

Novos padrões de interacção com o utilizador em aplicações Web usando Ajax

Licínio Venâncio Feiteira Mano, Licenciado em Novas Tecnologias da Comunicação pela Universidade de Aveiro



Universidade do Porto

Faculdade de Engenharia

FEUP

Departamento de Engenharia Informática

Dissertação submetida para satisfação parcial dos requisitos do grau de mestre em Engenharia Informática (Área de especialização: Engenharia de Software)

Dissertação realizada sob a supervisão de
Professor Doutor Ademar Manuel Teixeira de Aguiar,
do Departamento de Engenharia Informática
da Faculdade de Engenharia da Universidade do Porto

Porto, Março 2009

Resumo

A Web e as tecnologias que através dela gravitam encontram-se em contínua evolução. Com o passar dos anos a utilização da Web e da informação que através dela é partilhada tem-se vindo a modificar. Se no passado se falava em simples páginas Web ou websites, hoje em dia falamos em aplicações complexas capazes de substituir ou, pelo menos, reduzir a dependência das aplicações típicas de Desktop.

Na base destas evoluções estão um conjunto de aspectos como a necessidade e novos requisitos lançados pelos utilizadores, bem como, um conjunto de inovações tecnológicas, entre as quais se encontra o Ajax. A tecnologia Ajax tem potenciado a transposição de modelos de interacção com o utilizador, típicos de aplicações Desktop que se encontram amplamente disseminados entre aplicações e utilizadores, para as aplicações Web.

Para a evolução verificada importa sistematizar os fenómenos propulsores de mudança, bem como, ponderar desafios e oportunidades inerentes. Nesse sentido, o presente trabalho tenta captar e comparar as principais tendências de convergência entre modelos de interacção disponíveis em aplicações Web e Desktop, bem como, identificar e sistematizar os referidos modelos de interacção através de Padrões.

O trabalho consiste por isso na identificação, descrição e análise de novos padrões e paradigmas de interacção disponíveis para aplicações Web baseadas em tecnologia Ajax, sendo o Ajax o fenómeno propulsor da crescente inovação presente nessas aplicações.

O trabalho conduz-nos também por uma análise detalhada sobre a evolução das interfaces com o utilizador presentes nas tradicionais aplicações Web e Desktop, indicando tendências de convergência expectáveis de acordo com as mais-valias e constrangimentos de base inerentes a cada plataforma.

Como propulsor de mudança será prestada especial atenção ao fenómeno Ajax. A arquitectura Ajax será descrita detalhadamente e, de acordo com os desafios e oportunidades inerentes ao seu processo de maturação e disseminação, será apresentada uma perspectiva sobre como esta tecnologia poderá evoluir e coexistir com outras tecnologias no contexto Web.

O presente trabalho conclui que a tecnologia Ajax, devido à sua proximidade e respeito pelos standards, bem como, pelo facto de se tratar de uma tecnologia aberta à comunidade de programadores, tenderá a disseminar-se cada vez mais pela plataforma Web, potenciando com isso o surgimento e transposição de novos modelos de interacção com o utilizador.

Conclui ainda que a convergência dos modelos de interacção entre ambas as plataformas – Web e Desktop – é um facto objectivo e não uma teoria, e aponta como cenário provável numa evolução natural, uma convergência cada vez maior, descartando no entanto a perspectiva de fusão completa entre ambas as plataformas.

Abstract

The World Wide Web and evolving technologies that gravitate around it, push evolution forward and forward. Trough times the Web has changing as well as the way users relate and depend on it. The time for text based websites as gone, nowadays users see the internet as a powerful tool and platform for deliver and share rich content like video, audio and live communication. So the WWW is on its way to close the gap between Web and the Desktop applications.

In the heart of this evolution you can find strong demanding from users for more interaction, more user experience on the Web applications, as well as, technological innovations on progress. Ajax is one of the major technologies that are pushing forward Web applications, with special impact on user experience. With Ajax rise, the convergence between Web and Desktop is growing faster, and applications are getting more and more sophisticated.

For each step of evolution, it is important to reveal the basic aspects that drive the evolution flow. This work aims to reveal how the major convergence vectors are arriving to Web applications. In that way, this diploma thesis focuses on the identification, description and analyses of the new user interaction patterns and models available for Web Applications, possible only after the Ajax rise. Ajax itself it's leveraging the Web as it stands for standards and interaction models evolution.

This thesis provides an historic overview about user interfaces on traditional Web and desktop applications, as well as identifying the major differences and pointing the convergence tendencies between Desktop and Web user interfaces and interaction models.

This thesis also provides a brief description on what is Ajax, how it works and the major trends for the future of this Web technology. Understanding the evolution from the past to nowadays in user interaction on Web applications and the Ajax role in this evolution, allowed to trace some probabilistic scenarios about how interaction technologies, on Web and Desktop applications, will continue to evolve in the near future.

At the end of the work we start to understand that the future for Ajax technologies reveals several challenges and opportunities. Ajax is pointed as successful technology on the Web because it respects the standards and it is based on open and worldwide available technologies.

The continuous evolution and convergence of Web and Desktop will keep going on, but it is not expected the merge phenomenon between the two worlds. The diverging foundational aspects for each platform do not allow the complete fusion, instead the most probabilistic scenario points the difference between Web and Desktop will be transparent for the end users.

Prefácio

Invariavelmente a interação com o utilizador é deixada para segundo plano no que respeita à engenharia de software. Todavia, este trabalho pretende apresentar e fundamentar toda uma análise sistémica sobre esta disciplina do conhecimento, bem como, salientar a sua relevância e impacto no resultado e sucesso de um processo de engenharia de software.

Este trabalho apresenta um conjunto sólido de ideias sobre o estado da arte dos paradigmas actuais de interação com o utilizador e promove o debate sobre pistas, preocupações e tendências para a evolução em curso.

Acredito também que o impacto do trabalho possa variar de pessoa para pessoa, de acordo com o respectivo percurso científico e profissional, para as quais lanço desde já o desafio, e coloco-me à disposição, para conversar e debater abertamente a análise e ideias apresentadas neste trabalho.

Agradecimentos

Aproveito para agradecer e expressar o meu especial apreço por algumas pessoas que me acompanharam ao longo do processo de escrita deste trabalho e sem as quais não teria conseguido chegar até aqui.

Começo por agradecer a orientação e coordenação do Prof. Dr. Ademar Aguiar e a visão reveladora apresentada por Florian Moritz, cujas ideias inspiraram e influenciaram fortemente o trabalho realizado.

Uma palavra de agradecimento para todos os autores de trabalhos referidos pela fonte de conhecimento e inspiração que constituíram, inclusivamente, para toda comunidade da Wikipedia.

A todos os colaboradores da iUZ Technologies pela oportunidade e todas aquelas conversas filosóficas (que fomos tendo ao longo do tempo e pareciam não fazer sentido) na base das ideias mais interessantes presentes neste trabalho.

Não podia deixar de agradecer aos meus pais e irmã pelas condições, incentivos e força que me transmitiram ao longo de todo o mestrado no sentido de conseguir levar este desafio até ao fim.

Por fim, mas não menos importante, expressar a toda a minha gratidão, pelas preocupações, metodologia e acima de tudo injeções de motivação, para com a minha namorada Anna Kustra.

Índice

1	Introdução	11
1.1	Objectivos	11
1.2	Motivação	12
1.3	Contexto	12
1.4	Metodologia e resultados	13
1.5	Organização do documento	13
2	Convergência e comparação entre interacção na Web e no Desktop	15
2.1	Convergência entre Web e Desktop IU	15
2.1.1	Convenções e terminologia	15
2.2	Heurísticas de Usabilidade	19
2.2.1	Evolução das Interfaces com o Utilizador	22
2.2.2	Migração de Desktop para Web	34
2.3	Comparação entre IU Web e Desktop	37
2.3.1	Experiência de Utilização	38
2.3.2	Conteúdos e Informação	40
2.3.3	Paradigmas de Interacção	48
2.3.4	Usabilidade	55
2.3.5	Aspecto visual (<i>look and feel</i>)	65
2.4	Síntese e conclusões	67
3	Aplicações Ajax	68
3.1	Aspectos essenciais - Ajax	68
3.2	Tecnologias Ajax	70
3.3	Ajax - Desafios e oportunidades	72
3.3.1	Standards e Compatibilidade	73
3.3.2	Actualização dos Browsers	74
3.3.3	O Desktop	75
3.3.4	Flash	75
3.3.5	Two-Way Web	76
3.3.6	Armazenamento local de dados	76
3.3.7	XForms	77
3.3.8	JavaScript	78
3.3.9	Suporte ao Desenvolvimento	78
3.3.10	Maior largura de banda	78
3.4	Bibliotecas e Frameworks Ajax	79
3.5	Síntese e conclusões	80
4	Padrões Ajax	81
4.1	Padrões de Software	82

4.2	Padrões Ajax.....	84
4.3	Padrões Ajax centrados na Interação com o Utilizador	86
4.4	Anatomia de um Padrão	88
4.5	Padrão: Aplicação Ajax	90
4.6	Síntese e conclusões	97
5	Padrões Ajax centrados na interação com o Utilizador	98
5.1	Padrão – Display Morphing	99
5.2	Padrão – Page Rearrangement	102
5.3	Padrão – XMLHttpRequest Call	105
5.4	Padrão – User Action	108
5.5	Padrão – Scheduling	112
5.6	Padrão – XML Message.....	114
5.7	Padrão – Submission Throttling	117
5.8	Padrão – Progress Indicator	120
5.9	Padrão – Rich Text Editor	123
5.10	Padrão – Drag and Drop	125
5.11	Padrão – Malleable Content.....	128
5.12	Padrão – MicroLink	131
5.13	Padrão – Highlight.....	134
5.14	Síntese e Conclusões	136
6	Aplicações baseadas na Internet.....	137
6.1	Aplicações Web tradicionais	137
6.2	Aplicações Web baseadas em Ajax	138
6.2.1	Google Maps.....	138
6.2.2	Kiko Calendar.....	138
6.2.3	Google Docs	138
6.2.4	iGoogle.....	138
6.2.5	Google Mail	139
6.3	Mapeamento de Padrões Ajax em aplicações Web	139
6.4	Síntese e Conclusões	141
7	Conclusões	142
	Referências bibliográficas.....	144
	Anexos.....	147
	Anexo A – Lista de motores de busca	148

Lista de figuras

Figura 1: RIA: Categorização de acordo com a origem do processamento [1]	16
Figura 2: Geração 2: Linha de comandos do windowsXP	24
Figura 3: Nicola Pellow's Line Mode Browser	25
Figura 4: A Guerra dos Browsers [15]	26
Figura 5: Quantidade de informação digital produzida em 2007 [35]	40
Figura 6: IU da Google para pesquisa de informação	41
Figura 7: IU - Live Search: pesquisar informação	43
Figura 8: IU - Yahooo Search: pesquisar informação.....	43
Figura 9 e 10 : IU – Windows XP: Pesquisar (esquerda). IU – Leopard Spotlight: pesquisar informação (direita)	44
Figura 10: IU – Google Desktop (Red Hat Linux)	45
Figura 11: Explorador de Ficheiros (Windows XP) Clique único vs. duplo clique	49
Figura 12: Browser: Página Seguinte e Anterior (esquerda) vs. Anular (direita).....	53
Figura 13: Ambiente de trabalho (Linux)	56
Figura 14: Teclas <i>Enter</i> e <i>Return</i> (teclado convencional).....	57
Figura 15: Meebo – Janelas dentro de uma aplicação Web	58
Figura 16: iGoogle – Manipulação directa de elementos em aplicação web.....	59
Figura 17: Google Docs – Interface Gráfica para o Utilizador	60
Figura 18: Google Docs: File Menu	61
Figura 19: Principais funcionalidades do Browser (Firefox)	61
Figura 20: GMDesk (Robert Nyman).....	63
Figura 21: Prism (Mozilla Labs).....	64
Figura 22: Prism – Criar acesso para aplicação web	64
Figura 23: Dojo, exemplo de GUI (Ajax File Browser).....	66
Figura 24: ZK – Direct Ria, exemplo de GUI (Layout de portal)	66
Figura 25: Detergente AJAX (produto da Colgate Palmolive)	74
Figura 26: Twisted – Event Driven Networking Engine	76
Figura 27: Padrões Ajax: Áreas de especialização	85
Figura 28: Elementos presentes nos esboços dos padrões.....	88
Figura 29: Padrão: Aplicação Ajax	90
Figura 30: W3C - Grupos de Interesse actuais	96
Figura 31: Variedade de dispositivos móveis	96
Figura 32: Padrão: Display Morphing	99
Figura 33: Google Mail: Corrector ortográfico	101
Figura 34: Padrão: Page Rearrangement.....	102
Figura 35: Kiko Calendar: detalhes de entrada no calendário.....	104
Figura 36: Padrão: XMLHttpRequest Call	105
Figura 37: GMail: conversa��o em tempo real.....	107
Figura 38: Padr��o: User Action	108

Figura 39: Padrão: Scheduling	112
Figura 40: Google Docs e Mail: Gravação automática	113
Figura 41: Padrão: XML Message	114
Figura 42: Google Maps	116
Figura 43: Padrão: Submission Throttling	117
Figura 44: Google Mail: Escolha de destinatários	119
Figura 45: Padrão: Progress Indicator.....	120
Figura 46: Google Mail – Carregamento de dados.....	121
Figura 47: Padrão: Rich Text editor.....	123
Figura 48: Google Docs – Editor de texto.....	124
Figura 49: Padrão: Drag and Drop	125
Figura 50: iGoogle – Reorganização de painéis.....	127
Figura 51: Padrão: Malleable Content.....	128
Figura 52: iGoogle – Malleable Content	130
Figura 53: Padrão: MicroLink.....	131
Figura 54: Google Mail – Lista de mensagens encadeadas.....	133
Figura 55: Padrão Highlight.....	134
Figura 56: Google Mail – Mensagens assinaladas.....	136

Lista quadros

Quadro 1: RIA: descrição e exemplos das várias categorias	16
Quadro 2: Heurísticas de Usabilidade de Jakob Nielsen aplicadas a AW e AD	20
Quadro 3: Gerações de Interfaces com o Utilizador.....	22
Quadro 4: Browsers: quota de mercado [22].....	28
Quadro 5: Lacunas da web tradicional vs. Mais-valias das RIA.....	33
Quadro 6: Biblioteca e Frameworks Ajax	80
Quadro 7: Padrões AJAX centrados na interação com o utilizador.....	86
Quadro 8: Mapeamento de Padrões Ajax em aplicações actuais	140

Glossário

Ao longo do trabalho irei traduzir, sempre que possível, os termos e abreviaturas utilizados. Irei no entanto recorrer aos termos originais regularmente uma vez que acredito que, devido à sua notoriedade e maturidade, tornam a sua comunicação mais prática e eficaz.

TERMO	DESCRIÇÃO
Ajax	Asynchronous JavaScript and XML
API	Application Programming Interface
Browser	Software que permite ao utilizador interagir com recursos disponíveis na Web
CSS	Cascading Style Sheets,
Desktop	Software disponível para computadores pessoais (Aplicações cliente)
DHTML	Dynamic HTML
DOM	Document Object Model
GUI	Graphical User Interface
HCI	Human Computer Interaction
HTML	Hyper Text Markup Language
IDE	Integrated Development Environment
ISO	International Organization for Standardization
JavaEE/JEE	Java Enterprise Edition, anteriormente conhecido por J2EE
JSON	JavaScript Object Notation
RIA	Rich Internet Applications
SDK	Software Development Kit
SOAP	Simple Object Access Protocol
IU	Interface com o Utilizador (User Interface - UI)
UML	Unified Modeling Language
URI	Universal Resource Identifier
Web	World Wide Web
Web 2.0	Segunda geração de aplicações desenvolvidas tendo por base a WWW
WWW	World Wide Web
WYSIWYG	What You See Is What You Get
XML	Extensible Markup Language
XPath	XML Path Language
XQuery	XML Query Language
Flash	Aplicação de autor para desenvolvimento de conteúdos/aplicações multimédia
URI	Uniform Resource Identifier
HTTP	Hypertext Transfer Protocol
FTP	File Transfer protocol

1 Introdução

1.1 Objectivos

Ao longo dos anos a Internet (World Wide Web - WWW) tem evoluído do ponto de vista conceptual e tecnológico. Esta evolução traduz-se directamente nas características das aplicações disponibilizadas para esta plataforma. Aplicações estas que têm evoluído com base em inovações tecnológicas e na maturidade de uma plataforma disponível para todo o mundo e por consequência cada vez mais poderosa.

A acompanhar esta evolução está a crescente importância e forma como as pessoas se relacionam e interagem diariamente com um conjunto de aplicações de cariz profissional e pessoal.

Os paradigmas de interacção com o utilizador são de facto uma das áreas onde maiores evoluções se podem observar. Nomeadamente, algumas das limitações apontadas às aplicações baseadas na Internet, comparativamente às suas congéneres desenvolvidas para Desktop, vão sendo ultrapassadas graças a novas capacidades dos *browsers*, bem como, à utilização de forma inovadora de tecnologias standard que se encontram amplamente disseminadas (Capítulo 6).

Uma das utilizações inovadoras mais relevantes nos tempos que correm, e sobre a qual recai o foco deste trabalho, é designada por Ajax e veio revolucionar a forma como os utilizadores interagem com aplicações baseadas na internet.

Nesse sentido pretende-se captar e documentar os principais padrões de utilização da arquitectura Ajax na base da revolução Web 2.0 no que respeito diz à interacção com o utilizador.

Este trabalho terá por base duas referências bibliográficas fundamentais:

- Rich Internet Applications (RIA): A Convergence of User Interface Paradigms of Web and Desktop, de Florian Moritz [1];
- Ajax Design Patterns: Creating Web2.0 (...), de Michael Mahemoff [3];

Muito do material presente neste trabalho é baseado nas ideias defendidas pelos referidos autores, sendo que a mais-valia do presente trabalho, consiste na ponte sistematizada entre as tendências de convergência apontadas por Florian e os Padrões de desenho apresentados por Michael.

Com base nesta duas visões complementares, será então traçada uma visão sistémica sobre a evolução verificada ao longo do tempo de existência da Web e apresentados cenários prováveis sobre para onde esta poderá continuar a evoluir.

1.2 Motivação

Compreender o que esteve na génese das transformações ocorridas na Internet nos últimos anos, na vertente relacionada de interacção com o utilizador, permitirá perceber concretamente o que mudou e acima de tudo os motivos na base dessa mudança. Desta forma teremos uma base sólida de conhecimento que nos permita encontrar as respostas para questões como:

- Os utilizadores estavam sedentos de novas formas de interacção ou tratou-se de um fenómeno puramente tecnológico?
- O porquê do sucesso do Ajax comparativamente com outras tecnologias de interacção com utilizador que surgiram no passado (e.g. Flash, JavaApplets) mas não atingiram tão grande notoriedade?
- Trata-se de um fenómeno passageiro que tenderá a desaparecer ou será cada vez mais utilizada passando a um estatuto de standard nas aplicações baseadas na internet?

Através da chave que eventuais respostas às questões indicadas nos possam apresentar, será possível decifrar, projectar e preparar o caminho a percorrer nos próximos tempos, no que respeito diz à engenharia de aplicações baseadas na internet.

E sendo servir o utilizador o propósito principal de uma grande maioria das aplicações, perceber de que forma o podemos fazer melhor, com maior eficiência e maior eficácia será por certo uma mais-valia para todos os que se dedicam à actividade de desenvolvimento de aplicações baseadas na Internet.

1.3 Contexto

A oportunidade de concretizar este trabalho surge no âmbito da dissertação no mestrado de Engenharia Informática, ministrado pela Faculdade de Engenharia da Universidade do Porto, o qual frequentei de 2006 a 2009. A participação no mestrado permitiu uma aproximação significativa às várias temáticas relacionadas com a Engenharia de Software.

Conjugando os conhecimentos adquiridos com o mestrado, com a formação de base, (licenciatura em Novas Tecnologias da Comunicação), juntamente com 6 anos de experiência na área de Engenharia de Software, surgiu a consciência, motivação e o *know-how* necessário para identificar e analisar o fenómeno indicado, bem como, a capacidade de apontar respostas para algumas das questões elencadas.

1.4 Metodologia e resultados

Para atingir os resultados pretendidos o trabalho realizado consistiu na pesquisa e análise da história e do estado actual das aplicações Web e Desktop. Análise esta enriquecida por ideias e direcções apontadas por diversos autores de diferentes áreas de especialização (e.g. programação, design e usabilidade).

Com base na análise realizada e constatações pessoais, provenientes de experiência no desenvolvimento de aplicações Web com especial enfoque na interacção com o utilizador, foi possível sistematizar alguns dos motivos na base do salto evolutivo verificado nos modelos de interacção com o utilizador na emergente nova geração de aplicações Web.

A sistematização apresentada contempla também a descrição e demonstração pormenorizada dos padrões de interacção com o utilizador mais relevantes no fenómeno em curso da convergência entre aplicações Web e Desktop.

Realço mais uma vez o papel preponderante de dos autores Florian Moritz [1] e Michael Mahemoff [3], cujos trabalhos serviram de guião para o trabalho realizado.

1.5 Organização do documento

Capítulo 1: Introdução

Este primeiro capítulo descreve os objectivos, motivação metodologia utilizada.

Capítulo 2: Convergência e Comparação entre Web e Desktop IU

O segundo capítulo pretende descrever o fenómeno da convergência entre aplicações Desktop e aplicações baseadas na Internet, identificando com isso as motivações e desafios que se colocam.

Capítulo 3: Aplicação Ajax

Neste capítulo descreve-se de forma detalhada o funcionamento e arquitectura do paradigma de desenvolvimento baseado em Ajax, descrevendo o funcionamento e papel das várias tecnologias inerentes.

Capítulo 4: Padrões Ajax

Neste capítulo pretende-se sistematizar e organizar as diferentes áreas de actuação para as quais foram encontradas soluções recorrentes (Padrões) com base na tecnologia Ajax.

Capítulo 5: Padrões Ajax centrados na interação com o utilizador

Neste capítulo, pretende-se identificar e descrever concretamente padrões Ajax que possibilitam a implementação de paradigmas de interação com o utilizador em aplicações baseadas na internet que correspondem às expectativas dos utilizadores, com base nos conceitos e modelos mentais provenientes de aplicações Desktop.

Capítulo 6: Aplicações Emergentes

Após identificar os padrões mais relevantes para a convergência anunciada, serão apresentados alguns exemplos de aplicações emergentes onde é possível verificar as mais-valias introduzidas pelos novos Padrões de interação com o utilizador, presentes em Aplicações Web que utilizam a tecnologia Ajax.

Capítulo 7: Conclusões

Por fim, pretende-se elencar um conjunto de deduções e conclusões resultantes da análise e experiências realizadas. Pretende-se também elencar, de igual forma, um conjunto de desafios em aberto e trabalho a realizar no âmbito do tema apresentado.

2 Convergência e comparação entre interacção na Web e no Desktop

2.1 Convergência entre Web e Desktop IU

Ao longo deste trabalho iremos comparar dois mundos: **Web** e **Desktop**. O mundo Web representa todo o conjunto de tecnologias e características inerentes às aplicações baseadas na plataforma World Wide Web (WWW) ou vulgarmente conhecida por Internet. O mundo Desktop representa todo um conjunto de tecnologias e características disponíveis em aplicações instaladas num computador. Devido ao facto de, no contexto deste trabalho, ambas as palavras representarem um contexto de utilização completo, ambas serão escritas com letra maiúscula no início.

2.1.1 Convenções e terminologia

Antes de entrarmos numa análise pormenorizada sobre IU em aplicações Web e Desktop, torna-se fundamental investir algum tempo em tentar perceber como se podem categorizar os diferentes tipos de aplicações de software.

