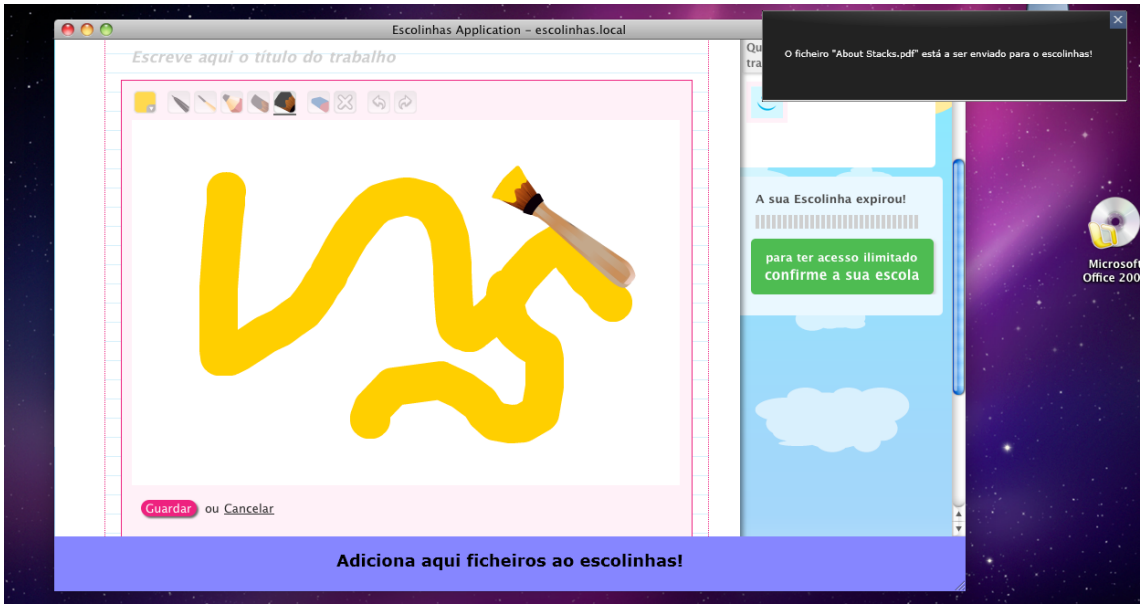

Figura 27: Submissão de um ficheiro em segundo plano enquanto se pinta no escolinhas

5.7.4.2 Decisões de desenho

A API de Drag and Drop de ficheiros do Silverlight permite nativamente gerir o largar de vários ficheiros sobre a aplicação. Posteriormente o envio dos ficheiros para o servidor Web contempla o envio desta informação usando o protocolo HTTP, recorrendo à classe HTTPWebRequest. Esta classe tal como todas as outras de Silverlight relativas à comunicação, efectuam o seu trabalho de forma assíncrona para o Thread que a chama, logo correndo em segundo plano, possibilitando o acesso às funcionalidades da aplicação enquanto a submição se encontra a decorrer.