Tipicamente, o modo de funcionamento das aplicações de software implica processamento do lado do servidor e/ou na máquina cliente. Uma vez que a quantidade de processamento realizada em cada um dos mundos (servidor ou máquina cliente) diverge nos diferentes tipos de aplicações, torna possível o exercício de organizar os diferentes tipos de aplicações desenvolvidas com base na arquitectura cliente-servidor.

Termos como Aplicações Desktop (AD), Web-enabled App, Smart Client, Web 2.0, Thin Client, Páginas e Sítios Web; são termos vulgarmente utilizados na tentativa de descrever sucintamente o contexto e paradigma de utilização destas aplicações. Surge então o termo Rich Internet Applications (RIA) ou Aplicações Ricas para a Internet, que pretende englobar os vários tipos de aplicações referidos anteriormente.

A Figura 1 ilustra uma proposta de organização dos diferentes tipos de RIA, apresentada por Florian Moritz [1], onde o eixo vertical determina a quantidade de processamento realizada do lado do servidor e do lado do cliente. O eixo horizontal classifica os diferentes tipos de aplicações de acordo com o paradigma principal na base do seu funcionamento.

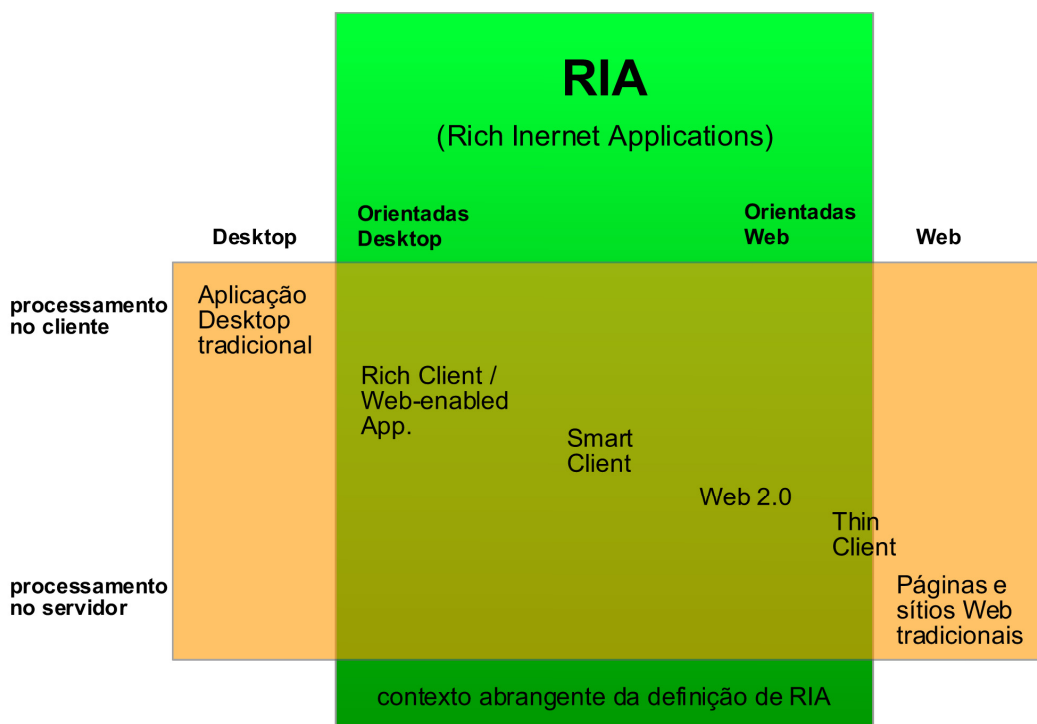


Figura 1: RIA: Categorização de acordo com a origem do processamento [1]

Quando tentamos descrever uma RIA surge inevitavelmente a necessidade de clarificar sobre se se trata de uma RIA orientada à Web (e.g. Ajax, baseada num Web Browser) ou se, pelo contrário, se trata de uma RIA orientada ao Desktop (e.g. Java, Flash integrado com AIR).

Por vezes identificar a categoria exacta de uma aplicação torna-se numa tarefa complicada, de qualquer forma, a Figura 1 apresenta-se como uma excelente ferramenta para desambiguar a categoria mais adequada para cada aplicação. Esta classificação revela-se importante na medida em que as várias categorias contemplam aplicações com paradigmas de interacção diferenciados que influenciam de forma significativa os padrões de interacção com o utilizador.

Quadro 1: RIA: descrição e exemplos das várias categorias

Categoria	Descrição
Aplicação Desktop tradicional	Representa a típica aplicação que necessita de ser instalada num sistema operativo. E.g.: Microsoft Word, Adobe Photoshop
Rich Client / Web-enabled App.	Aplicação Desktop que recorre à conectividade disponível na Web, mas funcionam também em modo offline ou conectividade reduzida. E.g.: Microsoft Outlook, NetBeans

Smart Client	E.g. iTunes, gTalk
Web 2.0	E.g.: gMail, LinkedIn
Thin Client	Aplicações baseadas num Browser, instalado na máquina do cliente, e numa ligação à Web. Vulgarmente requerem que o utilizador se autentique para aceder à sua área pessoal. E.g.:
Páginas e sítios Web tradicionais	Aplicações iniciadas a partir da Web, bastando ao utilizador um Browser para as poder utilizar. Não requerem instalação na máquina cliente. E.g.: www.google.com , www.fe.up.pt

O termo RIA tenta descrever as aplicações situadas algures entre os dois mundos Web e Desktop. Uma RIA pode necessitar ou não de um browser para ser utilizada, mas o aspecto principal reside no facto de uma RIA combinar as mais-valias de ambos os mundos: Web e Desktop.

2.1.1.1 RIAs orientadas para a Web e Desktop [38]

As RIAs orientadas para a Web apresentam uma experiência de interacção mais rica e personalizada. Consideremos como exemplo das redes sociais como o “hi5¹” e o “facebook²”, calendários e clientes de email como os disponibilizados através da “Google Mail³ e Calendar⁴” em que para as utilizarmos necessitamos de um Browser. Estas são de todo as aplicações (RIA) orientadas para a Web.

No outro lado temos aplicações RIA que necessitam de ser instaladas no computador e cujo *look and feel* se aproxima ao conceito de aplicação de Desktop (AD). Estas aplicações caracterizam-se pelas suas Interfaces com o Utilizador (IU) repletas de interactividade e inovadores mecanismos de interacção. No entanto, o que as faz caracterizar como RIA, e não como aplicações Desktop, está relacionado com o facto de recorrerem à Web para determinadas funcionalidades que potenciam a sua utilização. Consideremos o “Apple iTunes⁵” como exemplo de uma RIA orientada ao Desktop. No seu estado puro funciona como um leitor multimédia no computador com capacidade de reproduzir vídeos e música. Por outro lado, quando ligado à Web permite a compra de música, funcionando praticamente como o web site de compras

¹ Hi5: <http://www.hi5.com>

² Facebook: <http://www.facebook.com>

³ Google Mail: <http://mail.google.com>

⁴ Google Calendar: <http://www.google.com/calendar>

⁵ iTunes: <http://www.apple.com/pt/itunes/download/>

online, permite o download de trailers de filmes e programas de entretenimento e outros conteúdos multimédia.

Aliás, o exemplo do “Apple iTunes” vai até um pouco mais além do conceito de RIA, uma vez que permite também interacção com dispositivos móveis, como o “ Apple iPod e Apple iPhone⁶”. Neste contexto o exemplo apresentado é uma aplicação transversal a várias categorias apresentadas, incorporando mais-valias de ambos os mundos (Web e Desktop). Será no entanto considerada como uma RIA orientada ao Desktop por dois motivos:

- Necessita de instalação no computador;
- Funciona mesmo sem conectividade com a Web;

2.1.1.2 WEB 2.0

O termo **Web 2.0** é recorrentemente utilizado na tentativa de classificar este tipo de aplicações que apresentam características de ambos os lados (Desktop e Web). De facto o termo Web 2.0 foi introduzido por Tim O'Reilly em 2005 [7] e [8]:

“This paper was the first initiative to try to define Web2.0 and understand its implications for the next generation of software, looking at both design patterns and business modes. Web 2.0 is the network as platform, spanning all connected devices; Web 2.0 applications are those that make the most of the intrinsic advantages of that platform: delivering software as a continually-updated service that gets better the more people use it, consuming and remixing data from multiple sources, including individual users, while providing their own data and services in a form that allows remixing by others, creating network effects through an "architecture of participation," and going beyond the page metaphor of Web 1.0 to deliver rich user experiences.”

A proposta de definição apresentada vai para além das mais-valias das aplicações Desktop e Web. Apresenta de uma forma inovadora uma visão para a Internet como uma plataforma de serviços e aplicações. Um aspecto mais importante, até que a tecnologia utilizada, é o efeito das redes de sociais, o facto de o conteúdo desta **Web 2.0** ser produzido por um conjunto de utilizadores e não por uma única entidade.

No que toca a tecnologia, a Web 2.0 baseia-se também na utilização de HTML em colaboração com Ajax, um conjunto de tecnologias maduras e com uma utilização alargada pela Web. Iremos abordar o tema Ajax com maior profundidade no capítulo “3 Aplicações Ajax” deste trabalho.

⁶ iPod - <http://www.apple.com/pt/ipodclassic/> , iPhone - <http://www.apple.com/pt/iphone/> ; Acesso: 2009-03-11

2.1.1.3 “Dois irmãos separados à nascença”

Como podemos observar pela análise apresentada no ponto anterior, a tentativa de atribuir nomes e categorias para os termos utilizados ao longo do trabalho é tudo menos simples. Contudo, serve de base para a discussão dos temas abordados nos pontos seguintes.

Os dois mundos apresentados Aplicações Desktop (AD) e Aplicações Web (AW), têm vindo a evoluir lado a lado, ainda que na sua origem não tenha sido colocada ou prevista a possibilidade de se poderem fundir no futuro.

As AW têm por base páginas estáticas de HTML⁷ para o transporte de informação. Por outro lado as AD pretendiam ser aplicações sem rede (*standalone*) com uma forte componente de interactividade.

Pretender que AW, originalmente pensadas para o transporte de informação, apresentem a interactividade típica de uma AD recorrendo às tecnologias actualmente disponíveis para a plataforma Web, é de todo uma intenção ambiciosa mas possível em determinados aspectos.

Ao longo deste trabalho serão por isso demonstrados Padrões de Interação com o Utilizador, provenientes originalmente de AD, que começam a estar disponíveis na geração emergente de aplicações **Web 2.0**, assim como, explicado o papel potenciador desempenhado pela arquitectura Ajax na evolução em curso.

2.2 Heurísticas de Usabilidade

Sempre que falamos no desenvolvimento de aplicações e no respectivo desenho das interfaces com o utilizador (IU), é fundamental ter em conta um conjunto de heurísticas de usabilidade que servem como guias de orientação sobre boas ou más soluções.

Uma Interface com o Utilizador (IU) é a componente de um produto que permite ao utilizador introduzir instruções e receber feedback da aplicação.

A norma ISO 9241-11 [9], apresenta a seguinte definição de usabilidade:

“Usability: the extent to which a product can be used by specified users to achieve specified goals with effectiveness, efficiency and satisfaction in a specified context of use.”

A definição apresentada sugere que a usabilidade de um produto, neste caso uma aplicação de software, caracteriza a eficácia, eficiência, simplicidade e satisfação com que os utilizadores utilizam e obtêm os resultados esperados de uma aplicação num determinado contexto. O que significa que quanto melhor for a experiência de utilização de um determinado produto, melhor a usabilidade do mesmo.

⁷ HTML: Hypertext Markup Language

Jakob Nielsen's apresenta 10 boas práticas de usabilidade (heurísticas de usabilidade) [10], para o desenho de Interfaces para o Utilizador (IU), sem distinguir se se tratam de aplicações Desktop ou Web, nem considerando a tecnologia em que estas serão implementadas.

O quadro seguinte apresenta algumas considerações, relativamente à aplicação dessas heurísticas a AW e AD.

Quadro 2: Heurísticas de Usabilidade de Jakob Nielsen aplicadas a AW e AD

Heurísticas de Usabilidade	Aplicações Web (AW)	Aplicações Desktop (AD)
Visibilidade do estado do sistema	Devido ao paradigma <i>request-response</i> , antes de efectuarmos o pedido não sabemos se a aplicação está ou não disponível	Perceptível de imediato devido à comunicação estreita entre a aplicação e o sistema operativo
Proximidade com o contexto do utilizador	Devido à maturidade da área e competências de design de interacção e design gráfico, esta aproximação ao vocabulário e contexto de utilização em que uma determinada aplicação vai ser utilizada, esta heurística está disseminada de igual forma pelos dois tipos de aplicações	
Controlo do utilizador e flexibilidade de utilização	Tipicamente as operações de controlo estão disponíveis através de botões do Browser. Qualquer tentativa de implementar soluções específicas para estas operações, poderá entrar conflito com os controlos do browser.	As operações de controlo de uma aplicação, de acordo com o sistema operativo em que está a ser executada, disponíveis consistentemente em todas as aplicações.
Standards e consistência	Devido à sua maturidade pelo facto de existirem à mais tempo, existem mais standards actualmente para AD ainda que, com o decorrer da evolução nos últimos anos, têm surgido trabalhos na óptica da definição e adaptação de standards para Interfaces com o Utilizador (IU) de Aplicações Web (AW).	
Prevenção de erros	O design de Interfaces para o Utilizador (IU) depende da estratégia e experiência de desenvolvimento do Designer de Interacção. Todavia, tradicionalmente devido à maior interactividade que possibilitam, as Aplicações Desktop (AD) apresentavam esta característica de uma forma mais intuitiva para o utilizador. Na actualidade, com o advento do Ajax, as Aplicações Web (AW) começam a recuperar terreno através de soluções semelhantes às utilizadas nas AD.	
Modelos conceptuais	Esta heurística depende da implementação e experiência do Designer de Interacção, não havendo diferenças significativas entre AW e AD.	
Flexibilidade e eficiência de utilização	Restrições com base nas tecnologias de interacção com o utilizador disponíveis. Impacto significativo do Ajax na evolução	Maior facilidade e flexibilidade na implementação e inovação nos paradigmas de interacção, e.g. manipulação directa, através de

	do paradigma e padrões utilizados.	<i>drag and drop</i> de elementos na interface com o utilizador.
Aspecto estético da IU	Maior flexibilidade e facilidade de inovar e criar interfaces gráficas mais apelativas. Maior criatividade e maior diversidade podem causar uma maior confusão nos utilizadores.	Restrição ao nível de componentes standards e predefinidos pela tecnologia e sistema operativo. Ao mesmo tempo, constitui o mais importante aspecto na consistência e coerência entre AD.
Reconhecimento e recuperação de erros	Não existem grandes diferenças a assinalar, depende contudo da competência e experiência dos designers envolvidos.	
Ajuda e documentação	Tipicamente as AD tendem a apresentar ajuda de contexto mais adequada, uma vez que têm acesso a vários aspectos da interacção com o utilizador. Esta heurística tem vindo a ser melhorada na AW, ainda que na grande maioria das vezes se optem por estratégias de estatística e extrapolação para determinar o ponto e os conteúdos de ajuda a disponibilizar.	

A análise apresentada no quadro anterior, constata que cada uma das dez heurísticas pode ser conseguida, com maior ou menor dificuldade em AW e AD. Mais importante do que o tipo de aplicação a desenvolver é a preocupação, competência e experiência do Designer de Interacção responsável.

Ainda que a análise anterior pretenda ser agnóstica no que toca às tecnologias utilizadas, do ponto de vista tecnológico será mais fácil conseguir respeitar todas as heurísticas num contexto que permita maior flexibilidade na implementação da IU e respectiva experiência de utilização. Esta flexibilidade é maior em Aplicações Desktop (AD) uma vez que envolvem tecnologias com acesso aos recursos físicos do sistema como processador, placa gráfica e disco rígido, possibilitando a implementação de modelos de interacção mais sofisticados.

Por outro lado, as tecnologias disponíveis para design de interfaces com o utilizador (IU) em Aplicações Web (HTML e CSS⁸) permitem elevar a criatividade dos designers a novos níveis. Todavia, esta flexibilidade resulta numa maior diversidade de modelos mentais de utilização e paradigmas que podem confundir os utilizadores e por vezes diminuir ou piorar a experiência de utilização de uma determinada aplicação.

No actual estado de evolução e maturação de ambos os mundos (Web e Desktop), as diferenças na interactividade disponível tende a esvanecer. Isto é, a utilização inovadora de arquitecturas como o Ajax, está na origem de uma forte impulso à convergência de Padrões de Interacção com o Utilizador presentes em Aplicações Web (AW) e Aplicações Desktop (AD).

⁸ HTML e CSS: tecnologias desenvolvidas de raiz para formatar e trabalhar o aspecto gráfico da informação

2.2.1 Evolução das Interfaces com o Utilizador

No sentido de melhor percebermos a evolução ocorrida ao longo dos tempos no que respeita às Interfaces com o Utilizador (IU), importa compreender as diferentes etapas percorridas e as grandes evoluções que as distinguem umas das outras.

O tópico actual irá traçar uma perspectiva histórica sobre as várias gerações de IU para Aplicações Desktop (AD), tentar decifrar como essa evolução está a ocorrer nas Aplicações Web (AW) e por fim enquadrar de que forma as RIA representam uma evolução e convergência entre ambos os mundos.

2.2.1.1 Evolução das IU nas Aplicações Desktop

Muito antes do conceito de Interface com o Utilizador (IU) existir ou ser perceptível como tal, o conceito em si consistia em estruturas muito simples. Jacob Nielsen, através do seu trabalho em “*Generations of User Interfaces*” [11] de 1993, apresenta a sua visão sobre as gerações de interfaces que existiram desde o início da era informática.

O quadro seguinte apresenta um resumo sobre a perspectiva do actor.

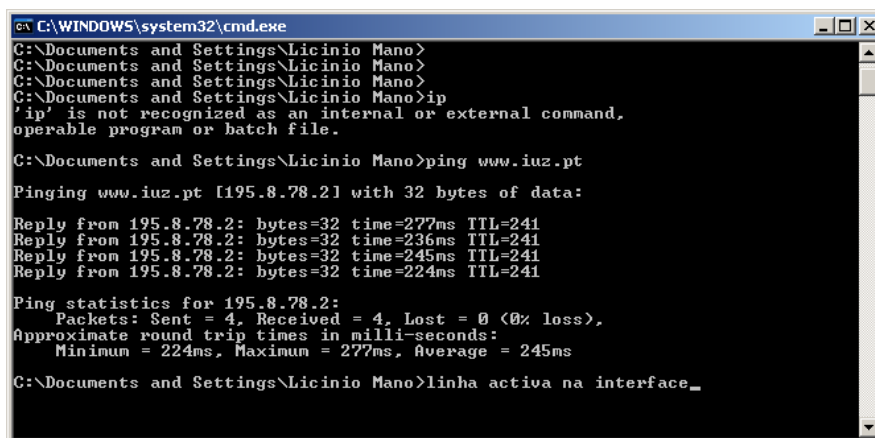
Quadro 3: Gerações de Interfaces com o Utilizador

GERAÇÃO	DESCRIÇÃO DA INTERFACE COM O UTILIZADOR
Geração 0	Pré-história (até 1945) Não existente, utilização directa no hardware.
Geração 1	Pioneiro (de 1945 a 1955): Sistemas Batch <i>0 Dimensões:</i> Interfaces com o utilizador não interactivas, em que a interacção era restrita a um ponto no tempo. O utilizador tinha de introduzir todos os comandos de uma só vez, antes sequer de obter o resultado de alguma dos referidos comandos.
Geração 2	Histórico (de 1955 a 1965): Linha de comandos <i>1 Dimensão:</i> As Interfaces com o Utilizador (IU) baseadas em linhas de comandos eram típicas de grandes computadores, permitindo até o acesso de múltiplos utilizadores. A interactividade é restrita à modificação da última linha presente na IU. As linhas anteriores não podem ser alteradas. Este tipo de interfaces ainda está disponível nos actuais sistemas operativos ver Figura 2: Geração 2: Linha de comandos do windowsXP.
Geração 3	Tradicional (de 1946 a 1980): Ecrã completo <i>2 Dimensões:</i> Interfaces com o Utilizador que possibilitava a alteração de todo o ecrã, permitindo também que o utilizador navegue para trás e para a frente nas operações que pretende. A interacção é baseada em menus hierárquicos onde a apresentação de informação é conseguida basicamente através de

	caracteres e objectos gráficos simples.
Geração 4	Moderno (de 1980 a 1995): Interfaces Gráficas para o Utilizador / WIMP <i>2,5 Dimensões:</i> Os fundamentos base para as Interfaces Gráficas para o Utilizador (Graphical User Interface - GUI) começaram a surgir algures em 1960. Mas a sua utilização comercial apenas aconteceu por volta de 1980. Tipicamente caracterizam-se por: Janelas (<i>Windows</i>), <i>Icons</i>, <i>Menus</i> e Rato (<i>Pointing device</i>) – WIMP. Nielsen designa esta geração como tendo 2 dimensões e meia, devido ao paradigma aberto pela utilização de Janelas. Outra grande característica, reside na introdução do Rato como dispositivo de interacção com elementos presentes na IU. Surgem então as metáforas com o mundo real, e.g. Windows e Desktop. Dá-se o nascimento das Aplicações de Desktop (AD).
Geração 5	Futuro (de 1995 em diante): Próxima Geração de Interfaces Nielsen sugere que chegaremos às três ou mais dimensões, através da inclusão de conceitos como: tempo, áudio, comandos de voz e realidade virtual. Nielsen considera também que estes sistemas se caracterizam por elevada portabilidade e conectividade, antevendo mesmo um aumento da comunicação e partilha de informação. Sugere também que estes IU irão permitir a sua personalização de acordo com as preferências de cada utilizador. Nielsen refere também a possibilidade de poderem surgir também IU que não requerem comandos, mas de acordo com as acções do utilizador adaptam a IU ao contexto e tarefas que o utilizador pretende realizar. Tomemos por exemplo o trabalho Pattie Maes e grupo de interesse por ela fundado – Fluid Interfaces Group⁹. O dos projectos mais promissores trata-se do <i>sixthsense</i>: <i>“SixthSense’ is a wearable gestural interface that augments the physical world around us with digital information and lets us use natural hand gestures to interact with that information.”¹⁰</i> O <i>sixthsense</i> apresenta-se como uma interface de interacção gestual, que permite ao utilizador interagir, recorrendo a gestos comuns do dia-a-dia, com informação digital. Será esta a próxima geração?

⁹ Pattie Maes, <http://web.media.mit.edu/~pattie/>, Acesso: 2009-03-24

¹⁰ Sixthsense, <http://www.pranavmistry.com/projects/sixthsense/index.htm>, Acesso: 2009-03-24



```

C:\WINDOWS\system32\cmd.exe
C:\Documents and Settings\Licinio Mano>
C:\Documents and Settings\Licinio Mano>
C:\Documents and Settings\Licinio Mano>
C:\Documents and Settings\Licinio Mano>ip
'ip' is not recognized as an internal or external command,
operable program or batch file.
C:\Documents and Settings\Licinio Mano>ping www.iuz.pt
Pinging www.iuz.pt [195.8.78.2] with 32 bytes of data:

Reply from 195.8.78.2: bytes=32 time=277ms TTL=241
Reply from 195.8.78.2: bytes=32 time=236ms TTL=241
Reply from 195.8.78.2: bytes=32 time=245ms TTL=241
Reply from 195.8.78.2: bytes=32 time=224ms TTL=241

Ping statistics for 195.8.78.2:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
    Approximate round trip times in milli-seconds:
        Minimum = 224ms, Maximum = 277ms, Average = 245ms
C:\Documents and Settings\Licinio Mano>linha activa na interface_

```

Figura 2: Geração 2: Linha de comandos do windowsXP

A perspectiva de Nielsen para 1995 em diante, corresponde à realidade que podemos constatar actualmente. Efectivamente as aplicações aumentaram o seu grau de personalização, de acordo com as preferências de cada utilizador, e recorrem cada vez mais à conectividade com a Web para disseminar, partilhar e até mesmo actualizar código e conteúdos. A partilha transversal de informação entre diferentes aplicações continua ainda a ser batalha em curso, mas a possibilidade de exportar a informação em diferentes formatos representa um grande passo no sentido da partilha global.