5.7.5 Acesso à aplicação em offline

5.7.5.1 Interface

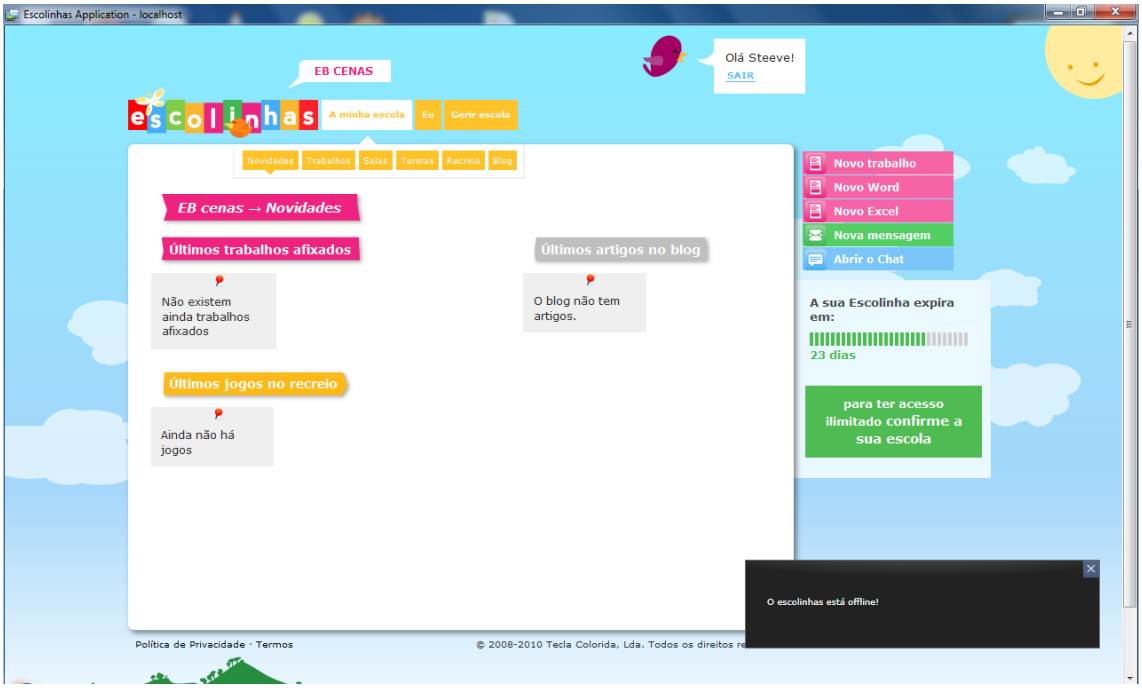

Figura 28: Notificação de que o Escolinhas Desktop entrou em modo offline

5.7.5.2 Decisões de desenho

Do ponto de vista da persistência necessária ao armazenamento de ficheiros necessários para os casos de utilização já referidos em 5.7.1, 5.7.2 e 5.7.3, devido ao facto de que o Silverlight não se apresentar para já com um motor de Base de Dados, foi utilizado XML para armazenar a estrutura necessária ao armazenamento de informação sobre os ficheiros. Como exemplo para um documento do tipo Office é armazenado um nó no documento XML para cada documento criado. Cada nó contêm o identificador na base de dados do documento ou "-1" caso este seja um documento ainda não submetido para o servidor, o caminho para o ficheiro, e o tipo de documento (Word, Excel ou Power Point). Foi escolhido armazenar os dados utilizando a estrutura XML pela facilidade de criação e manipulação dos dados de um ficheiro XML que o Silverlight apresenta.

Para além dos trabalhos é necessário persistir a interface Web. Mais uma vez foi utilizado XML para armazenar a informação, como a localização no sistema de ficheiros e a que URI da aplicação Web se referem as páginas HTML armazenadas. Para o Browser Web integrado na aplicação seja compatível com a página HTML armazenada, esta não pode conter referencias para ficheiros de código Javascript (excepto código *inline* com o HTML"), assim como não pode ter referência para imagens, mais uma vez excepto estas não se encontrem definidas no URI usando o *data URI Shema*[20]. Uma das maiores limitações desta forma de incluir imagens

20 Forma de armazenar dados, como imagens, ficheiros de código Javascript, folhas de estilo CSS, etc. em linha com o código HTML.

num documento HTML prende-se pelo facto de que o tamanho máximo de um ficheiro armazenado desta forma é de 32 KB em Internet Explorer, não existindo limite em Safari. Como em Windows o navegador integrado utiliza o Internet Explorer como Browser estamos desta forma limitados a ficheiros com menos de 32KB. Mesmo com esta restrição é possível apresentar toda a interface Web do escolinhas.pt em modo offline. As páginas Web visíveis para o utilizador em modo offline corresponde às páginas que ele visitou durante as ultimas vezes que navegou na aplicação em modo online.