As Interfaces com Utilizador sem comandos explícitos, ainda não se encontram disseminadas pelo utilizador comum, de modo que levarão algum tempo a serem reconhecidas como tal.

O facto de Nielsen não ter dado a devida relevância à ligação em rede e o factor preponderante que esta viria a assumir no desenvolvimento de aplicações, reside da data em que o seu trabalho foi publicado – 1994. A Web estava no início (1991) e havia ainda sido perspectivado o crescimento exponencial que veio a verificar desde então. Por esse motivo é compreensível que Nielsen não tenha dado o devido destaque às tecnologias que permitem desenvolver Aplicações Web e respectivas Interfaces com o Utilizador (IU).

2.2.1.2 Evolução das IU nas Aplicações Web

Ao longo dos últimos 18 anos as tecnologias utilizadas na Web têm evoluído no sentido de propiciarem uma experiência de utilização semelhantes à que os utilizadores estão habituados a encontrar no Desktop. Para compreendermos a evolução em curso, temos de olhar para o passado e perceber os fundamentos iniciais da Internet ou WWW (world Wide Web) na altura em que foi concebida.

Quando foi introduzida em 1991 por Tim Berners-Lee existia apenas uma aplicação para publicar páginas simples de HTML num servidor. Para consultar essa informação, era necessário utilizar um browser multi-plataforma como o apresentado na Figura 3 [13].

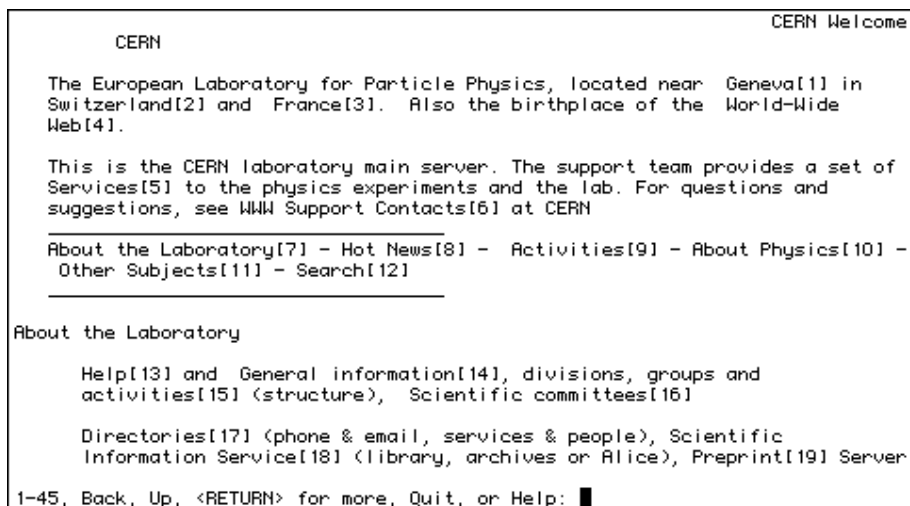


Figura 3: Nicola Pellow's Line Mode Browser

A primeira Aplicação Web, uma página web com interactividade, tratava-se de uma lista de números de telefone. Seguiram-se vários Browsers semelhantes até que em 1993 surgiu o “Mosaic” [16] que apresentava já uma interface gráfica para utilizador e a possibilidade de mostrar imagens.

Conforme fora inicialmente imaginada, o funcionamento da WWW tinha por base três conceitos:

- **URI** (*Uniform Resource Identifier*), um conjunto de caracteres simples e compacto que possibilita a identificação unívoca de cada recurso.
- **HTTP** (*Hypertext Transfer Protocol*), baseado no FTP (*File Transfer Protocol*) para o transporte e transferência de páginas Web.
- **HTML** (*Hypertext Markup Language*), para criar e apresentar informação de acordo com um determinado formato ou estilo.

Após o aparecimento dos primeiros dá-se início ao fenómeno agora denominado de “A guerra dos Browsers” (*Browser Wars*) [14]. Como é possível verificar na Figura 4: A Guerra dos Browsers [15], a referido fenómeno mantém-se até aos dias de hoje. A guerra pela maior fatia de mercado continua acesa e com ela a batalha da inovação está ao rubro fomentando ao surgimento de novas e inovadoras tecnologias.

THE BROWSER WARS

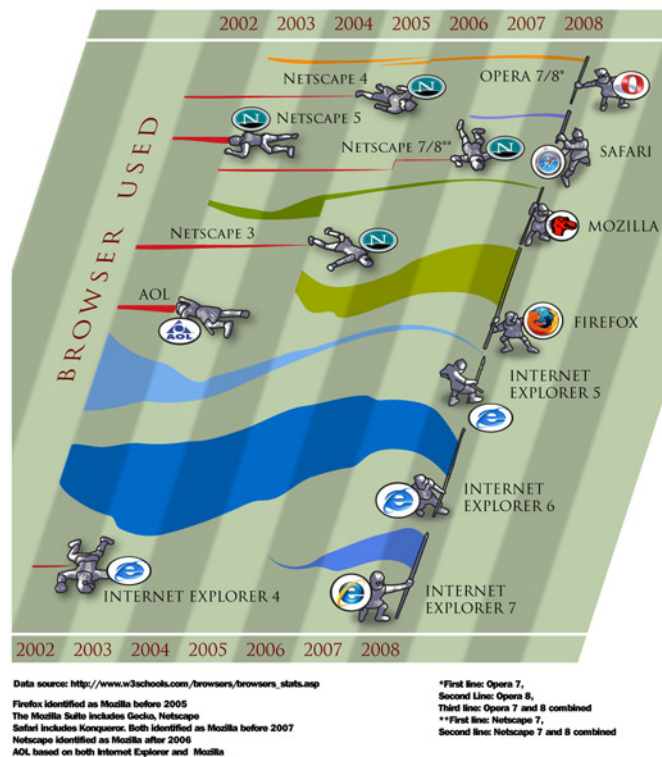


Figura 4: A Guerra dos Browsers [15]

No entanto, até chegarmos ao estado actual, foram várias as inovações tecnológicas que passaremos a descrever de seguida, com especial relevância nas evoluções tecnológicas na base da arquitectura em destaque neste trabalho – Ajax.

Por volta de 1994 surge a 1ª versão do Browser Netscape. Este browser foi desenvolvido pelos autores do Mosaic. Em 1996 surge a 2ª versão do Netscape que duas novas funcionalidades: frames e JavaScript.

JavaScript, trata-se de uma linguagem de programação do lado do cliente (corre no browser e não no servidor) com sintaxe semelhante ao Java¹¹, que revolucionou o conceito de interacção e *user experience* em páginas Web. Com a introdução de JavaScript nas páginas Web passou a ser possível modificar e controlar formulários, os conteúdos das páginas, aplicar efeitos nas imagens e os tradicionais efeitos de *hover* (quando o rato passa sobre uma área de interacção esta apresenta uma animação, e.g. muda de cor, muda a imagem, aumenta o tamanho da letra).

Este incremento de interactividade proporcionado pelo JavaScript, deve-se ao facto de este permitir aceder e modificar a estrutura do documento Web (Documento Object Model - DOM). A partir da 3ª versão do Netscape o JavaScript passou a ser um standard aprovado pela ECMA (European Computer Manufacturers Association).

¹¹ Java:

Começa então uma nova fase da Web com páginas mais interactivas, devido a uma linguagem *client-side* que era o JavaScript.

Em 1997 surge uma nova evolução relacionada com a 4ª versão do Netscape, o DHTML (Dynamic HTML). Esta nova etapa caracterizou-se pela introdução de um novo conjunto de tags HTML e, mais relevante ainda, a introdução de um novo standard, pelo W3C (World Wide Web Consortium): Cascade Style Sheets (CSS 1).

Nos anos seguintes as tecnologias evoluíram e tornaram-se amplamente disseminadas por toda a Web. Em 1995, surge então o Microsoft Internet Explorer, integrado directamente no sistema operativo mais utilizado na altura - "Microsoft windows". Com esta vantagem competitiva sobre os restantes browsers, o Internet Explorer passou rapidamente para uma posição dominante no mercado dos browsers, atingindo mesmo o monopólio do mesmo.

Mais importante do que a fatia de mercado que cada browser deteve ao longo do tempo, é o facto de trabalharem com diferentes tecnologias, especialmente tecnologias não standards. Os standards existiam, mas na tentativa de ganhar mercado ao rival, ambos optaram por implementar os seus próprios standards. Este fenómeno dificultou tremendamente a vida aos programadores de aplicações Web, uma vez que tinham de contemplar o desenvolvimento de aplicações compatíveis entre browsers, ou em último caso, indicar que a aplicação era apenas compatível com um determinado browser.

No ano de 2000 surge o Opera 4 que implementa correctamente o standard W3C DOM. A versão 4 e 5 do Opera tornaram-no no browser mais estável, mas não introduziram grandes novidades e inovações tecnológicas.

Em 2001 surge o Internet Explorer 6 (IE6) mas com um pouco suporte para CSS. Em 2002 surge o Mozilla Firefox (projecto *open source* da Mozilla Foundation [17]) que com base no Gecko Engine [18] apresentava um melhor suporte para CSS do que o IE6.

Em 2003 surge o Safari, lançado pela Apple para o sistema operativo Macintosh. O Safari teve por base um browser disponível para Linux – o Konqueror [19]

De 2006 em diante verificamos o lançamento do IE7 e novas versões do Firefox (actualmente na versão 3), Opera (actualmente na versão 9) e o Safari possui também uma versão para o sistema operativo Microsoft Windows. Mesmo com as inovações o IE7 apresenta ainda algumas lacunas na implementação dos conceitos DOM, permitindo aos outros browsers a oportunidade de recuperar fatia de mercado.

Em 2008 surge o Chrome [20], um inovador e até revolucionário browser lançado pela Google. Seguem-se um sumário dos principais motivos apresentados pela Google [21] para o desenvolvimento do Chrome:

- **Implementação de raiz**, uma vez que as pessoas passam cada vez mais tempo no browser e utilizam-nos de diferentes formas e fins (ver vídeos, jogar jogos, conversar em tempo real), muito distintas dos objectivos iniciais

definidos para um browser, a Google considera importante investir tempo no desenvolvimento de um browser de raiz, capaz de corresponder às expectativas dos utilizadores, tendo em conta o tipo de utilização de um browser na actualidade.

- **Design minimalista e transparente**, no sentido de dar espaço, a aplicações e conteúdos, viverem por si só, evitando que os controlos do browser se sobreponham à importância da Aplicação Web (AW) ou página Web que estamos a consultar.
- **Estabilidade**: o facto de existir a possibilidade de abrir várias *tabs* num browser requer também medidas prevenção e recuperação de erros. Nesse sentido, o objectivo será evitar que todo o browser bloqueie quando uma determinada *tab* encontra um erro.
- **Performance**: Desenvolvimento do V8, um motor JavaScript mais potente, para suportar a próxima geração de aplicações Web, que não seria possível nos browsers actuais, que recorrem ao Ajax para permitir uma maior interactividade.
- **Open source**: como uma boa aplicação Web 2.0 o código fonte está disponível para download, na tentativa de recolher contributos de todos os utilizadores e programadores interessados em participar no desenvolvimento do browser.

De facto, as motivações e ideias apresentadas pela Google para o desenvolvimento do Chrome representam novidades inovadoras e revolucionárias do ponto de vista dos browsers e, mais importante de tudo, relevantes para o utilizador final. Não obstante as “boas” intenções da Google, os números apresentados pelo site w3schools [22], disponíveis no Quadro 4, indicam que o Chrome tem vindo a ganhar terreno e em pouco mais de 6 meses de existência consegue deter já uma maior expressão o que o Safari e Opera. No que toca aos grandes *players* o mercado encontra-se repartido entre Microsoft Internet Explorer (IE) e Mozilla Firefox (FF), sendo que o FF tem vindo a ganhar utilizadores desde que foi lançado no mercado.

Quadro 4: Browsers: quota de mercado [22]

2009	IE7	IE6	IE8	Fx	Chrome	S	O
January	25.7%	18.5%	0.6%	45.5%	3.9%	3.0%	2.3%
2008	IE7	IE6	IE5	Fx	Chrome	S	O
December	26.1%	19.6%		44.4%	3.6%	2.7%	2.4%
November	26.6%	20.0%		44.2%	3.1%	2.7%	2.3%
October	26.9%	20.2%		44.0%	3.0%	2.8%	2.2%
September	26.3%	22.3%		42.6%	3.1%	2.7%	2.0%
August	26.0%	24.5%		43.7%		2.6%	2.1%
July	26.4%	25.3%		42.6%		2.5%	1.9%
June	27.0%	26.5%	0.5%	41.0%		2.6%	1.7%
May	26.5%	27.3%	0.7%	39.8%		2.4%	1.5%
April	24.9%	28.9%	1.0%	39.1%		2.2%	1.4%
March	23.3%	29.5%	1.1%	37.0%		2.1%	1.4%
February	22.7%	30.7%	1.3%	36.5%		2.0%	1.4%
January	21.2%	32.0%	1.5%	36.4%		1.9%	1.4%

Para além da “guerra” dos browsers o tipo de aplicações disponíveis na Web tem evoluído ao longo do tempo. Os pontos seguintes tentam sistematizar e descrever as diferentes etapas desse processo evolutivo, assim como, apresentar as características principais dos conteúdos e aplicações disponíveis em cada etapa.

Web 1.0

É o nome dado, à posteriori, ao estado da Web, estilo de design e interacção utilizados antes do advento da Web 2.0. Muitos referem-se à data de 2001 (*rebentamento da bolha*) como o ponto de viragem significativo na indústria da Web.

De acordo com a Wikipedia [23], a Web 1.0 caracterizava-se por:

- Páginas de conteúdo estático.
- Utilização de *framesets* (uma página com um frameset permite a apresentação de um ficheiro HTML por cada frame disponível)
- Tags HTML proprietárias, introduzidas com a Guerra dos Browsers (e.g. blink, marquee, bgsound).
- Guestbooks, que permitiam aos visitantes de um site indicar que tinham estado presentes, deixando o seu nome, contacto de email e eventualmente uma mensagem.
- Botões animados, tipicamente gifs animados.

Formulários enviados por email, isto é, o utilizador preenche um formulário e quando carrega em enviar, este é enviado através do seu cliente de email com a informação introduzida no formulário.

Web 2.0

Designação utilizada para um empolgante surgimento de um conjunto de novas aplicações com mais e melhor interactividade e *user experience*. Envolve em grande polémica, a designação surge em 2004 numa conferência intitulada “*The web as Platform*” – A Web como plataforma. Tim O’Reilly [24] avança com a seguinte definição:

Web 2.0 is the business revolution in the computer industry caused by the move to the Internet as a platform, and an attempt to understand the rules for success on that new platform.

Efectivamente o surgimento da Web 2.0 fica associado ao surgimento de redes sociais online, partilha de vídeo, Wikis, Blogs e folksonomies¹².

Características de aplicações Web 2.0:

- As aplicações são disponibilizadas como um serviço e não como um pacote de software, diminuindo o custo e esforço de crescimento e escalabilidade (e.g. Google Documents vs. Microsoft Office).

¹² A **folksonomia** é uma maneira de indexar informações. Esta expressão foi cunhada por Thomas Vander Wal. É uma analogia à taxonomia, mas inclui o prefixo *folks*, palavra da língua inglesa que significa pessoas.

- Bases de dados de informação gerada pelos utilizadores directamente, impossível de obter por outras formas (e.g. hi5)
- Os utilizadores contribuem directamente no desenvolvimento e evolução das aplicações (e.g. código fonte disponível no Sourceforge.net).
- A aplicação cresce e é controlada pela própria comunidade que a utiliza (e.g. Wikipedia)
- Aplicações disponíveis para diversos dispositivos (e.g. Web, Desktop, telemóveis, PDA's)
- Interfaces com os utilizadores mais interactivas (e.g. GMail), modelos de desenvolvimento ágeis (e.g. XP, Scrum).

Modelos de negócio inovadores (e.g. utilização de plugins).

Web2.0 + Ajax

Do ponto de vista tecnológico, a evolução tecnológica por base da maior interactividade e melhor experiência de utilização presente nas aplicações Web 2.0, trata-se da arquitectura e conceito Ajax (*Asynchronous JavaScript and XML*). Não se trata de uma nova tecnologia, uma vez que se baseia em mecanismos disponíveis no browser desde 2001, mas sim de um novo conceito e arquitectura de desenvolvimento de AW.

O termo Ajax foi introduzido por Jesse James Garret em 2005 no seu largamente referenciado artigo “**Ajax: A New Approach to Web Applications**” [25]. Neste artigo Jesse James Garret, refere que a Web tradicional (Web 1.0) estava condicionada ao paradigma *request-response*. O novo conceito sugerido pelo artigo apresentava um paradigma alternativo¹³ que permitiria ultrapassar este constrangimento melhorando com isso a interactividade e experiência de utilização das aplicações Web (AW).

Fundamentos do conceito/arquitectura Ajax:

- Interfaces com utilizador desenvolvidas através da utilização de XHTML e CSS;
- Interacção e animações com base no DOM (Document Object Model);
- Transferência e manipulação de informação no formato XML;
- Processamento assíncrono com base em XMLHttpRequest;
- Programação em JavaScript para combinar as referidas tecnologias;

Com base neste conceito, algumas aplicações típicas de desktop, foram reintroduzidas como Aplicações Web (e.g. MS Outlook, A. Photoshop, MS. Word). Mais informação disponível no ponto 2.2.2.

Web 2.0 + Plug-ins

Para além das tecnologias já referidas, incorporadas no *Browser* e ,por isso, dependentes da respectiva implementação, existem também um conjunto de tecnologias que permitem ultrapassar determinadas limitações ou condicionantes inerentes existência do *Browser*. Estas tecnologias são designadas por **Plug-In**.

¹³ O paradigma de funcionamento do conceito Ajax será explorado com maior detalhe no Capítulo 3 Aplicações Ajax.

Os *Plug-in* com mais história na Web são:

- Java Applets¹⁴ (Sun Microsystems)
- Flash¹⁵ (Adobe)
- QuickTime¹⁶ (Apple)
- Real Player

Estes Plug-ins estendem as capacidades das aplicações Web permitindo:

- Transmissão de vídeo e áudio (e.g. YouTube e Last.fm);
- Animações sofisticadas;
- Interactividade avançada (e.g. jogos online);
- Capacidade de armazenamento de dados;
- Capacidade de processamento;
- Independência do browser em que são executados;

O primeiro Plug-in disponível na Web (1995) foi Java Applets. O intuito inicial da Java era possibilitar a disponibilização de aplicações Web (AW) com interactividade e capacidades semelhantes às aplicações Desktop (AD). Em 1996 foi lançado o Flash 1.0 pela Macromedia (adquirida pela Adobe em 2005).

Ainda que amplamente disseminadas pela Web actual, nem as JavaApplets nem o Flash, conseguiram destronar a disponibilização tradicional de informação com base em HTML e tecnologias associadas.

Um dos principais motivos indicados para tal, prende-se com o facto de ambos representarem fontes de informação fechadas, não permitindo por isso, a sua indexação por motores de busca. Na medida em que o ponto de partida, para uma grande maioria de utilizadores da Web, é um motor de busca, o facto de o conteúdo em Flash não ser visível para o motor de busca, reduz a visibilidade que esse conteúdo poderia ter.

*Não obstante as limitações apresentadas, quando a finalidade de uma aplicação passa pela transmissão de elementos multimédia, uma maior interactividade ou capacidade de armazenamento e processamento, inevitavelmente a solução passa pela utilização de um dos referidos **Plug-ins**.*

Web 3.0 [26][27][28]

Mais polémico do que o termo Web 2.0 é o termo Web 3.0. Inevitavelmente quando se tenta perceber o que virá a seguir (*“What’s the next big thing”*) as opiniões divergem e contrariam-se.

No entanto existem duas ideias que prevalecem sobre a próxima etapa de evolução da Web.

Web Semântica e Computação Ubíqua

¹⁴ Java Applet, Wikipedia, http://en.wikipedia.org/wiki/Java_applet , Acesso: 2009-03-10

¹⁵ Adobe Flash, Wikipedia, http://en.wikipedia.org/wiki/Adobe_flash , Acesso: 2009-03-10

¹⁶ Quick Time, Wikipedia, <http://en.wikipedia.org/wiki/Quicktime> , Acesso:2009-03-10

Resumidamente, o conceito de **Web semântica** está relacionado com a ideia de informação que descreve informação. Isto é, vamos afastar-nos das pesquisas por palavra-chave (*keywords*), deixando a Web de ser um mundo de documentos / recursos (identificados por um URI) para um mundo de informação. Não só as pessoas mas também os sistemas computacionais, poderão tirar partido desta maior semântica na informação.

Na base da Web semântica estão um conjunto de linguagens e tecnologias como:

- RDF (Resource Description Framework¹⁷): linguagem para representar a informação na Web
- OWL (Ontology Web Language)¹⁸: linguagem para definir e descrever ontologias na Web
- XML (eXtensible Markup Language)¹⁹: recomendação W3C para a expansão de linguagens de descrição de conteúdos.

Para além a *Web semântica*, a **Computação Ubíqua** é também indicada como uma das evoluções na Web 3.0.

Neste contexto a Computação Ubíqua está relacionada com a possibilidade de ser quebrada a barreira os mundos Desktop e Web (convergência total), passando as aplicações a estar disponíveis num servidor algures e o utilizador poder, através de um login em qualquer computador do mundo, ter acesso aos seus dados e aplicações.

Num cenário como o referido, os *web sites* passariam a *web services* e a troca de informação entre aplicações e pessoas, com base em formatos standard e independentes da aplicação, sistema operativo e dispositivo (computadores, telemóveis, electrodomésticos, dispositivos portáteis).

Aguardemos atentamente pelos sinais da evolução.

De facto, sendo o mundo Web algo tão recente (32 anos segundo Elliotte Rusty Harold²⁰) e em constante evolução e mutação, torna-se arriscado tentar perspectivar eventuais cenários para o seu futuro. Trata-se no entanto de um exercício interessante e pertinente, na medida em que permite captar as tendências e necessidades dos utilizadores da Web actual e adoptar medidas que permitam, às empresas e profissionais interessados, participar na discussão, desenvolver projectos experimentais e, mais importante, colaborar para uma melhor e mais funcional *World Wide Web*.

2.2.1.3 Ritch Internet Application (RIA)

O termo *Rich Internet Applications* (RIA) foi utilizado pela primeira vez por Jeremy Allaire, no seu trabalho de 2002 "*Macromedia Flash MX - A next-generation rich client*" [29]. Ainda que o trabalho tenha sido escrito à luz das intenções e capacidades do

¹⁷ RDF, Wikipedia, <http://pt.wikipedia.org/wiki/RDF> , Acesso: 2009-03-10

¹⁸ OWL, Wikipedia, <http://pt.wikipedia.org/wiki/Owl> , Acesso: 2009-03-10

¹⁹ XML, Wikipedia, <http://pt.wikipedia.org/wiki/XML> , Acesso: 2009-03-10

²⁰ Data de nascimento da Web: 22 de Novembro de 1977 (*Happy 30th Birthday Internet*, <http://cafe.elharo.com/internet/happy-30th-birthday-internet/>, Acesso:2009-03-10)

Flash, reflecte de uma forma eficaz as lacunas detectadas na Web tradicional que propiciaram a oportunidade / necessidade das RIA.