5.7.6 Acesso a toda a interface Web

5.7.6.1 Interface

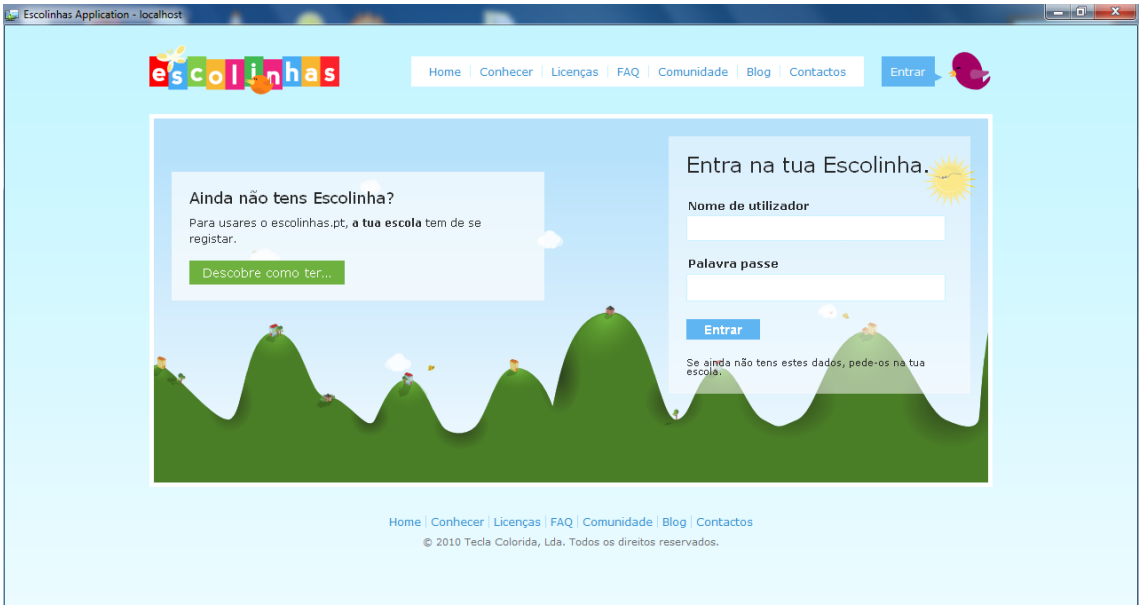

Figura 29: Formulário de autenticação do Escolinhas Desktop

5.7.6.2 Decisões de desenho

Para permitir que seja possível o acesso a todas as funcionalidades da plataforma escolinhas na versão Desktop, apenas é necessário usar o navegador integrado no Silverlight.

5.7.7 Outras decisões de desenho

5.7.7.1 Autenticação

A autenticação na aplicação, como na chamada a Web Services REST faz-se recorrendo à autenticação na aplicação pelo método normal da aplicação Web. Desta forma não é necessário adicionar Web Services de autenticação à aplicação Web. As chamadas HTTP de mais baixo nível em Silverlight têm a possibilidade de serem chamadas utilizando o mesmo cliente que o do navegador integrado. Desta forma a autenticação no Browser integrado pelo utilizador abre em simultâneo o acesso aos Web Services da aplicação.

5.7.7.2 Imagens na interface Web

Como já explicado anteriormente existe a limitação de utilizar imagens que ocupem menos que 32 KB, pelo menos em Windows, visto que em Macintosh esta restrição não se aplica. Caso a imagem seja maior que 32KB, esta simplesmente esta aparece. Para resolver a situação decidiu-se usar um método de encriptação com percas de forma a converter uma imagem com dimensões superiores a 32KB para 32KB. O formato adoptado foi o JPEG.

O nosso objectivo é conseguir uma imagem com tamanho inferir a 32 KB, mas no tipo de compressão JPEG não é muito fácil calcular o valor de compressão necessário para obter esse resultado. Desta forma, para resolver este problema, é utilizado inicialmente uma taxa de compressão de 90. O processo seguinte é iterativo onde a taxa de compressão é reduzida em 90% em cada iteração comparando seguidamente com o valor do tamanho da imagem com aquela respectiva taxa de compressão.. Quando a imagem comprimida se encontrar dentro dos parâmetros necessários para a iteração, e é escolhido aquele valor como taxa de compressão para aquela imagem.