Quadro 5: Lacunas da web tradicional vs. Mais-valias das RIA

Lacunas da Web tradicional	Mais-valias das RIA
Impossibilidade de processamento de informação (lado do cliente), dependência do paradigma <i>request-response</i> para actualização e apresentação das páginas e impossibilidade de armazenamento de dados (lado do cliente);	<i>“Provide an efficient, high-performance runtime for executing code, content and communications.”</i> 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)	<i>“Integrate content, communications, and application interfaces into a common environment.”</i> Contexto integrado para conteúdos, comunicações e interfaces com aplicação
Falta de Object Oriented Design(ODD) para o desenvolvimento de aplicações de grande dimensão.	<i>“Provide powerful and extensible object models for interactivity.”</i> 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) componente e bibliotecas de componentes para facilitar e acelerar o desenvolvimento de aplicações mais complexas.	<i>“Enable rapid application development through components and re-use.”</i> Possibilitar a utilização de componentes previamente desenvolvidos e reutilização de código para acelerar o processo de desenvolvimento de aplicações.
Não apresentar uma separação clara entre: - Lógica de apresentação - Interface com o utilizador - Lógica da aplicação	<i>“Enable the use of web and data services provided by application servers.”</i> 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.	<i>“Embrace connected and disconnected clients.”</i> Abranger utilizadores com e sem ligação à Web.

De uma forma geral, os requisitos base para as RIA convergem para características chave de aplicações de Desktop (AD) que contribuem para a sua maior interactividade e melhor **experiência de utilização** (abordada com maior detalhe no ponto 2.3.1).

Segundo a Wikipedia [31], uma possível definição de RIA poderia ser:

[...] Web applications that have the features and functionality of traditional desktop applications. RIAs typically transfer the processing necessary for the user interface to the Web client but keep the bulk of the data (i.e., maintaining the state of the program, the data etc) back on the application server. [...]

Propositadamente ou não, a definição de RIA apresentada não contempla os Web Browsers. De facto as opiniões divergem, se por um lado existem aqueles que defendem que as RIA devem proporcionar, no contexto do browser, as características disponíveis, outros defendem que a disponibilização destas características só será possível se adoptarmos uma orientação ao Desktop. Surgem por isso (conforme é possível verificar na Figura 1: RIA: Categorização de acordo com a origem do processamento) as RIA's orientadas à Web e ao Desktop.

Um outro factor preponderante na ascensão das RIA será o aumento da largura de banda e aumento da capacidade de processamento dos computadores (cliente e servidor). Estrategicamente ou não temos verificado essa tendência em ambas as industriais: Telecomunicações e Hardware. Actualmente temos Prestadores de Serviço de Internet (ISP - *Internet Service Providers*) a disponibilizar acessos a 100MB (www.zon.pt), assim como, a capacidade de processamento e armazenamento de informação cresceu de uma forma bastante significativa.

A Web e tudo o que com ela se relaciona estão em constante evolução, causando por vezes um sufoco, uma vez que não permite que ninguém fique à espera na retaguarda pelo que a seguir virá. Todos querem fazer parte da próxima *onda*.

2.2.2 Migração de Desktop para Web

À parte de tudo o que se possa escrever sobre as vantagens e mais-valias de cada mundo Web e Desktop, um facto consumado é que começa a existir um conjunto de aplicações que se propuseram a percorrer e encurtar a distância entre ambos os mundos. Um exemplo bastante popular é o Goolge Docs²¹.

O Google Docs trata-se de um conjunto de aplicações disponibilizadas pela Google, desenvolvidas com base em Ajax. As aplicações estão apenas disponíveis online sendo necessária a utilização de um Web Browser para as poder utilizar.

Algumas das aplicações disponíveis são um processador de texto, um editor de apresentações, e folhas de cálculo. Um dos aspectos relevantes é a portabilidade dos documentos gerados e a capacidade de exportação dessa informação em diferentes formatos (e.g. .PDF, XML, etc).

Seguem-se mais alguns exemplos de aplicações [30] que tentam combinar vantagens de aplicações Web e Desktop:

²¹ Goolge Docs, <http://docs.google.com>

- gOffice, Web Word Wizard (<http://www.goffice.com>);
- Zoho Writer, Online Word Processor (<http://writer.zoho.com>);
- eyeOS, Cloud Computing's Operating System (<http://www.eyeos.org/>);

2.2.2.1 Porquê aplicações Web?

Porque é que apesar das aplicações Desktop corresponderem ao propósito para que foram inicialmente pensadas, continuamos a assistir ao fenómeno de migração destas para versões Web?

Em seguida são apresentados alguns motivos e motivações que podem estar na base do referido fenómeno:

- Uma grande parte dos utilizadores, no dia-a-dia utiliza diversos computadores (casa, trabalho, universidade, telemóvel, *cyber-café*, etc.), de forma que, a solução mais natural para ter sempre disponível a sua informação e manter as suas preferências de utilização, passe por esta informação ser armazenada online.
- Mais grave do que ter vários computadores é não ter nenhum. Sem um computador pessoal os utilizadores não podem instalar aplicações desktop e armazenar a sua informação. Nesta situação, as aplicações Web representam uma alternativa eficaz (podem sempre pedir um computador emprestado);
- Uma aplicação Desktop necessita inevitavelmente de ser instalada. Este procedimento pode intimidar, pelo menos faz pensar duas vezes, um utilizador antes de avançar com a instalação. Ao contrário, uma aplicação não necessita de qualquer instalação e pode começar logo a utilizar a mesma (eventualmente pode existir a necessidade de um registo prévio);
- Nos dias que correm a largura de banda e capacidade de armazenamento e processamento dos servidores centrais, possibilitam a implementação de infra-estruturas de serviços bastante eficazes. Ao contrário do que se pode imaginar, esta centralização de recursos pode traduzir-se na redução de alguns custos (e.g. manutenção e actualização de hardware das máquina clientes);
- A arquitectura Ajax representa um forte empurrão na forma de pensar, desenvolver e utilizar aplicações Web. Com o amadurecimento deste conceito, e tecnologias inerentes, a sua utilização permite atingir novos níveis de performance e interactividade. Isto é, a evolução surge apenas devido ao aumento da largura de banda, mas também, pela inovação inerente à forma como são desenvolvidas as aplicações Web.
- Para os programadores e empresas de desenvolvimento de software, o lançamento de uma aplicação Web é menos dispendioso. Não necessita de embalagem (pelo menos física), não existem preocupações relacionadas com o desenvolvimento para diferentes plataformas (e.g. Windows, Mac, Linux), logo os programadores podem dedicar-se apenas a um único código fonte, a

actualização procede-se de uma forma centralizada (todos os utilizadores ao mesmo tempo) evitando a preocupação da compatibilidade com versões anteriores e, por fim, uma vez que alógica aplicacional correr no servidor, podem utilizar a linguagem de programação que pretenderem, sem que isso tenha impacto para o utilizador final.

- Ao contrário de o que acontecia há alguns anos atrás, o desenvolvimento de aplicações Web está bastante mais simplificado e acessível. Para a simplificação do desenvolvimento contribuíram factores como:
 - Profissionais com mais e melhores competências e conhecimentos sobre a arquitectura Web;
 - IDE (Integrated Development Environment), que simplificam e agilizam a escrita e teste de código fonte (inclusivamente os próprios browsers apresentam funcionalidades para *debugging*²²);
 - Componentes e bibliotecas que permitem a reutilização de código, tornando o processo de desenvolvimento mais rápido (especialmente após o aparecimento do Ajax);
 - Ainda que existam algumas diferenças, os browsers tendem a respeitar cada vez mais os standards, facilitando o desenvolvimento de aplicações *cross-browsing*.
- Outro factor reside no fenómeno da segurança. Algumas empresas consideram que as determinadas aplicações na sua *Intranet*²³ permitem um maior nível de controlo, monitorização e auditoria (e.g. acessos, que realizou determinadas acções, quem consultou determinada informação).
- No desenho e desenvolvimento de uma aplicação são tomadas decisões no sentido de facilitar o acesso, por parte dos utilizadores, a uma aplicação. Nesse sentido, o desenvolvimento de aplicações Web reduz tremendamente a complexidade do desenvolvimento para diferentes dispositivos (e.g. computadores, PDA's, telemóveis) e sistemas operativos.
- Uma das grandes mais-valias da Web é permitir a comunicação com utilizadores geograficamente dispersos. No entanto a comunicação entre eles requer a utilização de software compatível. Como os Web browsers são implementados com base em standards e estão disponíveis para todas as plataformas, dispositivos e sistemas operativos, a sua utilização facilita o desenvolvimento de aplicações de cariz colaborativo e partilha.

²² *Debugging*: processo de encontrar e corrigir os erros/falhas presentes numa aplicação de software

²³ *Intranet*: rede de computadores privada (tipicamente refere-se à infra-estrutura de computadores e servidores de uma empresa ou organização)

Ainda que alguns dos motivos e motivações apresentados possam lançar alguma discussão, é incontornável o facto de existirem actualmente diversas mais-valias no desenvolvimento de aplicações Web. Sem nunca esquecer ou abdicar das aplicações Desktop (AD) em contextos e cenários que o justifiquem, o desenvolvimento e migração das mesmas para a Web apresenta-se um grande atractivo para a indústria de Engenharia de Software.

Não se trata apenas de uma moda, mas sim de uma tendência que modelou as evoluções ocorridas no passado e que continuam a acontecer.

2.3 Comparação entre IU Web e Desktop

Como vimos nos pontos anteriores, uma RIA pode ser mais orientada à Web ou ao Desktop. Simon Morris apresenta, na sua maneira crítica e divertida, a três futuros possíveis (“religiões” como ele lhes chama) para as RIA: *Browserism*, *Neo-Desktopism* e *Pragmatic Neo-Desktopism* [32]. (Como ele próprio refere, os nomes são apenas sugestões e caso alguém encontre termos mais adequados não hesite em sugerir.)

- a) No ***Browserism***, defende-se que os Web Browsers com recurso a HTML a tecnologias como o Ajax, serão o futuro do software com interface para o utilizador. Segundo Morris, esta visão falha uma vez que as tecnologias base não foram pensadas para este tipo de utilização. Vai mesmo mais longe ao afirmar que contrariam o propósito definido por Tim Berners-Lee na origem da WWW.
- b) Por outro lado temos o ***Neo-Desktopism***, uma visão oposta à anterior *Browserism*. Nesta visão defende-se a ideia da evolução das aplicações Desktop tradicionais (que requerem instalação) para um novo conceito onde estas possam ser independentes de uma instalação na máquina local do cliente (e.g. Java Web Start). Desta forma seria possível iniciar a aplicação através do browser, esta seria descarregada e inicializada sem necessitar de instalação na máquina do cliente. Na sua perspectiva, esta visão falha devido à falta da ubiquidade (disponibilidade e transversalidade) existente num Web Browser.
- c) Por fim existe ainda uma terceira visão - ***Pragmatic Neo-Desktopism***. Nesta vertente defende-se que o Web Browser será de facto o standard da ubiquidade no que toca à interacção como utilizador, no entanto, devido às limitações inerentes à utilização do browser, serão necessárias tecnologias como Java Applets e Adobe Flash para possibilitar o desenvolvimento e disponibilização de RIA's para os utilizadores.

A análise apresentada por Morris trata-se de uma visão pessoal, baseada em exemplos é um facto, mas não deixa de ser apenas uma opinião do autor. No entanto, revela, na minha opinião, um aspecto fundamental. O facto de pensarmos o futuro à

base das tecnologias actuais será sempre um problema. Inevitavelmente serão dados novos e inovadores passos na convergência entre os dois mundos, ainda que, a perspectiva mais segura para o futuro resida no respeito pelos standards e ubiquidade.

Nos pontos seguintes iremos comparar as aplicações Web e Desktop em diferentes aspectos e indicar tendências e perspectivas de convergência. Sempre que possível serão apresentados exemplos dos conceitos descritos.

2.3.1 Experiência de Utilização

Historicamente as aplicações Web (AW) e Desktop (AD) diferem no nível de experiência de utilização que propiciam aos seus utilizadores. As RIA surgem com a intenção de usufruir com o melhor de ambos os mundos, Web e Desktop, de modo que deveriam ser de fácil acesso como um AW e oferecer a interactividade e experiência de utilização de um AD.

A experiência de utilização de um sistema pode ser definida como a sensação de satisfação (obtenção dos resultados pretendidos) resultante da eficácia e eficiência das interacções de um utilizador com um determinado sistema, produto ou serviço.

A experiência de utilização de um sistema representa um dos aspectos a ter em conta no que toca a Interacção Humano Computador (*Human Computer Interaction*). Na base de uma experiência de utilização estão os conceitos de Interactividade e *Responsiveness* (termo para o qual não encontrei tradução).

Interactividade, descreve a forma e mecanismos disponíveis para um utilizador, no sentido de permitir a entrada de instruções e respectiva comunicação do resultado obtido.

Responsiveness, descreve a rapidez e eficácia com que um sistema interactivo responde a uma instrução do utilizador (e.g. tempo de resposta de um browser após ser introduzido o endereço de uma página Web que pretendemos consultar).

Um dos factores preponderantes na janela de oportunidade para as RIA's é de facto a perspectiva e necessidade de uma melhorada experiência de utilização.

Web

Com base na tradicional utilização do HTML a experiência de utilização é bastante reduzida. A interactividade e *responsiveness* da aplicação estão limitadas ao paradigma de *request-response*, permitindo apenas feedback para o utilizador após a actualização dos conteúdos da página. Validações em campos de formulários acontecem apenas após submissão do formulário ou na página seguinte. Importa não esquecer que a Web foi pensada para a troca e ligação de conteúdos e não para a disponibilização de aplicações interactivas.

Desktop

Ao contrário do mundo Web, o mundo Desktop beneficia da alargada Interactividade e Responsiveness disponível nas AD. Formulários avaliados à medida que são preenchidos, alertas em tempo real que previnem o surgimento de erros ou disponibilização de ajudas activas. As acções do utilizador podem ser monitorizadas com maior pormenor, permitindo que a GUI seja actualizada com base em eventos diversos, dos quais alguns podem não ser directamente originados pelos utilizadores. As AD podem funcionar em dois modos *on* e *offline*, comunicando com o servidor apenas quando uma ligação em rede está disponível (e.g. Cliente de email).

A integração das várias camadas da aplicação: a) Camada de apresentação, b) Lógica Aplicacional e c) fonte de dados; é mais simplificada uma vez que se encontram todas na mesma máquina, não necessitando de percorrer a “distância” inerente a um servidor remoto.

Convergência

Com o aparecimento do Ajax alguns dos constrangimentos inerentes ao mundo Web têm vindo a ser ultrapassadas. Como o Ajax surgiu a possibilidade de tornar invisível para o utilizador o processo de *request.response* de conteúdos do servidor. Surge também a possibilidade de actualizar de forma assíncrona os conteúdos de uma página, bem como, a possibilidade de lançar alertas e avisos à medida que o utilizador vai interagindo com os elementos presentes na página Web. O Ajax tornou ainda possível a utilização de animações e paradigmas de interacção mais sofisticados como *drag-and-drop*²⁴.

Não obstante os novos horizontes possíveis através da utilização da arquitectura Ajax, as aplicações Web continuam residentes um Web Browser, estando por isso restrita a certas componentes de interactividade e *responsiveness* apenas disponíveis para AD. Como referido anteriormente, tecnologias como o Flash permitem superar algumas destas restrições: armazenamento de dados, comunicação de áudio e vídeo, comunicações em tempo real.

À luz do fenómeno evolutivo que verificamos nas aplicações Web podemos assumir que estas estão a convergir ou, pelo menos, adoptar diversas características da experiência de utilização típica de AD. Todavia, consideramos que a fusão entre aplicações Web e aplicações Desktop não será concretizada na sua plenitude. Acreditamos que com as próximas inovações tecnológicas as fronteiras, no que toca à experiência de utilização, se irão esbater mas nunca substituir por completo tendo em conta os pressupostos base na origem de ambos os mundos, Web e Desktop.

²⁴ *Drag-and-drop*, paradigma de interacção que permite ao utilizador seleccionar um elemento e arrasta-lo pela área de interacção disponível.

2.3.2 Conteúdos e Informação

Independentemente da sua orientação à Web ou ao desktop, na base de todas as aplicações estão os conteúdos, a informação e o conhecimento daí inerente. Em 2007, segundo um estudo da IDC patrocinado pela EMC [33], o Universo de Informação Digital era composto por - $2,25 \times 10^{21}$ bits (o equivalente a 281 *exabytes*²⁵ ou 281 biliões de *gigabytes*), 10% maior do que o previsto (ver figura Figura 5). Este estudo avança também com uma previsão para 2011, onde se estive que o mesmo universo ira crescer cerca de 10 vezes.

O mesmo estudo indica que, conforme previsto, que a produção de conteúdos ultrapassou a capacidade de armazenamento do mesmo, sendo que este fenómeno se tende a alargar até 2011, aumentando a quantidade de informação produzida mas que acaba por não ser armazenada. O crescimento a nível da produção de conteúdos está relacionado com: TV Digital, câmaras de vídeo vigilância, novos acessos à Web em países em desenvolvimento, aplicações baseadas em sensores (RFID), *data-centers* para suporte de computação ubíqua (computação em nuvem) e as, tão populares, redes sociais online. Todas estas emergentes plataformas contribuem para a produção e partilha de informação no Universo Digital.

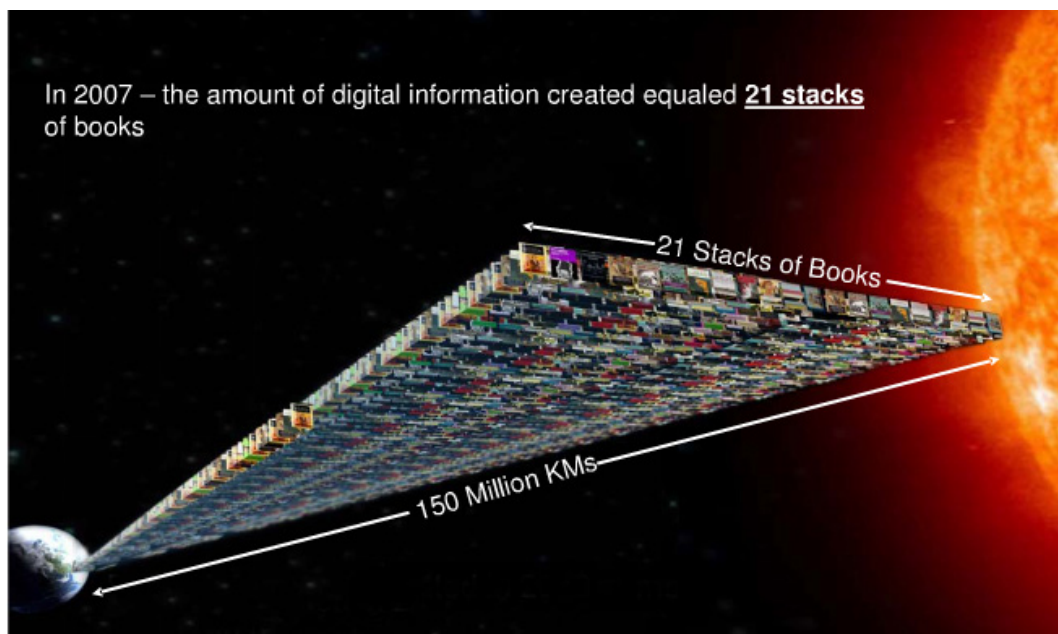


Figura 5: Quantidade de informação digital produzida em 2007 [35]

Os tópicos seguintes tentam captar as estratégias adoptadas na Web e no Desktop para proporcionar aos utilizadores um maior controlo, conforto e proveito na pesquisa e utilização dos conteúdos e informação disponível.

²⁵ Exabyte: Wikipedia, <http://en.wikipedia.org/wiki/Exabyte>, Acesso: 2009-03-11

2.3.2.1 Web Semântica e Desktop Semântico

Pela introdução do tópico actual é possível compreender o conceito de Web Semântica perspectivado para a próxima geração desta plataforma. Por outro lado, a mesma necessidade urge do lado do Desktop para a qual já foram dados os primeiros passos em direcção ao Desktop Semântico.

WEB

Como tivemos oportunidade de verificar na análise sobre as próximas evoluções para a plataforma Web (Web 3.0), o conceito de Web Semântica²⁶ não se trata apenas de mais um termo que está na moda, mas sim uma necessidade cada vez mais presente. O universo de informação digital está em expansão a grande velocidade e sem os mecanismos previstos pela Web Semântica não será possível tirar partido da quantidade de informação disponível, pelo contrário, caminharemos em direcção ao caos no universo (do caos nasce a ordem²⁷).

Um dos factores chave na Web Semântica é a capacidade das máquinas conseguirem interpretar, associar e indexar melhor a informação por nós gerada, de forma a acelerar o processo de pesquisa da mesma, paralelamente com o aumento da qualidade dos resultados apresentados (qualidade neste contexto significa, grau de correspondência com a informação pretendida pelo utilizador).

As bases das Interfaces com o Utilizador para esta Web Semântica estão lançadas (ver Figura 6).



Figura 6: IU da Google para pesquisa de informação

O sucesso de um motor de busca, neste caso o Google, reside na forma simplificada como podemos indicar a informação que pretendemos consultar. Logo, o grande desafio para a evolução em questão, não passará tanto pela evolução da IU mas sim na forma como as fontes de dados estão relacionadas e na qualidade da descrição de cada conteúdo disponível na Web.

²⁶ Semantic Web Enabled Web Services, <http://sws.semanticweb.org/>, Acesso: 2009-02-11

²⁷ Teoria do caos (Chaos Theory), Wikipedia, http://en.wikipedia.org/wiki/Chaos_theory, Acesso: 2009-03-11

Desktop

O conceito de Desktop Semântico não passa de uma extensão do conceito de Web semântica, para o mundo Desktop^{28,29}. Neste cenário, a fonte de dados de conteúdos a pesquisar apresenta especial enfoque na informação residente na máquina local do utilizador. Consideremos a proposta de definição avançada por Leo Sauermann, no seu trabalho “*Overview and Outlook on the Semantic Desktop*” [34]:

“(...) The Semantic Desktop is an enlarged supplement to the user’s memory.”

Mais uma vez, devido ao aumento da quantidade de informação produzida e armazenada, a possibilidade de melhor descrever os conteúdos produzidos, permitirá uma melhor pesquisa e consulta da mesma à posteriori.

Ao contrário da Web, o Desktop Semântico apresenta o desafio extra relacionado com o facto de existirem diferentes Sistemas Operativos (SO) e com eles diferentes formas e estratégias de catalogação e organização de conteúdos.

Convergência

Os conceitos de Web Semântica e Desktop Semântico surgem como estratégias de futuro para gestão de informação e conhecimento. Nesse sentido a ideia chave, de acordo com Leo Sauermann [54], passa por adaptar as tecnologias, que possibilitam a Web Semântica, para Desktop permitindo que cada utilizador consiga utilizar o seu computador como a sua Web Semântica pessoal. Como desafio, em prole deste objectivo, Sauermann indica a necessidade de interfaces para programação de aplicações (API – *Application Programming Interfaces*) standard que permitem a criação de *Frameworks* e serviços que disponibilizem esta funcionalidade ao utilizador.