5.7.7.3 Detecção de modo offline

Em Silverlight a detecção de que nos encontramos em modo offline não é feita pela aplicação. Ao que realmente temos acesso no Silverlight 4 é se existe ou não uma conectividade a uma alguma rede, podendo ser unicamente local.

Para detectar a perca de ligação à internet por parte da aplicação foi necessário adicionar um Thread que monitoriza-se o estado da ligação. Para o fazer este Thread efectua pedidos periódicos ao sitio da Web google.com, lançando um evento quando o pedido de comunicação não for realizado com sucesso, significando uma perca de ligação à internet.

5.8 Dificuldades

A principal dificuldade encontrada na implementação deste projecto encontrou-se nas limitações ou inadequabilidade das tecnologias actuais para a implementação de uma RIA

desktop do tipo Browser dedicado, nomeadamente as limitações existentes do Browser Web integrado. Estas limitações foram abordadas no ponto 4.2. Outra dificuldade encontrada prende-se com o facto de que a tecnologia que mais se adequava ao projecto, Microsoft Silverlight, se encontrar no inicio desta dissertação em versão *Beta*, passando pelas versões *Release Candidate* e por ultimo a versão final do Silverlight 4. Estas alterações implicaram analisar as diferenças entre as versões da tecnologia de forma a compreender as novas funcionalidades ou possíveis novas limitações da versão deste tecnologia, para além de ser necessário a actualização de todas as ferramentas necessárias ao seu desenvolvimento, e inclusive a actualização do sistema operativo de desenvolvimento. Uma das limitações existentes na RIA desenvolvida prende-se com o facto de que em Silverlight não é possível cancelar a acção de retroceder ou avançar do Browser, causando o aparecimento da página com o erro Http 404 no Browser quando se efectua a ação de retroceder em offline, ficando a aplicação sem uma interface utilizável. Mais uma vez nota-se que o Browser integrado não foi pensado para servir de suporte a este tipo de utilização, o que faz com que a solução encontrada apresente algumas limitações.

5.9 Conclusões

Actualmente as tecnologias de RIA existentes não se encontram particularmente orientadas ao desenvolvimento de RIAs desktop do tipo Browser dedicado. Com as vantagens anunciadas deste tipo de aplicação Desktop relativamente a uma RIA do tipo cliente de desktop espera-se que surjam mais exemplos deste tipo de aplicação para além do iTunes ou deste projecto que influenciem a tecnologia a evoluir no sentido de permitir de uma forma mais simples e com menos limitações implementar RIAs do tipo Browser dedicado.

6 Conclusões

De acordo com o trabalho efectuado, chegamos à conclusão que actualmente as RIAs desktop mais comuns contam actualmente com alguns exemplos, por vezes promocionais de tecnologia, mas que apresentam pouca adopção por parte do público. As causas apontadas para isso sugerem-se que sejam:

- Interfaces diferentes entre a versão Desktop e Web;
- Apesar de permitir interfaces mais ricas por vezes retira algumas funcionalidades importantes da aplicação (e.g. Incapacidade de correr as aplicação do Facebook na versão desktop implementada em Silverlight);
- Principalmente com a desvantagem anterior esta não se pode tornar um substituto da aplicação Web, e os utilizadores por conveniência preferem apenas continuar a utilizar a aplicação Web.

Um exemplo de RIA muito bem sucedida e das primeiras existentes, o iTunes, não apresenta as desvantagens enunciadas pelo facto de conter dentro da aplicação desktop um Browser integrado, em que a interface da aplicação passa em parte pela interface Web. Esta extensão da aplicação Web com a aplicação Desktop elimina as questões de interfaces diferentes nas funcionalidades Web da aplicação, e o facto de esta permitir ser em todas as situações um substituto da aplicação Web.

Do ponto de vista tecnológico, nomeadamente no que a ferramentas mais recentes de desenvolvimento de RIAs Desktop diz respeito, estas apesar de permitirem ter um Browser Web integrado, conclui-se que este é limitativo, e acarreta problemas que necessitam de soluções astutas para os contornar. Utilizando outro tipo de tecnologia como motores de Browser open source é possível implementar sem as restrições encontradas, pesando neste caso o custo de desenvolvimento, mas aumentando a viabilidade.