2.3.2.2 Pesquisa de informação

A funcionalidade “Pesquisar” é uma das mais populares em aplicações Web e Desktop, devido ao facto de um utilizador não conseguir memorizar os URI (caminhos para a informação) de toda a informação que pretende. Por outro lado existe um conjunto imenso de informação que o utilizador desconhece à partida, logo nunca poderia conhecer o respectivo URI.

Num cenário em que o Universo de Informação Digital (pessoal e universal) está em constante crescimento, a forma mais fácil e eficiente para encontrar informação é através da funcionalidade “pesquisar.”

Web

Tipicamente quando um utilizador pretende procurar informação na Web recorre a um motor de busca³⁰ (e.g. Google, Live Search, Yahoo Search, Sogou). Desta forma a

²⁸ Desktop Semântico, Wikipedia, http://en.wikipedia.org/wiki/Semantic_desktop , Acesso: 2009-03-11

²⁹ Página oficial Desktop Semântico <http://www.semanticdesktop.org>, Acesso: 2009-03-11

³⁰ Anexo A – Lista de motores de busca

pesquisa realizada incidirá sobre um conjunto alargado de páginas e informação disponível na Web. A qualidade dos resultados apresentados ao utilizador pode depender de dois aspectos: as palavras-chave utilizadas ou a qualidade da descrição existente num determinado conteúdo.

No passado, a pesquisa de largos repositórios de informação era uma tarefa árdua e apenas passível de ser realizada por profissionais experientes com domínio de linguagens de interrogação de fontes de dados. Actualmente, as Interfaces para o Utilizador (IU) da funcionalidade de pesquisa encontram-se simplificadas (ver), estando por isso disponíveis para novos utilizadores e utilizadores mais experientes (pesquisa avançada).

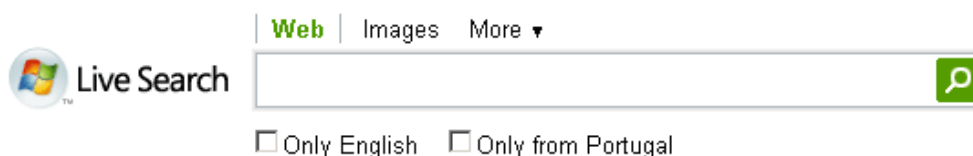


Figura 7: IU - Live Search: pesquisar informação



Figura 8: IU - Yahoo Search: pesquisar informação

Desktop

Ainda que para Desktop a funcionalidade de “Pesquisar” esteja igualmente disseminada, o facto de a pesquisa incidir sobre o nome de pastas e ficheiros é manifestamente limitador (e.g. Windows XP, ver Figura 9). Ao longo do tempo foram introduzidas melhorias permitindo ao utilizador indicar um conjunto de caracteres, referindo que podem existir outros antes ou depois, no entanto não se compara a facilidade de utilização da Pesquisa na Web.

Mais uma vez, a temática do Sistema Operativo entra na discussão, devido ao facto de nos diferentes sistemas operativos a funcionalidade de “Pesquisar” apresentar diferentes níveis de maturidade. No caso do Leopard (sistema operativo da Apple), a funcionalidade de pesquisa (Spotlight), para além do nome dos ficheiros (em diferentes formatos), acede ao conteúdo dos mesmos, incluindo emails, contactos e eventos em calendário e apresenta um mais completo conjunto de resultados.

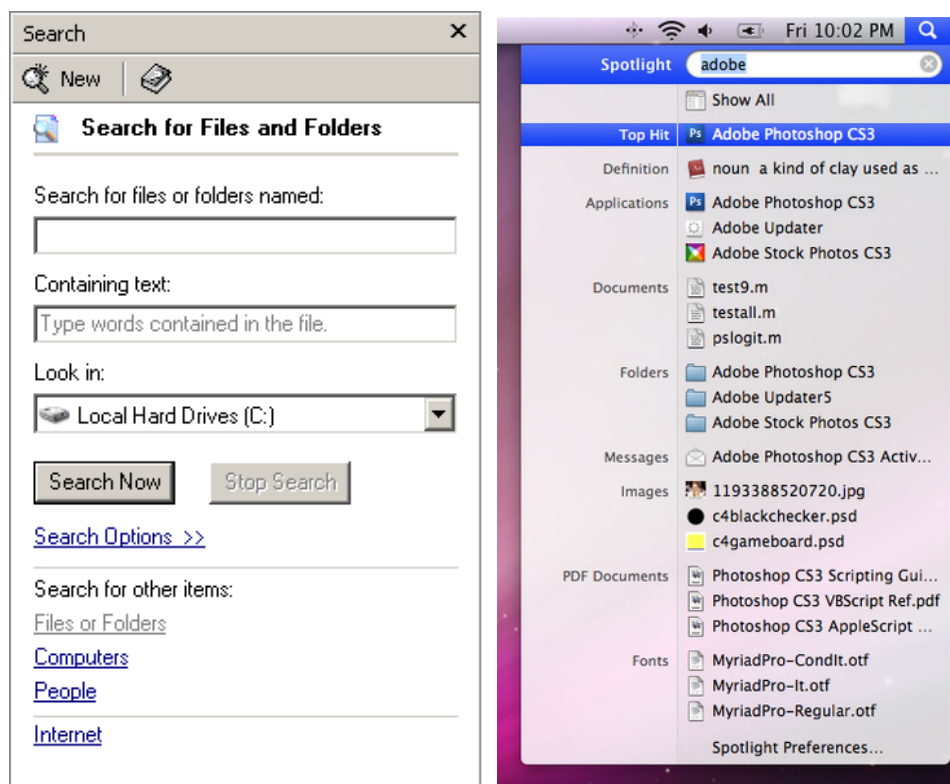


Figura 9 e 10 : IU – Windows XP: Pesquisar (esquerda). IU – Leopard Spotlight: pesquisar informação (direita)

Convergência

No aspecto da pesquisa de informação, a Web apresenta soluções mais evoluídas e interessantes para o utilizador. A convergência e adopção, para Desktop, destes paradigmas de sucesso começam a ganhar expressão com introdução de aplicações inovadoras como o “Google Desktop Search” (ver Figura 10) e o “Windows XP Search Companion”.

Efectivamente, os grandes motores de busca disponibilizam actualmente aplicativos³¹ que podem ser descarregados e instalados na máquina do utilizador, que permitem melhorar tremendamente o processo de pesquisa de informação na Web privada (máquina local) do utilizador.

Tendo ainda em perspectiva a análise sobre Web Semântica e Desktop Semântico, a funcionalidade “Pesquisar” irá beneficiar com as inovações introduzidas pelos referidos conceitos. O utilizador não precisará de preocupar com a localização da informação mas sim na essência da informação que pretende consultar, isto é, ser-lhe-á possibilitada a capacidade de “encontrar uma agulha no palheiro”.

³¹ Lista completa de aplicações de pesquisa para Desktop:
http://en.wikipedia.org/wiki/List_of_search_engines#Desktop_search_engines , Acesso: 2009-03-11

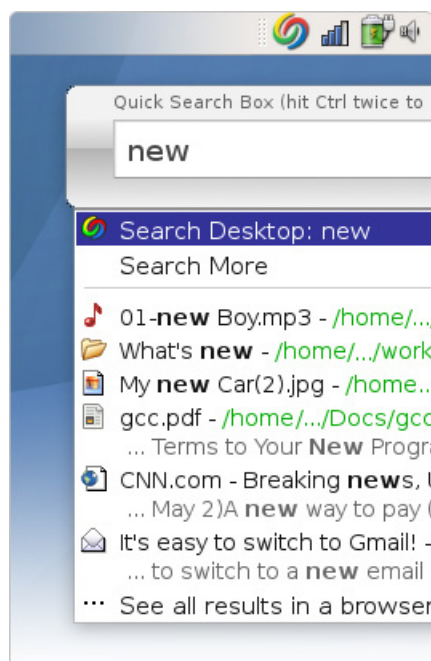


Figura 10: IU – Google Desktop (Red Hat Linux)

2.3.2.3 Controlo da informação

Inicialmente as aplicações Desktop (AD) armazenavam toda a informação localmente, não havendo a possibilidade de qualquer acesso remoto. A conectividade inerente à Web tornou possível o armazenamento dessa informação em servidores remotos geograficamente dispersos (e.g. países ou continentes diferentes do local de acesso).

A vantagem óbvia do armazenamento remoto consiste na disponibilidade acesso à informação em qualquer ponto do mundo em que existe uma ligação à Web (e.g. caixa de correio electrónico).

Por outro lado, o armazenamento remoto de informação implica alguns riscos como a perda de privacidade da informação, vulnerabilidade a ataques de hackers³² e abuso na utilização da mesma, por parte da empresas que detêm os servidores, para fins que não os originalmente contemplados com os utilizadores.

Web

De acordo com a sua arquitectura base, a informação deverá estar em servidores remotos de alta disponibilidade (24x7x52), de modo a facilitar a partilha e consulta de informação.

O que os pressupostos iniciais da Web não previam era a troca de informação de carácter privado ou confidencial. O pressuposto chave era a partilha de informação.

³² Hacker: pessoal que acede ilegalmente a um computador, Wikipedia, [http://en.wikipedia.org/wiki/Hacker_\(computer_security\)](http://en.wikipedia.org/wiki/Hacker_(computer_security)), Acesso:2009-03-11

Actualmente esse pressuposto mantêm-se actual, ainda que, devido à natureza pessoal e profissional de determinado tipo de aplicações Web, algumas empresas passam a ter acesso a informação pessoal e privada (e.g. fotografias, amigos, agenda de compromissos).

Desktop

O fenómeno do armazenamento de informação remota, em aplicações remotas é menos usual. Podemos mais uma vez considerar o exemplo de cliente de email (e.g. Mozilla Thunderbird) que apresenta dois modos de funcionamento – on e offline. No modo online, liga-se ao servidor e interroga-o sobre se tem informação nova informação. Se existirem novos emails estes são descarregados e, caso têm sido configurados para o efeito (IMAP³³), o email fica igualmente no servidor de email.

Convergência

Tim O'Reilly refere, numa entrevista à revista *Wired* – “*Web 2.0 Is About Controlling Data*” [36], que, ao contrário do que se possa pensar, o principal aspecto da Web 2.0 não são os wikis, blogs ou redes sociais, mas sim **controlo da informação gerada**.

Informação essa gerada por milhares de utilizadores que partilham: fotos, vídeos, opiniões, emails, conversas, documentos e agendas; armazenados nos servidores de empresas de sucesso com margem para utilizar essa informação para os fins que entenderem. Ainda que existam os **termos e condições** de utilização de um determinado serviço, é certo que uma grande parte dos utilizadores não lhes presta a devida atenção.

A evolução em curso aponta o caminho da computação ubíqua ou em nuvem. Nessa direcção não há perspectiva de retorno para armazenamento local. De acordo com o conceito de computação ubíqua, o utilizador deverá poder aceder de qualquer ponto à sua informação. Num cenário como o indicado, o armazenamento remoto de toda a informação é uma possibilidade bastante provável. Consideremos os AD que possibilitam conversação em tempo real (IM – *Instant Messaging*)³⁴. Em qualquer ponto do mundo, podemos iniciar o cliente com os nossos dados de utilizador e passamos a ter disponível o nosso grupo de amigos e colegas com os quais podemos estabelecer comunicações em tempo real.

A chave para o problema da utilização abusiva de informação dos utilizadores, passará principalmente pela criação de legislação que garanta o direito à confidencialidade da informação e regule a utilização fontes de dados produzidas por uma comunidade de utilizadores.

³³ IMAP, Internet message Access Protocol, Wikipedia, <http://en.wikipedia.org/wiki/IMAP4#IMAP> , Acesso:2009-03-11

³⁴ IM, comunicações em tempo real entre dois ou mais utilizadores, Wikipedia, http://en.wikipedia.org/wiki/Instant_messaging, Acesso: 2009-03-11

2.3.2.4 SEO e Acessibilidade

Curiosamente, no ponto anterior abordamos a confidencialidade e utilização abusiva de informação dos utilizadores. No ponto actual iremos abordar as preocupações inerentes à disponibilização de informação na Web, no que respeito diz, à visibilidade da mesma por motores de busca (SEO – *Search Engine Optimization*)³⁵ e utilizadores com pessoas com deficiência (Acessibilidade).

SEO

Um dos aspectos fundamentais para o sucesso de uma presença na Web é a frequência com que esta aparece nos resultados de um motor de busca. Para tal, os programadores devem garantir um conjunto de técnicas que garantam que a informação se encontra disponível num formato perceptível para que os motores de busca a possam analisar e indexar.

Com o surgimento das RIA's e novas tecnologias tais como JavaApplets, Flash e Ajax, a informação passa a não estar tão facilmente disponível para consulta pelos motores de busca. Os motores de busca não conseguem chegar à informação principalmente por dois motivos:

- Aplicações de código fechado (e.g. JavaApplets e Flash);
- A informação simplesmente não existe nas páginas, é carregada assincronamente a pedido do utilizador (Ajax);

Ainda que em 2006 Bob Garner, no seu artigo de opinião [37], tenha indicado que por parte da Google este não seria um motivo de preocupação, uma vez que consideram que a grande maioria da informação na Web está sob a forma de páginas HTML estáticas.

“The vast majority of sites are still built as static Web pages, so we don't foresee a problem at this time. The nice thing is that people building RIA/AJAX sites tend to have a technical skill set, and thus at least consider the impact of search engine crawlability.”

As declarações apresentadas referem-se a 2006. Entretanto informação e tecnologias utilizadas na Web têm evoluído de tal forma que os motores de busca apresentam agora novas técnicas e estratégias para melhorar a indexação de RIA's.

Um exemplo muito simples, apresentado no Google Webmaster Central Blog³⁶, descreve a solução Hijax³⁷ para um problema típico de aplicações desenvolvidas em Ajax.

³⁵ Search Engine Optimization, Wikipedia, http://en.wikipedia.org/wiki/Search_engine_optimization, Acesso: 2009-03-11

³⁶ A spider's view of Web 2.0, <http://googlewebmastercentral.blogspot.com/2007/11/spiders-view-of-web-20.html>, Acesso: 2009-03-11

³⁷ Hijax: Progressive Enhancement with Ajax, <http://domscripting.com/presentations/xtech2006/>, Acesso: 2009-03-11

Acessibilidade³⁸

Ainda que a Acessibilidade de uma aplicação possa beneficiar com as opções de desenvolvimento tomadas tendo em vista a SEO, temos de ter presente o seguinte. SEO prende-se com técnicas e mecanismos que tornam a informação facilmente acessível para máquinas, enquanto a Acessibilidade promove boas práticas para que essa mesma informação esteja disponível para utilizadores que consultam as páginas através de ferramentas próprias para pessoas com deficiência (e.g. *Screen Readers*³⁹).

Por outro lado existem aplicações Web que não devem estar acessíveis aos motores de busca mas sim para pessoas com deficiência. Estamos a falar de aplicações que querem um login para acesso a uma área reservada (e.g. *home banking*, clientes de email, redes sociais)

Tradicionalmente as aplicações Desktop apresentam um bom suporte para Acessibilidade, assim como as tradicionais páginas com base em HTML. Os desafios colocam-se mais uma vez com o surgimento de novas tecnologias que fogem aos paradigmas tradicionais de transmissão e disponibilização de informação.

2.3.3 Paradigmas de Interacção

Actualmente os utilizadores deparam-se com diferentes paradigmas de interacção quando transitam entre aplicações Web e Desktop. Segundo Matthias Muller-Prove [38] um paradigma de interacção pode ser definido da seguinte forma:

“An interaction model is used to describe how the user’s actions correspond to changes in the system model and vice versa. It is the continuous flow of information between the user and the machine that constitutes the interaction with the PC. (...)”

Como a definição indica, um paradigma de interacção tenta descrever a forma como um utilizador dá instruções ao sistema e a resposta do mesmo a essas acções. A interacção caracteriza-se então por um fluxo contínuo de informação entre o utilizador e a máquina.

Neste ponto serão descritos alguns dos principais paradigmas e modelos de interacção presentes em AW e AD, bem como, identificadas as principais inconsistências entre as implementações desses paradigmas em ambos os mundos.

2.3.3.1 Clique único vs. Duplo clique

Em aplicações desktop o clique único serve para seleccionar um icon ou posicionar o cursor de selecção de texto. Noutras situações, o clique único poderá também servir para navegar num menu ou carregar em botões. Em 1983 o duplo clique é introduzido

³⁸ Página oficial sobre acessibilidade online em Portugal <http://www.acessibilidade.net/>, Acesso: 2009-03-11

³⁹ Screen Reader: software que tenta identificar e interpretar a informação presente num ecrã (...), Wikipedia, http://en.wikipedia.org/wiki/Screen_reader, Acesso: 2009-03-11

nas aplicações Desktop (AD), através Apple Lisa Desktop, e é utilizado principalmente como comando de execução de instruções.

Na Web a interacção toda a interacção é iniciada com um clique único. A instrução de execução é conseguida através de um clique único em links nos quais basta um clique para despoletar a acção. Um link está tradicionalmente identificado por um sublinhado e cor azul, mudando para roxo após a ter sido consultado pelo utilizador.

É no entanto possível identificar uma tendência de convergência no que toca à utilização do clique único, juntamente com o conceito de link, nas aplicações de Desktop. O “Explorador de Ficheiros” do Windows XP (ver Figura 11) permite ao utilizador definir o paradigma de interacção pretendido: duplo clique ou clique único.

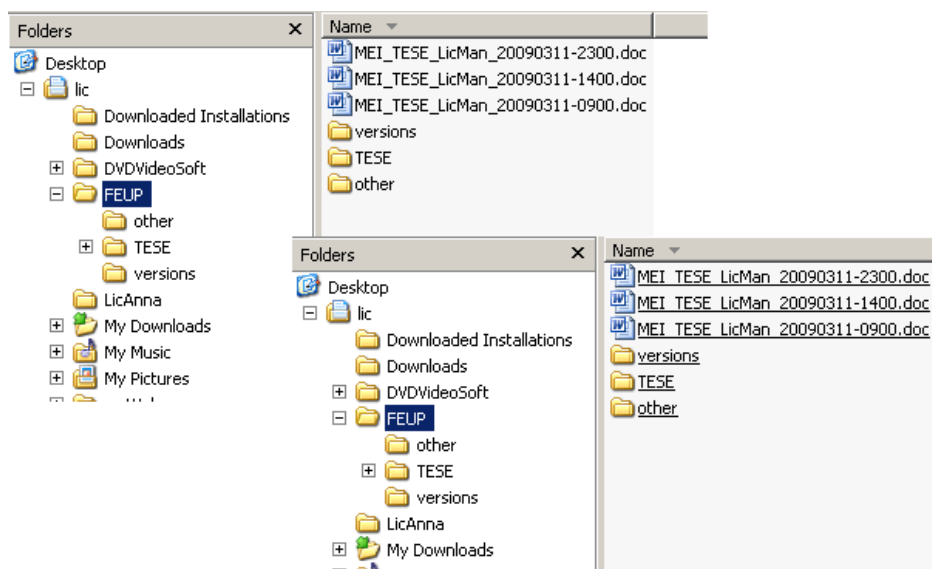


Figura 11: Explorador de Ficheiros (Windows XP) Clique único vs. duplo clique

A adaptação deste paradigma de interacção para Desktop está longe de ter sido conseguida na plenitude. No ecrã mais á direita presente na Figura 11, ainda que o utilizador tenha configurado a interacção para clique único, as diferentes áreas do ecrã apresentam diferentes paradigmas. A parte mais à esquerda utiliza apenas apresenta o sublinhado quando o rato passa por cima do ficheiro ou pasta, enquanto na parte direita do ecrã todos os recursos aparecem assinalados com um sublinhado.

Estes mecanismos inconsistentes prejudicam usabilidade do sistema pela confusão e ambiguidade que lançam na experiência de utilização.

2.3.3.2 Selecção de texto

Tradicionalmente todo o texto numa página e/ou aplicação Web a correr num Browser, pode ser seleccionado e copiado. Por outro lado é também possível seleccionar imagens e a estrutura de formulários.

Nas aplicações de Desktop não existe esta possibilidade. Dificilmente um utilizador poderá seleccionar e copiar o conteúdo de uma janela.

Com o surgimento de Plug-ins, tecnologias fechadas que correm nos browsers, a possibilidade de selecção de texto e imagens tende a desaparecer. Numa aplicação desenvolvida em Flash caso os conteúdos sejam carregados dinamicamente, não são passíveis de ser seleccionados. Por outro lado, um utilizador tem de ter presente a plataforma em que está, Web ou Desktop, para compreender se o paradigma de interacção está ou não disponível.

É necessária uma solução mais consistente e transversal aos vários tipos de aplicações e plataformas em que estas são disponibilizadas.

2.3.3.3 Manipulação directa

O termo *manipulação directa* é descrito na Wikipedia [40] como,

“Direct manipulation is a human-computer interaction style which involves continuous representation of objects of interest, and rapid, reversible, incremental actions and feedback. The intention is to allow a user to directly manipulate objects presented to them, using actions that correspond at least loosely to the physical world (...)”,

um paradigma de interacção baseado na representação gráfica de objectos e na possibilidade de o utilizador os poder manipular, de forma semelhante ao que faria no mundo real, conseguindo com isso executar um conjunto de acções num sistema informático.

Este tipo de paradigma está bastante presente em aplicações Desktop. Torna mais natural a interacção entre os utilizadores e os computadores, bem como, torna determinadas tarefas mais fáceis e rápidas de realizar. Não obstante as vantagens, apresentadas por este tipo de interacção, é igualmente constatável que a utilização do teclado, e atalhos de teclas, pode igualmente acelerar um processo de interacção especialmente para utilizadores mais experientes.

Um dos factores de sucesso da manipulação directa é o atraso e feedback dos sistemas às acções do utilizador. O atraso deve ser reduzido ao mínimo e o feedback imediato, de outra forma os utilizadores têm tendência de repetir a acção até que algo aconteça. O atraso e a falta de feedback das acções do utilizador podem causar dificuldades e constrangimentos à utilização deste paradigma de utilização, prejudicando significativamente a experiência de interacção.

Dois dos exemplos bastante familiares de manipulação directa são o conceito de Drag-and-Drop, adoptado pela maior parte dos sistemas operativos, e os editores WYSIWYG (*what you see is what you get*) existente em aplicações de edição de conteúdos multimédia (e.g. Adobe Photoshop, Adobe Premier, Audacity, etc.)

Por outro lado, este paradigma de interacção pode levantar grandes dificuldades para pessoas com deficiência, em particular visual ou motora. De modo a não se tornar um constrangimento para essas pessoas, aplicações que recorram a este paradigma devem contemplar soluções de interacção alternativas – Acessibilidade.

Os interfaces para o utilizador baseados em HTML não suportam este tipo de possibilidades. A introdução de tecnologias RIA (Java, Flash Ajax) veio tornar possível a utilização deste paradigma em aplicações Web. No entanto, devido ao atraso introduzido pela arquitectura cliente servidor, a utilização deste paradigma online tem sido cuidadosamente adoptado. Como referido, o atraso e falta de feedback neste tipo de interacção confundo os utilizadores dando inevitavelmente origem a erros. Por outro lado, devido aos modelos tradicionais de interacção que os utilizadores adquiriram para páginas Web, não estão habituados à possibilidade de poderem recorrer à manipulação directa em aplicações Web.

A convergência, na utilização deste paradigma, típico de aplicações Desktop para aplicações Web é notória. E ainda que a sua implementação para aplicações Web seja mais complexa, é notório o esforço que tem vindo a ser realizado nesse sentido. Esforço esse que poderá ser recompensado no futuro por se tratar de um paradigma de interacção mais natural, bem como, consistente entre os dois mundos Web e Desktop.

2.3.3.4 Drag and Drop

O conceito de *drag and drop*, como referido no ponto anterior é uma das características da manipulação directa. Devido à sua especificidade e relevância, faremos uma análise mais pormenorizada neste mecanismo de interacção.

Genericamente, através do *drag and drop* arrastar objectos para diferentes secções do ecrã onde, dependendo da secção onde largarmos o objecto, será despoletada uma acção.

Seguem-se alguns exemplos disponíveis na Wikipedia [41]:

- Arrastar um ficheiro para uma aplicação despoleta a abertura do mesmo nessa aplicação;
- Ao arrastar um ficheiro para uma nova localização, podemos estar a criar um novo ficheiro ou apenas a mudar de localização;
- Organizar a disposição de elementos numa interface gráfica;
- Arrastar um trecho de texto de um ponto para outro.

Ainda que este seja um conceito tradicional nas AD, só recentemente têm começado a surgir na Web através das novas possibilidades abertas pelo Ajax. Actualmente já é possível utilizar o *drag and drop* de elementos dentro de uma página Web. No entanto, o próximo passo da evolução apresenta um desafio ainda maior.

Imaginemos que se pretende arrastar um ficheiro presente no Desktop directamente para uma aplicação Web específica a correr no Browser (e.g. fazer o upload de uma fotografia para a minha página pessoal). No cenário actual o Browser tentaria abrir esse ficheiro numa nova janela.

O exemplo referido demonstra bem que ainda que a convergência esteja em curso, existem ainda limitações e restrições que não permitem a implementação plena deste paradigma de interacção. Como referido anteriormente, estas restrições devem-se ao facto de as RIA orientadas à Web (funcionam dentro do Browser) não terem acesso ao sistema operativo nem ao Hardware.

Em suma, ainda que o Ajax e outras tecnologias RIA representem um passo fundamental no processo evolutivo do *drag and drop* nas AW, a perspectiva de que um dia a sua utilização esteja uniformizada e amplamente disseminada, novas possibilidades se abrirão para potenciar a experiência de utilização em determinadas aplicações.

2.3.3.5 Anular (*undo*)

Ou o botão de emergência, como é designado por utilizadores menos experientes. Um facto consumado é que as pessoas cometem erros na utilização do decorrer da utilização de uma determinada aplicação, seja ela Desktop ou Web. Urge daí a importância da existência de uma funcionalidade capaz de “apagar o erro” ou simplesmente anular uma acção realizada indesejada ou incorrectamente.

Nas aplicações Desktop, esta funcionalidade está quase sempre no menu editar ou disponível pelo comando de teclas “Ctrl+Z”. Podemos verificar que a sua utilização pode variar consoante a experiência do utilizador mas, mais cedo ou mais tarde todos os utilizadores cometem erros, logo rapidamente os utilizadores mais novos sentem a necessidade desta funcionalidade.

Caso não esteja disponível – gera-se o pânico – pois é algo que os utilizadores estão à espera que exista. A possibilidade de não poder anular uma acção, simplesmente não é aceitável em aplicações de software.

No que toca a aplicações Web, este conceito está longe de ser pacífico. Tradicionalmente, a utilização da Web consiste no paradigma de avançar para uma nova página ou voltar à página anterior. Surge então a confusão com a função “Anular (*undo*)”. A possibilidade de navegar para a próxima página ou página anterior, nada tem a haver com a capacidade de anular uma determinada acção. Devido ao modelo *response-request* tradicional nas aplicações Web, a função anular apenas actua sobre as acções realizadas na página actual. Por outro lado a função de próxima ou página anterior, permite ao utilizador navegar no histórico de páginas que visitou.

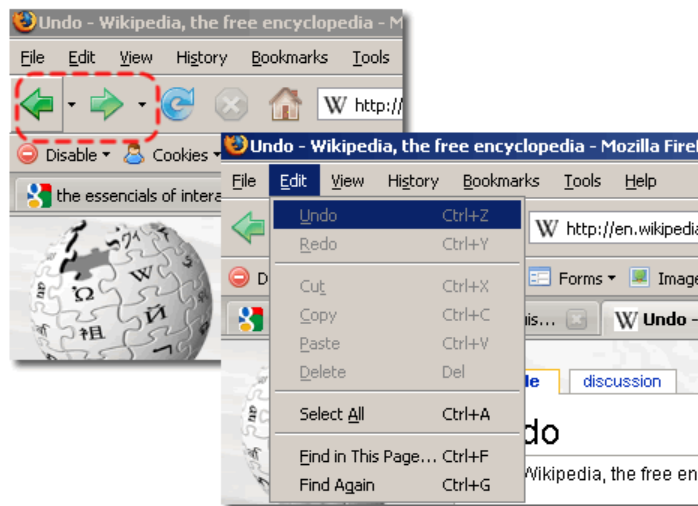


Figura 12: Browser: Página Seguinte e Anterior (esquerda) vs. Anular (direita)

Segundo o estudo realizado por Paul Pop [42], entre as diferentes funcionalidades presentes no estudo, a funcionalidade “Anular” era a tarefa que os utilizadores realizavam com maior rapidez em aplicações Desktop. Por outro lado, o estudo demonstra também que, nas aplicações Web, os utilizadores recorrem erradamente ao botão “Página anterior” na perspectiva de obterem um esperado da função “Anular”.

Este estudo empírico revela que existem efectivamente alguns conflitos e confusão na utilização da funcionalidade “Anular” dependendo do tipo de aplicações Web ou Desktop. Ainda que seja possível identificar uma tendência na convergência dos dois mundos, torna-se pertinente repensar a forma como esta funcionalidade está disponível nas aplicações Web, caso contrário a confusão na sua utilização estará na base de erros de utilização e má experiência de utilização.

Notificações e alertas

As notificações e alertas representam um paradigma de interacção particularmente relevante para aplicações que correm de forma invisível para o utilizador. As notificações podem ter por base eventos de sistema (e.g. nova mensagem de email; alerta para utilizador online) ou tarefas agendadas (e.g. agendas electrónicas, plataformas de trabalho colaborativo).

Em aplicações Desktop esta funcionalidade será implementada através que de janelas no centro do ecrã, ou de uma forma mais discreta e menos invasiva, alertas que surgem e desaparecem num dos cantos do ecrã ou área de interacção ou icons intermitentes.

Esta funcionalidade está igualmente disponível para aplicações Web que permitem implementar alertas na área central do ecrã (e.g. JavaScript) ou notificações com animação de *fade-in* e *fade-out* (animação suavizada de aparecimento e

desaparecimento de um elemento) através de bibliotecas de animação Ajax (e.g. Script.aculo.us⁴⁰).

Um bom exemplo de convergência é o Google Calendar, onde são apresentados notificações, recorrendo a Javascript, de eventos definidos pelo utilizador. No entanto, estas notificações aparecem na respectiva página Web aberta no Browser e não sobre as restantes janelas abertas no sistema operativo. Desta forma, se o utilizador tiver várias janelas do Browser abertas ou temporariamente estiver a utilizar uma aplicação Desktop, não irá visualizar a notificação apresentada.

Um outro exemplo da relevância de alertas e deficiente implementação na Web é o facto de, numa aplicação Desktop, quando inadvertidamente damos indicação para fechar uma aplicação ser apresentado uma notificação a indicar que existem alterações que ainda não foram guardadas. Numa aplicação Web tradicional este mecanismo não é possível logo, no cenário indicado, o utilizador perderia todas as alterações realizadas.

Mais uma vez, as limitações apresentadas provêm da limitação das aplicações Web funcionarem dentro de um Browser. Numa perspectiva de evolução, poderia ser interessante desenvolver uma Framework que permita às aplicações Web lançar notificações e alertas com o impacto e presença das notificações apresentadas por aplicações Desktop.

2.3.3.6 Conclusões da comparação dos paradigmas de interacção

Ainda que sejam visíveis as tendências de convergência entre os paradigmas de interacção presentes em aplicações Web e Desktop, são também evidentes os sinais de que a convergência será conseguida na sua plenitude.

Os pressupostos e restrições base inerentes às diferentes plataformas, Web e Desktop, limitam e promovem diferentes mecanismos de interacção. Esta análise leva-nos mesmo a colocar a hipótese de existirem paradigmas de interacção que nunca serão transpostos de uma plataforma para a outra devido a restrições tecnológicas ou simplesmente porque a sua transposição não introduz vantagens para os utilizadores.

Acreditamos no entanto que as RIA, mais orientadas ao Desktop ou à Web, representam um passo seguro na perspectiva de uma maior consistência na transposição e disseminação de paradigmas existentes em ambas as plataformas. Ao mesmo tempo irão permitir transpor o melhor de cada uma das plataformas, tornando mais uniforme a experiência de utilização, independentemente da plataforma em que o utilizador se encontre.

⁴⁰ Script.aculo.us, biblioteca Javascript com componentes visuais para potenciar as interfaces com o utilizador em aplicações Web, <http://script.aculo.us/>, Acesso: 2009-03-12

2.3.4 Usabilidade

O termo Usabilidade pretende “*definir a facilidade com que as pessoas podem empregar uma ferramenta ou objecto a fim de realizar uma tarefa específica e importante. (...)*” [43].

No que concerne à Interação Humano-Computador e Ciência da Computação, Usabilidade refere-se normalmente à “*simplicidade e facilidade com que uma interface gráfica, um programa de computador ou um website pode ser utilizado. (...)*” [43].

Neste ponto iremos tentar comparar os principais aspectos nos modelos mentais dos utilizadores relativos à usabilidade das aplicações Web e Desktop.

2.3.4.1 Modelos mentais

Consideremos um modelo mental⁴¹, como a explicação/representação de alguém sobre o modo de funcionamento de um determinado sistema, seja ele um sistema informático ou um outro sistema material. Nesta representação poderão também estar presentes as relações entre as várias componentes do sistema, bem como, o impacto que determinadas acções poderão ter noutros sistemas.

De uma forma, quase inconsciente, os modelos mentais ajudam-nos a moldar o nosso comportamento e a definir o modo como resolvemos os problemas com que somos confrontados.

Num sistema informático, mais concretamente na vertente de software, a preocupação com os modelos mentais dos utilizadores é crucial, uma vez que o facto de uma aplicação ser mais ou menos intuitiva para os utilizadores poderá definir o sucesso ou o fracasso dessa aplicação.

No contexto deste trabalho, a análise realizada tenta captar os diferentes modelos mentais associados a aplicações Desktop e Web e perceber, por um lado, se existem sinais de convergência e por outro, identificar os aspectos mais relevantes que possam influenciar a evolução desses modelos.

Desktop

No contexto das aplicações Desktop, o principal modelo mental gira em torno da metáfora do Ambiente de Trabalho (*Desktop*). Neste “Ambiente de Trabalho” os principais elementos são: Janelas, Ícones, Menus e Dispositivo apontador (JIMD) ou, pelo termo mais utilizado WIMP (*Windows, Icons, Menus and Pointing Device*).

⁴¹ Modelo mental, Wikipedia, http://en.wikipedia.org/wiki/Mental_model, Acesso:2009-03-12

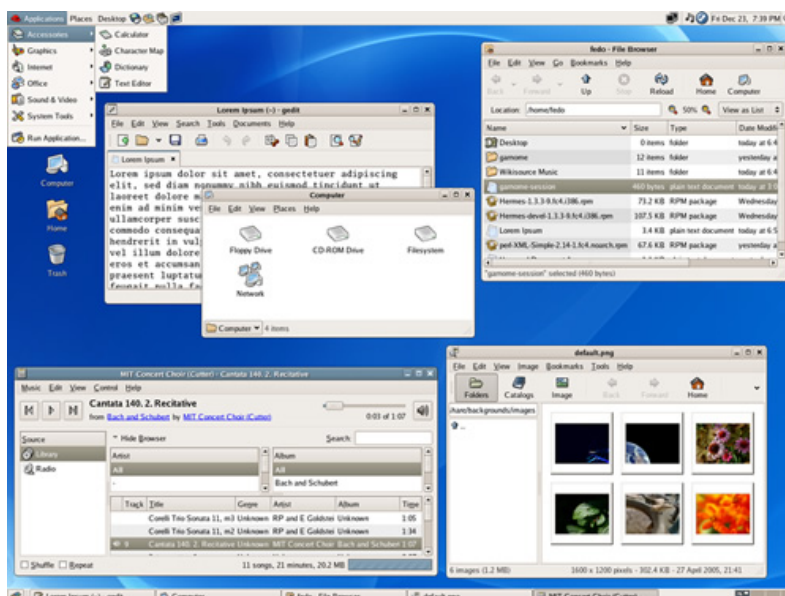


Figura 13: Ambiente de trabalho⁴² (Linux)

O sucesso deste modelo de interacção prende-se com o facto de tentar replicar a essência de determinadas tarefas realizadas no mundo material e a sua transposição para as aplicações de software. O facto de se utilizarem conceitos como: pastas, documentos, reciclagem; ajudaram de facto os utilizadores a interiorizar e realizar, de forma bastante intuitiva, tarefas ao nível das aplicações Desktop. Para além da metáfora do “Ambiente de trabalho”, o paradigma de interacção “Manipulação directa”, apenas possível graças aos *Pointing Devices* (e.g. rato do computador), permitiu levar mais longe a metáfora com o mundo real, possibilitando aos utilizadores manipular directamente e de uma forma natural (e.g. *drag and drop*), elementos na interface gráfica da mesma forma como procedem no mundo material.

Os menus representam acções e comandos. Cada menu permitirá realizar um conjunto de acções sobre o conteúdo de um determinado documento ou dar início a um novo processo ou tarefa com base no contexto em questão.

Através manipulação directa de ícones é possível consultar o conteúdo de um documento ou pasta, iniciar uma aplicação, bem como, despoletar um variável conjunto de acções. Tipicamente, uma acção é despoletada por um clique duplo sobre o ícone (acção replicada pela tecla *return* ou *enter*), e o *backspace* serve para apagar conteúdo.

⁴² Desktop metaphor, Wikipedia, http://en.wikipedia.org/wiki/Desktop_metaphor, Acesso-03-12



Figura 14: Teclas *Enter* e *Return* (teclado convencional)

Web

As aplicações Web surgem como uma sub-classe do conceito WIMP. Uma aplicação Web consiste, basicamente, em mais uma aplicação – o Web Browser, no *Ambiente de trabalho*. Da mesma forma que um processador de texto apresenta o conteúdo de um documento que pode ter texto, tabelas e imagens, o Browser permite a visualização de um documento HTML. A introdução de informação mantém-se através do teclado e rato e até aqui a consistência do modelo metal Web apresenta-se coerente com o Desktop.

Os problemas começaram a surgir com a evolução da Web, dos conteúdos e da sofisticação das aplicações que se pretende disponibilizar através desta plataforma. A ideia de um documento simples de HTML foi amplamente alargada passando agora a contemplar: imagens, vídeos, estilos, scripts, animações, etc..

Os elementos típicos no conceito WIMP não estão presentes numa página Web tradicional. Não é possível a manipulação de janelas dentro de uma página no Browser e quando muito é possível abrir nova janelas do Browser através de *pop-up* (caso esta funcionalidade não esteja bloqueada pelas preferências do utilizador.)

Os menus são utilizados de uma forma contraditória às aplicações Desktop. Uma vez que a aplicação Web não tem acesso ou controlo sobre os menus do Browser, cada aplicação Web apresenta o seu próprio menu, utilizados principalmente com o propósito de permitirem a navegação dentro da aplicação.

Na Web tradicional não é possível a manipulação directa de elementos na página Web, os ícones e imagens são utilizados com marcação de secções de texto, logótipos ou para reforço da existência de um link. Tipicamente, uma acção é despoletada por um clique único sobre um link (texto ou imagens), as teclas Enter e Return permitem submeter formulários ou activar links com *Focus* activo, e o *backspace* serve para apagar conteúdo.

Convergência

Efectivamente os modelos mentais presentes na Web e no desktop divergem em vários aspectos. A uniformização é uma prioridade no sentido de facilitar aos utilizadores alterarem entre aplicações Desktop e aplicações Web sem que isso exija grandes adaptações ou considerações.

Do ponto de vista da convergência necessária, a Web deverá continuar a evoluir no sentido de possibilitar de forma consistente largamente o disseminado modelo mental do Desktop. A ideia de que as aplicações Desktop deverão ser reformuladas de acordo com o modelo mental presente na Web é uma ideia limitadora. Mais interessante e provável do que reformular as aplicações de Desktop é a perspectiva evoluir contínua, e quem sabe reformulação, das tecnologias base da Web.

Algumas RIA emergentes apresentam já algumas pistas para essa evolução. Aplicações desenvolvidas tendo por base a arquitectura Ajax, como o Meebo⁴³ (ver Figura 15), apresentam por exemplo uma solução para a inconsistência ao nível da manipulação de janelas dentro de uma aplicação Web.

Um outro exemplo de manipulação directa de elementos numa página Web é o iGoogle⁴⁴ (ver Figura 16), onde o utilizador pode manipular (adicionar, remover, organizar) os vários painéis que pretende na sua página personalizada.

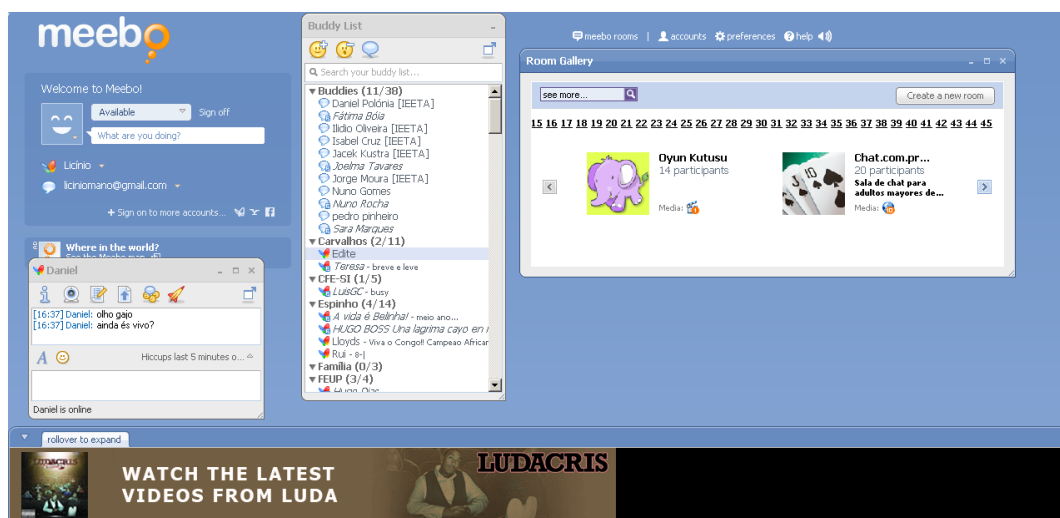


Figura 15: Meebo – Janelas dentro de uma aplicação Web

⁴³ Meebo, aplicação Web de comunicação em tempo real, desenvolvida com Ajax, <http://www.meebo.com/>, Acesso: 2009-03-12

⁴⁴ iGoogle, versão personalizada por cada utilizador da página inicial do motor de busca Google, <http://www.google.com/ig>, Acesso: 2009-03-12

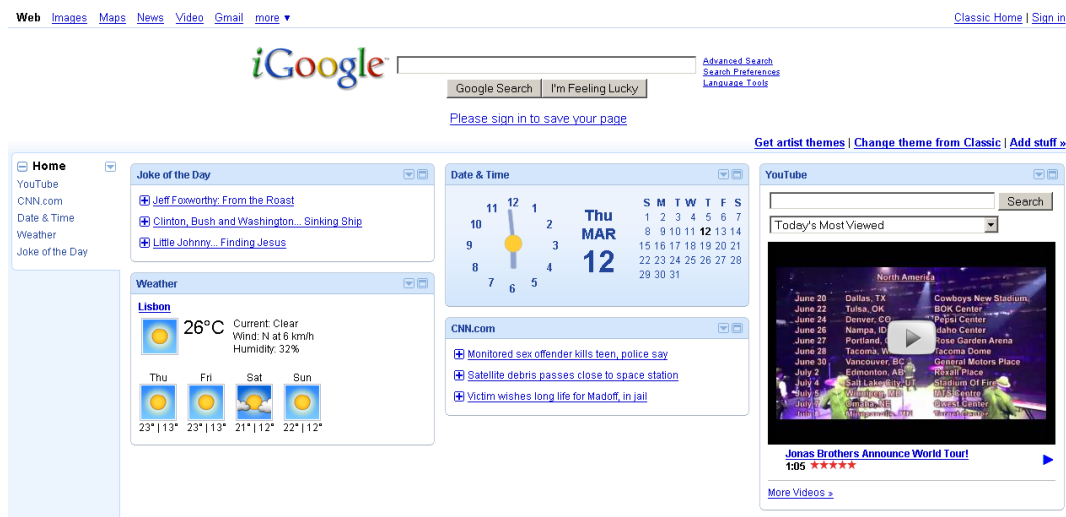


Figura 16: iGoogle – Manipulação directa de elementos em aplicação web

2.3.4.2 O Ficheiro

Desktop

Nas aplicações Desktop recorre-se ao conceito de ficheiro para agrupar um conjunto de informação. Sempre que o utilizador procede a alterações no conteúdo poderá gravar essas alterações no mesmo ficheiro ou criar um novo ficheiro (Guardar ou Guardar como). Existe ainda um mecanismo, invisível para o utilizador, que procede à gravação automática das actualizações de modo a permitir a recuperação das mesmas em caso de falha da aplicação (e.g. MS Word – *Recuperação automática*).

Web

Na Web não existe o conceito de ficheiro para os utilizadores. Para vários efeitos a informação está num servidor e pode ser consultada através de uma página Web. Em aplicações Web que possibilitam a actualização de conteúdo existe inevitavelmente o botão para “Guardar” ou “Submeter” alterações. Tradicionalmente, na plataforma Web não existe a possibilidade de “Recuperação automática”.

Convergência

Actualmente existem alguns exemplos de convergência de aplicações Web, no sentido de um transportarem modelo de funcionamento das aplicações Desktop. Consideremos como exemplo a aplicação Web Google Docs (ver Figura 17).

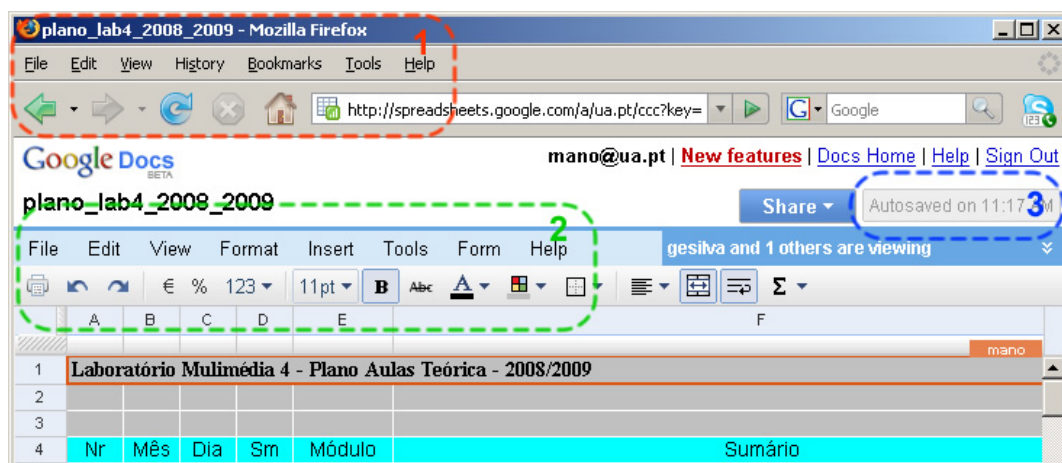


Figura 17: Google Docs – Interface Gráfica para o Utilizador

Do ponto de vista tecnológico esta aplicação Web representa uma evolução revolucionária, uma vez que transporta para aplicação as funcionalidades e modelos mentais tão disseminados das aplicações Desktop.