O projecto final vai de encontro ao especificado, efectuado em Silverlight Pelas mesmas questões tecnológicas a solução final apresenta-se com ligeiras limitações.

Por fim, acredito que as limitações actuais da tecnologia têm tendencia para desaparecer. Possivelmente nas próximas versões das principais tecnologias já seja possível desenvolver uma RIA com um Browser dedicado, com as limitações enunciadas retiradas e com consequente desenvolvimento mais facilitado. A primeira tecnologia que surgiu que possibilitava a criação de RIAs para Desktop foi o Adobe AIR à pouco mais de dois anos, o que justifica de certo modo as limitações encontradas.

O trabalho futuro mais imediato passa pelo teste da aplicação em ambiente Macintosh, e implementação das alterações necessárias para que nesta plataforma o Escolinhas Desktop corra de forma estável. A colocação deste projecto em produção, após garantia de funcionamento estável da aplicação, é de algum interesse para a empresa.

Referências

[ACID2] Acid2 Browser Test, http://www.webstandards.org/action/acid2/, Acesso: 2010-06-20

[AIRGAL] open source framework | Adobe Flash Platform runtime gallery, http://www.adobe.com/products/flashplatformruntimes/gallery/?promoid=GRLQA, Acesso: 2010.06-21

[AJAXLIB] Top JavaScript and AJAX Libraries, http://www.c-point.com/javascript_and_ajax_libraries.php, Acesso: 2010-01-29

[AJXPTR] Mahemoff, Michael, Ajax Design patterns: Creating Web 2.0 Sites withProgramming and Usability Patterns, USA, O'Reilly, 2006, ISBN: 0-596-10180-5

[ALL02] Allaire, Jeremy, Macromedia Flash MX - A next-generation rich client, http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf, Acesso: 2010-01-27

[AMYA] Amaya Home Page, http://www.w3.org/Amaya/, Acesso: 2010-06-20

[ARN94] Arena (web browser), http://en.wikipedia.org/wiki/Arena_%28web_browser%29, Acesso: 2010-06-20

[COM] Componenet Object Modelo, http://en.wikipedia.org/wiki/Component_Object_Model, Acesso: 2010-06-14

[CSS] Cascading Style Sheets, http://en.wikipedia.org/wiki/Cascading_Style_Sheets, Acesso: 2010-06-20

[ERW92] Erwise, http://en.wikipedia.org/wiki/Erwise, Acesso: 2010-06-20

[ESC10] escolinhas.pt :: Plataforma Colaborativa e Social para o Ensino Básico, http://escolinhas.pt/, Acesso: 2010-06-28

[FBUG] Firebug, http://getfirebug.com/, Acesso: 2010-06-20

[FLSHSTAT] Adobe - Flash Player Statistcs, http://www.adobe.com/products/player_census/flashplayer/, Acesso: 2010-06-17

[FREEONL] Free Online Software For The Office - A List of Free Online Software, http://webtrends.about.com/od/enterprise20/a/online-software.htm, Acesso: 2010-01-29

[GCHRM] Google Chrome, http://www.google.com/chrome, Acesso: 2010-06-20

[H5WIKI] HTML5, http://en.wikipedia.org/wiki/HTML5, Acesso: 2010-06-12

[HTML5] HTML5, http://dev.w3.org/html5/spec/Overview.html, Acesso: 2010-06-12

Referências

[JESJAM] Garret, Jesse James, Ajax: A New Approach to Web Applications, http://www.adaptivepath.com/ideas/essays/archives/000385.php, Acesso: 2010-06-17

[LIC09] Mano, Licinio, Novos padrões de interacção com o utilizador em aplicações Web usando Ajax, Faculdade de Engenharia da Universidade do Porto, Porto, 2009.