Não obstante o salto tecnológico que representam, existem algumas inconsistências relevantes. Na Figura 17 encontram-se assinaladas três secções específicas do ecrã:

1. Menu do Browser (aplicação Desktop), com diversos botões de controlo da aplicação;
2. Menu da aplicação Web propriamente dita, onde existem menus com nomes e funcionalidades duplicadas;
3. Informação a hora a que foi realizada a última gravação automática;

Começando a análise pela secção 3, para a qual rapidamente percebemos o seu propósito. Finalmente está disponível, numa aplicação Web, uma funcionalidade que trata da gravação automática das alterações realizadas pelos utilizadores.

As grandes inconsistências surgem nas secções 1 e 2 e na duplicação de menus que apresentam. Se repararmos com a devida atenção os menus: *File*, *Edit*, *View*, *Tools* e *Help*; encontram-se repetidos. Inevitavelmente a confusão instala-se nos utilizadores.

Mas afinal qual dos menus devo utilizar?

Os utilizadores mais experientes, com maior ou menor dificuldade serão capazes de distinguir que os menus do Browser possibilitam o controlo sobre a aplicação Desktop propriamente dita. O menu dentro da página Web refere-se estritamente aos conteúdos presentes na página.

Se analisarmos com mais pormenor para os diferentes menus *File* (ver Figura 18), facilmente percebemos que a confusão pode ainda aumentar. O número de itens repetidos é evidente, prejudicando muito a funcionalidade por trás de cada item. Consideremos a função Imprimir “*Print*” presente em ambos os menus.

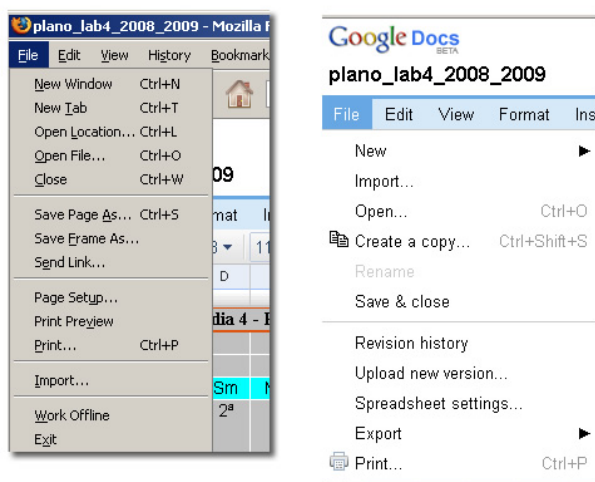


Figura 18: Google Docs: File Menu

O que será impresso se utilizar o imprimir do browser? E caso faça imprimir através da aplicação?

Este é o tipo de situação em que vemos reduzida a qualidade da experiência de utilização, quando a intenção base foi apresentar ao utilizador um modelo mental que este já conhecia do Desktop.

Com base no exemplo apresentado, podemos verificar que para a transposição consistente de determinados paradigmas do Desktop para a Web, não basta replicar, sendo também necessário adaptar e enquadrar com o contexto em que a aplicação Web é executada – no Browser.

2.3.4.3 O Browser

Ao longo do texto a palavra Browser ou Web Browser tem sido com letra maiúscula. Conforme indicado para as palavras Web e Desktop, também a palavra Browser representa mais do que uma aplicação Desktop. Representa todo um contexto necessário para o utilizador poder visualizar uma página Web.

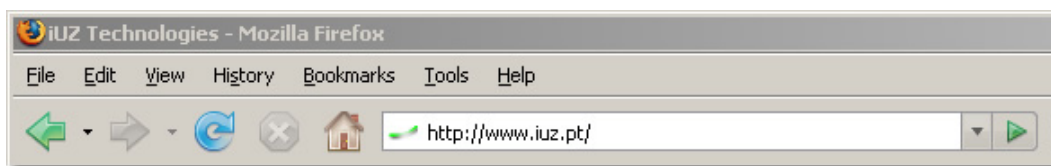


Figura 19: Principais funcionalidades do Browser (Firefox)

Desde 1991 que os Browsers são a forma preferencial e mais rápida para acesso à Web. O Browser permite o acesso a conteúdos na Web, bastando para isso a indicação do respectivo URI. Segue-se uma descrição das principais funcionalidades de navegação (ícones da esquerda para a direita) presentes no Browser Mozilla Firefox.

- Anterior: volta a página consultada antes da actual (navegação pelo histórico de páginas visitadas pelo utilizador);
- Seguinte: caso tenhamos voltado para uma página anterior, permite avançar para a página visualizada a seguir (navegação pelo histórico de páginas visitadas pelo utilizador);
- Actualizar: volta a solicitar / carregar a página actual;
- Parar: para o carregamento da página actual. Esta funcionalidade está apenas disponível enquanto decorre o processo de carregamento da página actual.
- Home page do utilizador: permite regressar à home page definida pelo utilizador;
- URI: identificador unívoco de recursos na Web;

O sucesso do Browser deve-se principalmente à sua simplicidade de funcionamento e disponibilidade para vários sistemas operativos (e.g. Windows, Macintosh e Linux), bem como, para diferentes dispositivos (e.g. computadores, telemóveis, PDA's, consolas de jogos, televisão interactiva). Inevitavelmente com o passar do tempo foram surgindo diferentes Browsers e, devido ao facto de cada um apresentar a sua própria implementação do HTML, aumentando a dificuldade da tarefa dos programadores de aplicações Web no sentido de garantirem que uma aplicação funcionaria da forma pretendida nos vários Browsers existentes.

Não obstante as mais-valias apresentadas pelos Browsers, Jacob Nielsen escreveu sobre a “Death of Web Browsers” – A morte dos Web Browsers, onde afirma que o aspecto chave para o futuro dos computadores é “eliminar os Web Browsers” [44]. Nielsen assenta a sua opinião no facto de que o acesso à informação não deveria depender da sua localização. Isto é, se a informação está na máquina local, estão deverá utilizar a pesquisa do sistema operativo, se está na internet, o utilizador tem de recorrer a um Browser.

Nielsen desmistifica o papel dos Browsers para o sucesso da Web. As 2 principais e fundamentais funções de um Browser são: apresentar blocos de informação e permitir a navegação entre eles. Sugere também que estas funções devem ser separadas e melhor suportadas pelos sistemas operativos de modo a proporcionar aos seus utilizadores uma experiência de utilização mais consistente com as restantes aplicações.

Nielsen remata a sua opinião, afirmando que os Browsers estão ultrapassados, uma vez que estão a ser utilizados como ferramentas para tarefas para as quais não foram desenhados. Actualmente os Browsers continuam a ser a tecnologia preferencial para acesso a informação disponível na Web, mas fazem-no com grave prejuízo da usabilidade que proporcionam aos seus utilizadores.

Convergência

As alternativas à utilização dos Browsers para aceder a informação disponível na Web ainda não se encontram maduras ou amplamente disseminadas. Contudo, começam a emergir algumas aplicações que pretendem proporcionar aos utilizadores uma experiência de utilização mais consistente com as restantes aplicações de desktop.

De seguida serão apresentados dois projectos que se apresentam como alternativas à utilização dos Browsers. Na sua essência, funcionam como Web Browsers, contudo a sua interface com o utilizador (IU) é mais consistente com as restantes aplicações Desktop.

Começemos pelo projecto GMDesk⁴⁵ (ver Figura 20). Trata-se de uma aplicação Desktop que possibilita aos seus utilizadores, o acesso a aplicações Web (e.g. Gmail, Google Calendar, Google Docs e Google Maps) da mesma forma como faria com outras aplicações Desktop.

Como outras aplicações Desktop, o GMDesk pode ser iniciado a partir de um ícone no ambiente de trabalho, permitindo a utilização de alt-tab (no Windows ou Linux) ou Cmd-tab (no Mac) para trocar de aplicação. O GMDesk possibilita também a utilização de atalhos do teclado para aceder aos diferentes serviços disponíveis. O GMDesk trata-se de um projecto pessoal de Robert Nyman.

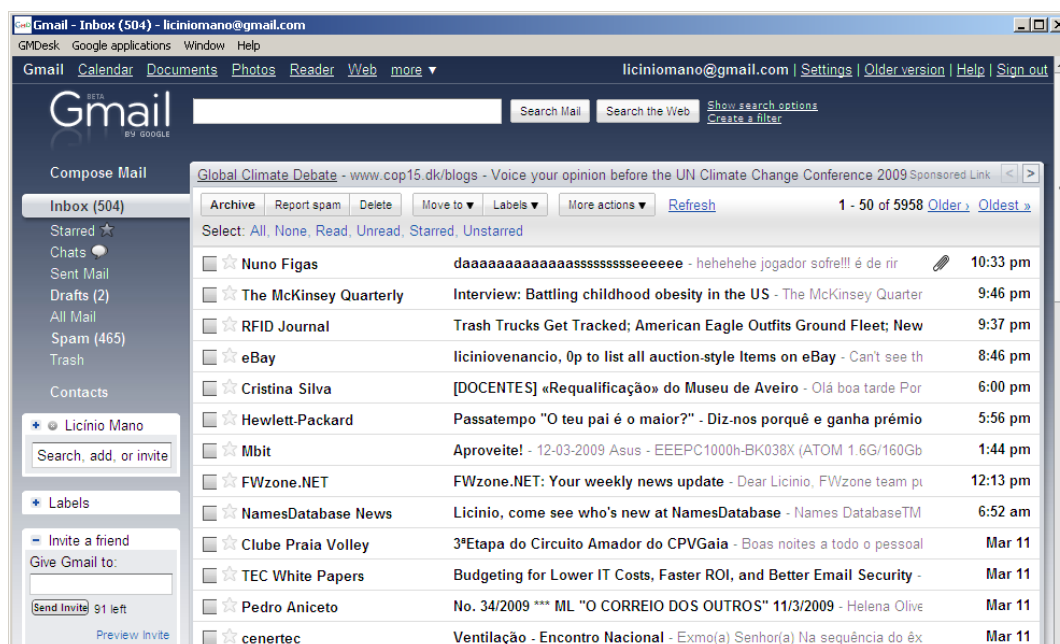


Figura 20: GMDesk (Robert Nyman)

O segundo projecto trata-se do Prism⁴⁶ (ver Figura 21) proveniente do Mozilla Labs. A Mozilla enquadra o Prism como “(..) *experiments to bridge the divide in the user experience*”

⁴⁵ GMDesk, Robert Nyman, <http://www.robertnyman.com/gmdesk/>, Acesso: 2009-03-12

⁴⁶ Prism, Mozilla Labs, <http://labs.mozilla.com/2007/10/prism/>, Acesso: 2009-03-13

between web applications and desktop apps and to explore new usability models as the line between traditional desktop and new web applications continues to blur. (...)”.

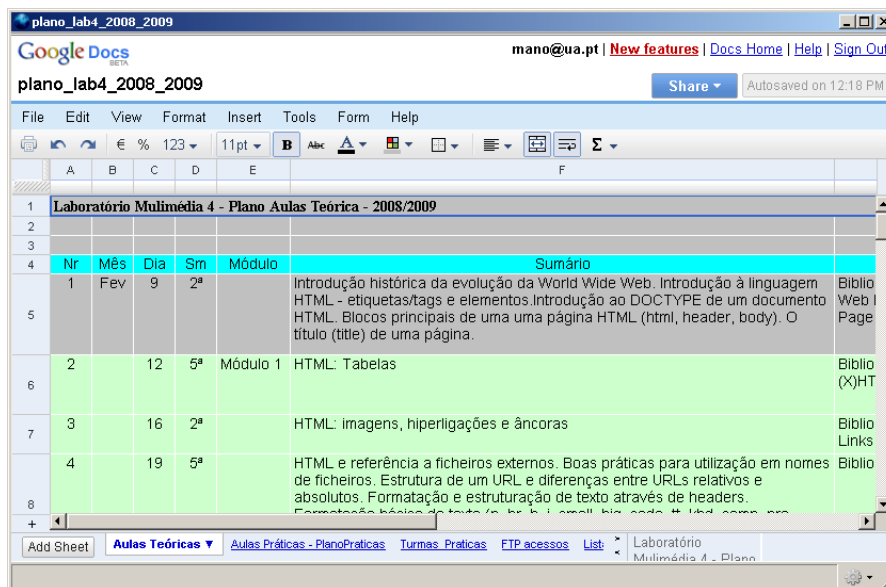


Figura 21: Prism (Mozilla Labs)

Após instalado, o Prism permite criar acessos directos a uma aplicação Web. Para tal, temos de configurar o acesso à aplicação pretendida, com base no formulário apresentado na Figura 22. Neste formulário podemos indicar o tipo de acesso que pretendemos configurar (e.g. ícone no ambiente de trabalho, entrada no menu Iniciar, ícone na barra de arranque rápido). Existe também um conjunto de opções a funcionalidades presentes na interface com o utilizador.

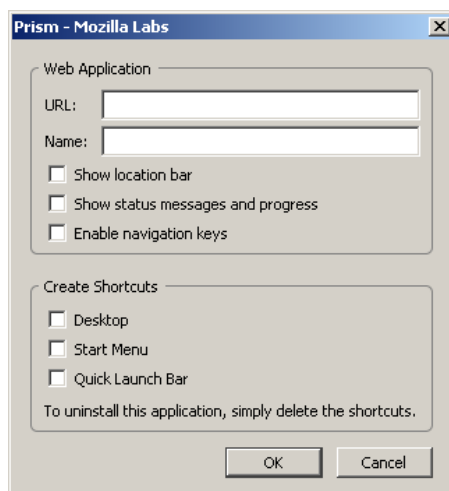


Figura 22: Prism – Criar acesso para aplicação web

Após utilizar por algum tempo as aplicações apresentadas, algumas vantagens tornaram-se evidentes. O facto de integrarem com as restantes aplicações no Desktop, agilizou imenso o processo de acesso às aplicações e experiência de utilização das mesmas. De referir a melhoria da experiência de utilização da aplicação Google Docs, a partir do momento em que passou a existir apenas um menu (ver Figura 21) e todas as funcionalidades aí presentes passaram a fazer todo o sentido.

Ainda que representem apenas experiências e tentem seguir as tendências de evolução apontadas como mais prováveis, na minha opinião, estas soluções representam uma etapa intermédia entre a geração actual e a próxima geração Web.

2.3.5 Aspecto visual (*look and feel*)

Ao compararmos aplicações Desktop (AD) com aplicações Web (AW) torna-se evidente a maior diversidade do aspecto visual das aplicações Web. Este fenómeno surge principalmente por dois motivos:

- O aspecto visual das AD depende directamente do look and feel do sistema operativo onde estão a ser executadas.
- As tecnologias base sobre as quais é desenvolvido a Interface Gráfica para o Utilizador (GUI) das AW, são mais flexíveis, promovendo a criatividade e diferenciação entre cada aplicação existente.

Em suma, a flexibilidade da Web permite o desenvolvimento de GUI mais **criativas e distintas** de *website* para *website*. Por outro lado, as aplicações Desktop apresentam GUI mais **standard e uniformes**, de aplicação para aplicação, facilitam a transição do utilizador entre aplicações.

Segundo Jenifer Tidwell [45], as tendências de convergência indicam que o *look and feel* das aplicações, sejam elas Desktop ou Web, representam um factor crítico de sucesso das mesmas junto do utilizador final. Por esse motivo podemos constatar que cada vez mais as aplicações Desktop apresentam interfaces com o utilizador inovadoras, criativas e em grande parte distintas da concorrência.

Por outro lado, é possível verificar uma tendência nas aplicações Web em direcção à standardização ou, pelo menos, na uniformização e consistência de determinados modelos mentais e paradigmas de interacção entre diferentes aplicações. Um importante contributo para esta tendência resulta do aparecimento de Bibliotecas e Frameworks (e.g. Dojo⁴⁷ e ZK – Direct RIA⁴⁸) que apresentam soluções completas, para desenvolvimento de GUI, e integração com diferentes linguagens de programação Web.

⁴⁷ Dojo, <http://www.dojotoolkit.org/>, Acesso:2009-03-13

⁴⁸ ZK – Direct RIA, <http://www.zkoss.org/>, 2009-03-13

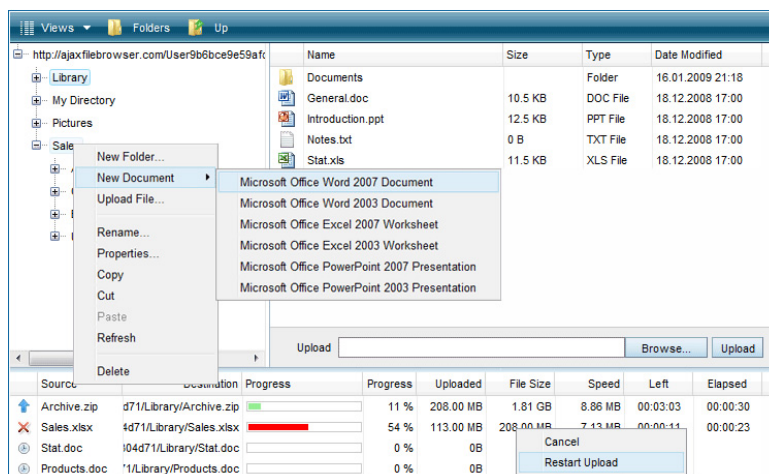


Figura 23: Dojo, exemplo de GUI (Ajax File Browser)

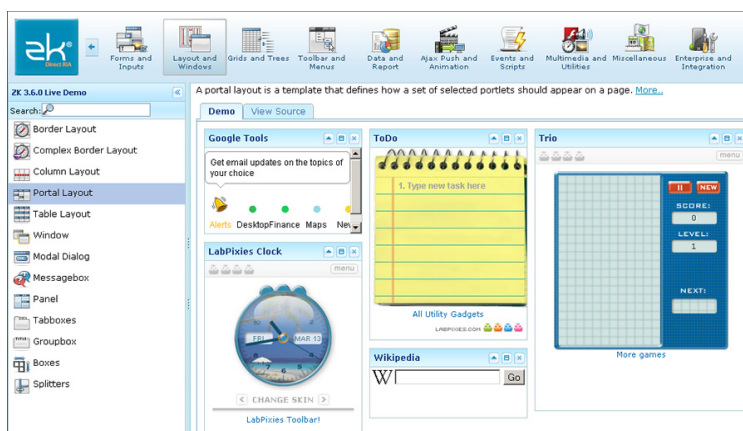


Figura 24: ZK – Direct Ria, exemplo de GUI (Layout de portal)

As bibliotecas indicadas disponibilizam um conjunto de componentes que podem ser utilizados para desenvolver de raiz a GUI de uma AW ou então para resolver situações mais pontuais como: notificações, caixas de diálogo, calendários, estruturas de dados. De modo a facilitar a tarefa dos programadores, estas Bibliotecas costumam apresentar um conjunto de exemplos (*Feature Explorer*) bem documentado dos componentes que disponibilizam e demonstração dos mesmos em funcionamento. Ficam os endereços de alguns desses demonstradores:

- Dojo, <http://dojocampus.org/explorer/>
- ZK, <http://www.zkoss.org/zkdemo/userguide/#f1>
- JQuery⁴⁹, <http://jqueryui.com/demos/draggable/>

⁴⁹ JQuery, <http://jquery.com/>, Acesso: 2009-03-13

2.4 Síntese e conclusões

Este capítulo apresenta uma perspectiva histórica sobre as Interfaces com o Utilizador (IU) de dois tipos de aplicações de software: Aplicações Desktop (AD) e Aplicações Web (AW). Esta perspectiva desvenda e contrapõe vantagens e desvantagens das IU dos tipos de aplicações em discussão, bem como, tenta captar a forma como ambas evoluíram ao longo do tempo. Evolução esta que, juntamente com a convergência que se verifica actualmente entre ambos os mundos, originou um novo paradigma: as Rich Internet Applications (RIA).

Permitiu também clarificar e fundamentar objectivamente que a convergência está em curso, não se tratando apenas de uma teoria. E ainda que tudo indique para a contínua evolução e convergência entre ambas as plataformas, o cenário mais provável exclui a fusão completa entre ambas as plataformas.

Devido às diferenças base na essência tecnológica da cada plataforma – Web e Desktop - o cenário mais provável passa pela convergência dos modelos e contexto de interacção até a um ponto em que se torna transparente para o utilizador final o facto de estar a utilizar uma aplicação Web ou Desktop.

3 Aplicações Ajax

Em 2005, Jesse-James Garret publicou o artigo “Ajax: A New Approach to Web Applications” [25]. Garret reforçou a ideia da crescente convergência entre aplicações Web e Desktop, salientando a emergência de um conjunto de sofisticadas aplicações Web (e.g. Google Maps, Google Suggest) baseadas numa inovadora arquitectura, à qual denominou de Ajax. Efectivamente, as tecnologias na base do Ajax já existiam há algum tempo, mas mesmo assim o artigo assinalou um ponto importante na história, pois representa a visão sólida e madura de arquitectura que todos sentiam mas não sabiam identificar – *não vinha nos livros*.

O Ajax é o acrónimo de “Asynchronous JavaScript + XML”, e ainda que, não estejam directamente presentes na designação, Garret referiu que tecnologias como o XHTML, CSS e DOM são igualmente importantes para todo o conceito Ajax.

Garret refere também que não é inocente o facto de se escrever “Ajax” e não “AJAX”. O termo Ajax pretende descrever muito mais do que um conjunto específico de tecnologias. Pretende descrever algo mais abrangente, uma forma inovadora de pensar e desenvolver aplicações Web tendo por base um conjunto de tecnologias standard e amplamente disseminadas.

3.1 Aspectos essenciais - Ajax

A arquitectura Ajax permite o desenvolvimento de Aplicações Sofisticadas para a Internet (RIA – Rich Internet Applications), promovendo interactividade e *Responsiveness* através da utilização de tecnologias Web standard, amplamente disseminadas e facilmente acessíveis pelos utilizadores finais.

Na sua essência a arquitectura AJAX representa uma nova fase na experiência de utilização proporcionada pelas aplicações Web aos seus utilizadores.

Aplicações Web ou Web Sites?

Ao longo deste trabalho temos utilizado preferencialmente o termo aplicações Web (AW) em vez da designação tradicional *website*. O motivo porque o fazemos deriva de uma realidade emergente, materializada pelo aparecimento de projectos inovadores, que apresenta o Browser como plataforma e a Web como sistema operativo. Hoje em dia as pessoas deixaram de utilizar a Web apenas para consulta de informação. A Web passou a ferramenta para actividades no dia-a-dia, assim como, um canal de entretenimento e socialização.

A complexidade das actividades realizadas com recurso à Web evoluiu e com ela evoluíram também a segurança, usabilidade e todos os aspectos derivados da criticidade e especificidade de cada aplicação. Por este motivo, consideramos cada vez mais apropriado o termo Aplicações Web e detrimento da tradicional designação *Websites* (sítios Web).

Usabilidade

Tradicionalmente após activar um link ou submeter um formulário, o utilizador é forçado a aguardar alguns segundos até conseguir visualizar o resultado da acção através do carregamento de uma nova página. Devido à arquitectura cliente-servidor, a mais pequena alteração na página necessita implica uma nova chamada ao servidor e a actualização de todo o conteúdo presente na página.