[Moritz] Moritz, Florian, Rich Internet Applications (RIA): A Convergence of User InterfaceParadigms of Web and Desktop Exemplified by JavaFX, University of AppliedScience Kaiserslautern, Deutschland, 2008

[MOS93] Mosaic, http://en.wikipedia.org/wiki/Mosaic_(web_browser), Acesso: 2010-06-20

[NCSA10] National Center for Supercomputing Applications at the University of Illinois, http://www.ncsa.illinois.edu/, Acesso: 2010-06-20

[NETSCP] Netscape Navigator, http://en.wikipedia.org/wiki/Netscape_Navigator, Acesso: 2010-06-20

[NEX91] WorldWideWeb, http://en.wikipedia.org/wiki/WorldWideWeb, Acesso: 2010-06-20

[OPERA] Opera (web browser), http://en.wikipedia.org/wiki/Opera_%28web_browser%29, Acesso: 2010-06-20

[OREILLY] O'Reilly, Tim, What is Web 2.0: Design Patterns and Business Models for the Next Generation of Software, USA, O'Reilly Media, http://papers.ssrn.com/sol3/papers.cfm?abstract_id=1008839 , Acesso: 2010-06-15

[PEAKEE] Peacekeeper - The Browser Benchmark from Futuremark Corporation, http://service.futuremark.com/peacekeeper/index.action, Acesso: 2010-06-20

[PLUGIN] Plug-in (computing), http://en.wikipedia.org/wiki/Plug-in_%28computing%29, Acesso: 2010-06-17

[REST] Representational State Transfer, http://en.wikipedia.org/wiki/Representational_State_Transfer, Acesso: 2010-06-23

[RIAF] A comparison of Rich Internet Applications (RIA) development frameworks, http://www.quesucede.com/page/show/id/ria_frameworks, Acesso: 2010-01-28

[RIASTAT] Rich Internet Application Statistics, http://www.riastats.com/, Acesso: 2010-01-28

[RVIDEO] RealVideo, http://en.wikipedia.org/wiki/RealVideo, Acesso: 2010-06-23

[SAFARI] Apple - Safari - Browse the web in smarter, more powerful ways., http://www.apple.com/safari/, Acesso: 2010-06-20

[SERVSOCK] Silverlight Server Socket: The Official Microsoft Silverlight Site, http://forums.silverlight.net/forums/p/177215/402690.aspx, Acesso: 2010-04-27

[SLVRLDSK] Silverlight on the desktop, http://agkdesign.wordpress.com/2009/03/20/silverlight-on-the-desktop/, Acesso: 2010-01-29

[SOAP] SOAP, http://en.wikipedia.org/wiki/SOAP_%28protocol%29, Acesso: 2010-06-23

[SWARTA] Swartz, Aaron, A Brief History of Ajax, http://www.aaronsw.com/weblog/ajaxhistory, Acesso: 2010-01-28.

[TUCDAV08] Tucker, David, AIR API - Introduction to the SQLite Database, http://www.insideria.com/2008/03/air-api-introduction-to-the-sq.html, Acesso: 2010-01-29

Referências

[URISCHM] data URI scheme, http://en.wikipedia.org/wiki/Data_URI_scheme, Acesso: 2010-06-24

[USRAGNT] User agent, http://en.wikipedia.org/wiki/User_agent, Acesso: 2010-06-24

[VISWGID] Desktop Migration to Web & Cloud, http://www.visualwebgui.com/tabid/664/Default.aspx, Acesso: 2010-01-27

[WCHRM] Google Chrome, http://en.wikipedia.org/wiki/Google_Chrome, Acesso: 2010-06-20

[WEB1] Web 1.0, http://en.wikipedia.org/wiki/Web_1.0, Acesso: 2010-06-20

[WEB2] Web 2.0, http://en.wikipedia.org/wiki/Web_2.0, Acesso: 2010-06-20

[WEB90] How It All Started: Pre-W3C Web and Internet Background, http://www.w3.org/2004/Talks/w3c10-HowItAllStarted/?n=15, Acesso: 2010-01-28

[WMVWIKI] Windows Media Video, http://en.wikipedia.org/wiki/Windows_Media_Video, Acesso: 2010-06-23

[WSERVC] Web service, http://en.wikipedia.org/wiki/Web_service, Acesso: 2010-06-23

[XREQ09] XMLHttpRequest, http://www.w3.org/TR/XMLHttpRequest/, Acesso: 2010-06-18