Com base em Ajax é possível quebrar esta rotina angustiante para os utilizadores:

1. É possível manipular elementos visuais presentes na página através de JavaScript, evitando a necessidade de um novo pedido ao servidor apenas para esconder ou dar feedback ao utilizador.
2. Os pedidos ao servidor podem ser realizados de forma invisível para o utilizador. Esta forma evita também a actualização completa da página, permitindo actualizar secções específicas da página.
3. É possível captar as acções do utilizador (e.g. teclas carregadas ou cliques do rato), permitindo a implementação de notificações, alertas e outras formas de interacção que melhoram a experiência de utilização de uma aplicação.

As possibilidades introduzidas pelo Ajax permitem aos programadores o desenho e desenvolvimento de aplicações com melhor performance e, acima de tudo, mais consistentes com os modelos mentais e paradigmas de interacção típicos das aplicações Desktop.

Interacção

Ainda que as tecnologias de base sejam diferentes das utilizadas em Desktop, as aplicações Web apresentam cada vez mais, paradigmas de interacção semelhantes aos utilizados para Desktop. Consideremos a manipulação directa de elementos GUI, mais concretamente a possibilidade de *drag-and-drop*. Este mecanismo está presente desde o início das GUI em torno do conceito WIMP. Finalmente, após 20 anos de experiências, este paradigma começa a ser mais usual nas aplicações Web potenciado pelo Ajax e criatividade dos programadores.

Começamos também a assistir à utilização recorrente de componentes complexos, como os editores de texto "What-You-See-Is-What-You-Get", para facilitar a introdução e edição de texto. Surgem também algumas tentativas de utilização de comandos de teclas, bem como, a utilização dos diferentes botões do rato.

Convergência ou evolução, depende da visão de cada um. Inegável é o facto é que a interacção nas aplicações Web está a mudar. Essa mudança passa pela adopção de transposição de alguns paradigmas provenientes do Desktop, mas perspectivamos também o surgimento de novos padrões e novas formas de tirar partido do Browser como plataforma e da Web como sistema operativo.

Standards

Uma aplicação Ajax necessita apenas de tradicional Web Browser (e.g. Firefox, Opera, Chrome, etc.). Ainda que o correcto funcionamento de uma aplicação em todos os Browsers possa necessitar de alguns afinamentos, o Ajax tira partido das funcionalidades base e standard dos Browsers, não necessitando por isso da instalação de *plugins*. Tão importante como o uso de tecnologias standard, trata-se do respeito das convenções definidas para as Interfaces com o Utilizador (IU) em aplicações Web.

Uma aplicação Ajax está disponível para qualquer utilizador do mundo, bastado para o efeito ter acesso a Web e um Browser.

3.2 Tecnologias Ajax

Neste tópico apresentamos uma perspectiva sintética das várias tecnologias utilizadas na arquitectura Ajax. Como é possível verificar, uma grande parte das tecnologias referidas são bastante familiares e populares no contexto do desenvolvimento de aplicações Web.

HTML / XHTML

O HTML, ou a sua variante mais restrita o XHTML, definem a estrutura inicial da página web. O motor Ajax vai alterando e actualizando continuamente esta estrutura, através da manipulação de elementos e definição de eventos. A utilização de XHTML é recomendada, no sentido de tornar o HTML mais consistente, facilitando a identificação e correcção de erros na estrutura HTML da página.

Tecnologia standard convencionada pelo W3C (<http://www.w3.org/MarkUp/>).

CSS - Aspectos essenciais

As CSS (Cascade Style Sheets) permitem refinar o aspecto visual dos elementos presentes numa página Web. Permite também separar estrutura da página do aspecto visual com que os elementos serão apresentados.

O facto de os Browsers apresentarem actualmente um suporte consistente desta tecnologia, tem permitido a disseminação desta tecnologia em detrimento da utilização de tabelas para a definição do layout das páginas (*tableless design*).

Na perspectiva das aplicações Ajax, a grande vantagem das CSS é permitirem uma manipulação consistente através de JavaScript. Com apenas algumas linhas de código é possível fazer um elemento aparecer, desaparecer ou mudar de posição numa página web.

Tecnologia standard convencionada pelo W3C (<http://www.w3.org/Style/CSS/>).

HTTP

Como os tradicionais websites, as aplicações Ajax comunicam com o servidor através de HTTP (Hypertext Transfer Protocol⁵⁰). A diferença reside no facto de em vez de responder com a estrutura completa de uma página, o servidor responde apenas com o resultado específico. Esse resultado será então processado pelo motor JavaScript da página e posteriormente adicionado aos elementos da página.

Da mesma forma a submissão e processamento de formulários deixa de implicar obrigatoriamente um pedido ao servidor de toda a página. A informação do formulário é submetida, e de uma forma transparente para o utilizador, o resultado do processamento no servidor é enviado para o Browser, onde é tratado pelo motor JavaScript e posteriormente apresentado na página.

Processamento do lado do servidor

O processamento do lado do servidor não é dispensável, muito pelo contrário. A persistência de dados, validação de dados submetidos por formulário ou processamentos mais complexos de informação, devem ser mantidos do lado do servidor de modo a garantir aspectos como performance e segurança da aplicação.

A arquitectura Ajax contempla no entanto, a possibilidade de um motor JavaScript presente na página tratar de todas as tarefas inerentes à camada de apresentação de informação, lógica da aplicação e comunicação com a fonte de dados. Desta forma, não seria necessário o processamento do lado do servidor. É uma possibilidade ainda que, conforme indicado, apresenta vantagens e desvantagens ao nível da performance e segurança da aplicação.

JavaScript

Com o aparecimento das aplicações Ajax, uma parte do processamento e tratamento das acções do utilizador passar para o lado da aplicação cliente – o Browser. O JavaScript⁵¹ (inicialmente designada por ECMAScript⁵²) é a linguagem de programação que coordena as actividades no Browser.

⁵⁰ HTTP, Wikipedia, <http://en.wikipedia.org/wiki/HTTP>, Acesso: 2009-03-13

⁵¹ JavaScript, Wikipedia, <http://en.wikipedia.org/wiki/Javascript>, Acesso: 2009-03-13

⁵² ECMAScript, Wikipedia, <http://en.wikipedia.org/wiki/ECMAScript>, Acesso: 2009-03-14

Até ao aparecimento do Ajax, os programadores tinham pouco conhecimento sobre JavaScript devido à reduzida importância que este detinha no desenvolvimento de uma aplicação Web. Com a emergência e sucesso do Ajax o JavaScript passou a ser considerada uma linguagem tão importante e influente como as restantes linguagens de programação do lado do servidor.

XML e DOM

A DOM (*Documento Object Model*⁵³) é a estrutura hierárquica que representa a estrutura XML de um documento XHTML. Através de JavaScript é possível aceder e manipular a estrutura DOM de uma página Web.

Poderá surgir uma outra estrutura DOM resultante de um pedido ao servidor. Isto é, a resposta do servidor a um pedido solicitado através de Ajax poderá vir no formato XML. Esta resposta terá então a sua estrutura DOM que poderá ser percorrida e manipulada através de JavaScript.

Em alternativa ao XML a resposta do servidor poderá vir num formato simples de texto ou no formato JSON (JavaScript Object Notation⁵⁴).

Event Model

De modo a permitir que o JavaScript suportasse a possibilidade de responder a eventos como cliques do rato ou teclas pressionadas, surgiu o conceito de *Event Model*. Este modelo é a chave para a implementação de interactividade dentro do Browser.

Web Remoting

Web Remoting representa a capacidade do JavaScript comunicar directamente com um servidor remoto. Graças ao *Web Remoting* é possível implementar chamadas ao servidor invisíveis para o utilizador. A base de funcionamento do *Web Remoting* reside no objecto XMLHttpRequest, que abordaremos mais à frente neste trabalho.

3.3 Ajax - Desafios e oportunidades

Neste trabalho abordamos o tema do Ajax na tentativa de descrever uma arquitectura madura e consistente. No entanto, trata-se de uma tecnologia relativamente recente apresentando um conjunto de desafios e oportunidades para o seu amadurecimento e disseminação. Podemos até afirmar, o Ajax esteve na base do movimento “Web 2.0” e que todos os dias são lançadas novas ideias e projectos com base nesta tecnologia.

⁵³ DOM, Wikipedia, http://en.wikipedia.org/wiki/Document_Object_Model, Acesso: 2009-03-13

⁵⁴ JSON, Wikipedia, <http://en.wikipedia.org/wiki/Json>, Acesso: 2009-03-13

As grandes questões em torno do Ajax passam por perceber se é apenas mais uma moda tecnológica e como prevê a coexistência com outras inovações no contexto Web.

Neste ponto iremos identificar e apontar algumas dicas para desafios e oportunidades que se colocação sobre o processo de maturação e disseminação do Ajax.

3.3.1 Standards e Compatibilidade

A compatibilidade entre as API (*Application Programming Interfaces*) veio facilitar desenvolvimento de aplicações Web. Ainda que permaneçam algumas diferenças entre os diferentes Browsers, o factor chave é que estes procedam à implementação dos standards e convenções lançados por organismos uniformizadores como o W3C.

No entanto este cenário pode mudar a curto/médio prazo. Os mesmos fenómenos e necessidades na origem de aplicações com base em Ajax, estão também a potenciar o surgimento de outras tecnologias semelhantes. Se por um lado da diversidade nascem as alternativas, por outro poderá causar o surgimento de tecnologias divergentes, e pior ainda, não compatíveis.

Um grupo com especial interesse no estudo e lançamento dos alicerces para novas evoluções na Web é o WHATWG (*Web Hypertext Application Technology Working Group*⁵⁵). O factor preponderante no trabalho deste grupo é o conceito “Aplicação” e a visão da Web como plataforma para aplicações. Este grupo foi fundado em 2004, após um evento do W3C e os principais promotores são: Apple, Mozilla Foundation, Opera Software e o próprio W3C.

A ausência da Microsoft deste grupo levanta algumas incertezas sobre o futuro da Web. A ausência de um *player* importante como a Microsoft na discussão sobre as evoluções na Web e de que forma os Browsers a devem acompanhar, abre logo espaço para a especulação sobre quais funcionalidade e quais as inovações que serão introduzidas pelo próximo Internet Explorer (IE). Será que terá funcionalidades específicas para a Framework Atlas⁵⁶ (Framework Ajax proprietária da Microsoft), por exemplo a capacidade de armazenamento local de dados?

Este tipo de incertezas e eventuais divergências na implementação das APIs dos Browsers poderá abrir um novo ciclo da tão famosa “Browser Wars”. Se assim for, surgem duas alternativas possíveis para o desenvolvimento de aplicações Web:

- Desenvolvimento específico para um determinado Browser (opção polémica, mas por vezes revela-se a opção mais pragmática);
- Evitar o recurso a funcionalidades específicas de um determinado Browser, apostando nos standards, convenções, e recorrer a *plugins* caso uma funcionalidade pretendida não seja suportada por um determinado Browser.

⁵⁵ WHATWG, site oficial, http://wiki.whatwg.org/wiki/FAQ#The_WHATWG, Acesso: 2009-03-13

⁵⁶ Atlas, Framework Ajax da Microsoft, <http://www.asp.net/ajax/>, Acesso: 2009-03-13

3.3.2 Actualização dos Browsers

Independentemente do futuro reservando para a Web os Browser continuarão a evoluir. Actualmente, sempre que é lançada uma nova versão os utilizadores têm de descarregar e instalar a nova versão.

Estarão os utilizadores dispostos a este ciclo repetitivo?

Justificar a mudança ou a instalação por evoluções tecnológicas como Ajax não basta. Aliás, se perguntarmos a um utilizador comum da Web se faz ideia do que é o Ajax, em Portugal, a resposta mais provável é: “Um produto de limpeza. (ver Figura 25)”.



Figura 25: Detergente AJAX (produto da Colgate Palmolive)

Como tal, serão necessários argumentos de peso para fazer com que os utilizadores tradicionais actualizem o Browser presente na sua máquina. Mais improvável ainda será a mudança para outro Browser.

Por outro lado, os Browsers têm desenvolvido estratégias inovadoras de auto actualização. Desta forma o utilizador é apenas notificado para reiniciar o Browser para as actualizações fazerem efeito. Portanto, se a versão actual do Browser instalado já contemplar esta funcionalidade, em princípio, o Browser estará sempre actualizado com as novas tecnologias mais actuais.

Aceitação dos utilizadores

Numa primeira instância podemos afirmar que os utilizadores apreciam as novas aplicações Web e paradigmas de interacção possíveis nas aplicações desenvolvidas com Ajax. A questão coloca-se agora na reacção colocação sobre a disseminação do Ajax por toda a Web.

Como irão os utilizadores reagir a funcionalidades como drag-and-drop ou outros paradigmas de interacção, típicos de aplicações Desktop, disponíveis em aplicações Web?

Numa aplicação Ajax os URI e o botão para a página anterior não apresentam o mesmo comportamento das aplicações Web tradicionais. Devido ao facto de através do Ajax ser possível evitar a actualização de toda a página, o Browser não consegue perceber as mudanças de informação na página, logo o botão “Anterior” não permite voltar ao estado anterior em que a página se encontrava. Por outro lado, se a aplicação mantém sempre o mesmo URI, então não é possível assinalar (*Bookmark*) a página actual (e.g. adicionar aos favoritos) e voltar a entrar na mesma com base num URI.

Como irão reagir os utilizadores a estas diferenças?

A resposta virá com o tempo e com as soluções criativas que encontrarmos para auxiliar e apoiar os utilizadores a tirarem partido das novas evoluções sem prejudicar a experiência utilização da geração emergente de aplicações baseadas em Ajax.

3.3.3 O Desktop

Pela convergência das aplicações Web através da transposição de paradigmas de interacção típicos de Desktop, muitos anunciaram a fim das aplicações de Desktop. Felizmente, como podemos verificar existem um conjunto de restrições e pressupostos base que não permitem a convergência completa entre os dois mundos, Web e desktop.

Ambos os mundos continuam a evoluir, contudo o Ajax permitiu encurtar as diferenças e tornar mais consistente a transposição de determinados modelos mentais e paradigmas de interacção de um mundo para o outro.

Neste momento a visão mais provável para o futuro de aplicações de software passa por uma abordagem híbrida:

- Para interacções avançadas e utilizações de longa direcção, as aplicações Desktop (AD) continuaram a representar alternativa mais adequada.
- Para aplicações mais leves e cujo factor preponderante seja a ubiquidade e rapidez de utilização, as aplicações Web (Web) constituirão a alternativa mais adequada.

3.3.4 Flash

O Adobe flash desempenha um papel interessante no mundo Ajax. O Ajax apresenta vantagens ao nível do respeito e suporte dos standards. Por outro lado, o Flash permite acesso ao hardware (e.g. câmara Web, microfone, placa de som), gráficos e

animações sofisticadas, armazenamento local e funcionamento com ou sem conectividade com a Web.

Ambas as tecnologias estão em constante evolução e não é possível prever qual delas sairá vencedora neste duelo. Na minha perspectiva, o futuro passará mais pela complementaridade das tecnologias, do que pela anulação de uma delas. No entanto, o facto de o Flash ser uma tecnologia fechada ao contrário do Ajax, que tem por base tecnologias e convenções standard, poderá garantir ao Ajax alguma vantagem. Como referimos anteriormente, tecnologias que respeitem os standards e convenções permitem uma maturação mais rápida e consistente, assim como, atraem um maior universo de empresas e pessoas dispostos a investir e desenvolver com base nessas tecnologias.

3.3.5 Two-Way Web

Dê por onde der, na Web é sempre o Browser que despoleta a chamada ao servidor. Actualmente não é possível ao servidor, abrir o Browser no cliente e apresentar uma mensagem.

Ainda que esta não seja uma questão central para o Ajax, começa a surgir um grande interesse sobre este tipo de interacção recorrendo às tecnologias chave presentes na arquitectura Ajax. As soluções começam a surgir e ainda que requeiram inovações tecnológicas significativas, vale a pena estar atento a projectos como o Twisted⁵⁷, um servidor que suporta ligações de longa duração.



Figura 26: Twisted – Event Driven Networking Engine

3.3.6 Armazenamento local de dados

À medida que a adopção do Ajax se vai disseminando, novos requisitos funcionais vão surgindo. A possibilidade de armazenamento local de dados, tem sido algo que as actuais tecnologias Web não conseguiram solucionar. Actualmente, o armazenamento de dados do lado do cliente, é apenas possível com recurso a *Cookies*. Todavia, os *Cookies* não permitem o armazenamento de grandes blocos de informação (aproximadamente até 4Kb de informação [46]), bem como, não prevêm a possibilidade de a informação local não ser comunicada ao servidor.

⁵⁷ Twisted, site oficial, <http://twistedmatrix.com/trac/>, Acesso: 2009-03-14

Todavia, o maior desafio que se coloca para as RIA orientadas à Web, prende-se com a possibilidade de funcionarem *on* e *offline*. Eventualmente os Browsers poderão vir a desempenhar um papel importante para esta possibilidade ser viável, ***mas a que custo?*** Tanto a Microsoft como a Mozilla já incluíram esta funcionalidade no *roadmap* de desenvolvimento dos seus Browsers. A questão fundamental agora, passa por definir se será uma funcionalidade standard ou específica de cada Browser.

Outro desafio tecnológico estará relacionado com a sincronização à posteriori da informação (após ser trabalhada *offline*) e, por fim mas não menos importante, o impacto nos modelos mentais do utilizador relativo à percepção sobre se a informação está armazenada local ou remotamente.

Como será de prever as evoluções tecnológicas implicam sempre mudanças e compromissos. O desafio está em tentar perceber quais as mudanças que os utilizadores estão dispostos a aceitar em prole de uma melhor experiência de utilização.

3.3.7 XForms

XForms é um novo standard, definido pelo W3C, para a formulários na Web. Conforme presente na descrição do W3C [47]:

“XForms is an XML application that represents the next generation of forms for the Web. XForms is not a free-standing document type, but is intended to be integrated into other markup languages, such as XHTML, ODF or SVG. (...)An XForm allows processing of data to occur using three mechanisms:

- a declarative **model** composed of formulae for data calculations and constraints, data type and other property declarations, and data submission parameters*
- a **view** layer composed of intent-based user interface controls*
- an imperative **controller** for orchestrating data manipulations, interactions between the model and view layers, and data submissions.”*

Resumidamente, o standard XForms pretende possibilitar a separação dos dados da informação de apresentação dos mesmos, melhorar a capacidade de manipulação dos dados e permitir uma maior interactividade através da actualização de elementos do formulário em resultado a acções do utilizador noutros elementos. Algo também possível através do Ajax.

As questões que se colocam agora é qual dos paradigmas terá mais sucesso e, ainda que o Ajax seja neste momento mais conhecido dos programadores, o facto de o XForms ser uma standard W3C poderá potenciar a sua utilização. Mais uma vez, mais do que se anularem uma à outra, a possibilidade de XForms e o Ajax poderem coexistir não deve ser colocada de parte.

3.3.8 JavaScript

Com a emergência do Ajax, os programadores começaram a prestar mais atenção ao JavaScript. De linguagem de segundo plano, passou a ser conotada como uma linguagem essencial para o desenvolvimento de aplicações Web. Esta renovada atenção pelo JavaScript impele a linguagem para nova fase de maturação onde é provável que as suas funcionalidades base sejam estendidas à medida que novos requisitos vão sendo lançados pelos programadores.

Actualmente existem já alguns melhoramentos em previstos: suporte para multi-threading e vários melhoramentos ao nível da sintaxe base da linguagem [47].

Outro aspecto que se poderá evoluir nos próximos anos, será a utilização de JavaScript como linguagem do lado do servidor. Actualmente, a maior valia do JavaScript reside no facto de ser uma linguagem que corre no Browser do lado do cliente. Este crescimento poderá contribuir para implementações mais complexas, mas simultaneamente mais consistentes, de aplicações Web que necessitam de processamento de ambos os lados da rede – *Servidor e Cliente*.

Confirma-se por isso a tendência de evolução do JavaScript paralelamente à sua utilização em aplicações Ajax. Podemos afirmar que a coexistência das duas está assegurada, uma vez que o sucesso de ambas as tecnologias será maior quando utilizadas de forma integrada.

3.3.9 Suporte ao Desenvolvimento

Existe já um vasto universo de Bibliotecas e Frameworks de suporte ao desenvolvimento de aplicações Ajax (ver ponto 3.4). É também provável que à medida que o Ajax vá ganhando popularidade entre os programadores mais, melhores e mais sofisticadas bibliotecas possam aparecer.

Por outro lado será fundamental que IDE's como o NetBeans ou Eclipse ofereçam suporte para Ajax e tecnologias adjacentes, como actualmente o fazem para tecnologias como Java e PHP. Funcionalidades como *refactoring* de código, ajuda à escrita de código (IntelliSense – sugestões automáticas de código) e *debugging*.

Ainda assim, um novo conjunto de ferramentas será necessário para suportar o desenvolvimento de aplicações Ajax. Devido à sua natureza de comunicação com a rede, o desenvolvimento de aplicações Ajax poderá beneficiar com ferramentas que permitam: visualização de tráfego, pedidos ao e respostas do servidor e soluções para automatização de testes.

3.3.10 Maior largura de banda

Ainda que tenhamos assistido ao surgimento de aplicações revolucionárias como o Google Suggest (<http://www.google.com/ig?complete=1&hl=en>), que mostrou ser possível a questionar e receber um resposta do servidor à medida que vamos

carregando em cada tecla, o advento de maior largura de banda poderá permitir o desenvolvimento de funcionalidades consideráveis impraticáveis na Web.

Será o Ajax uma boa ferramenta para aplicações de trabalho colaborativo, jogos em rede ou visualização interactiva de dados em constante actualização? A resposta a estas perguntas irão começar a chegar à medida que a largura de banda, assim como, como toda a infra-estrutura de rede (e.g. servidores e máquinas cliente) e os próprios Browsers forem evoluídos.

No centro destes ingredientes estarão também as tendências e relevância que a indústria de software depositar na utilização da tecnologia ou conjunto de tecnologias denominadas por Ajax.

Em conclusão

Após a leitura dos pontos anteriores é natural uma sensação de incerteza sobre o futuro e evolução do Ajax. De facto, esta incerteza representa um fenómeno bastante comum no mundo das tecnologias. Para a indústria do software, esperar para que todas as incertezas de esclareçam representa chegar mais tarde e, perder com isso, a título de inovador o revolucionário.

Por outro lado uma utilização mais conservadora do Ajax representará uma evolução mais lenta mas mais segura e consistente, uma vez que poderá beneficiar dos casos de estudo que vão surgindo, conseguindo dessa forma replicar modelos de sucesso e evitar erros cometidos por outros.

O Ajax já existe há algum tempo na Web. Ainda assim a sua etapa de maturação está apenas no princípio estando longe de uma disseminação global. Existem ainda vários desafios e oportunidades que têm de ser solucionados. Para todos os efeitos, o caminho por nós recomendado passa pelo acompanhamento dos standards e convenções ou tendências globais. O caminho menos aconselhado passa pelo desenvolvimento de aplicações com base em funcionalidades e características específicas de um determinado Browser ou plataforma.

Consideramos por isso que a chave para o futuro do Ajax passa pela **Ubiquidade e integração com tecnologias standard** e convencionadas por grupos acima de interesses meramente económico ou comerciais.

3.4 Bibliotecas e Frameworks Ajax

Actualmente existem um alargado número de bibliotecas e Frameworks Ajax que contribuem para o desenvolvimento mais rápido e eficiente destas aplicações. Um facto igualmente conhecido, é o que o número de bibliotecas de Frameworks têm aumentado significativamente, devido ao crescente interesse pela utilização de Ajax no desenvolvimento de aplicações Web.

Em seguida é apresentada uma síntese das Bibliotecas e Frameworks mais relevantes, segundo um estudo de Dezembro de 2007 da Burton Group⁵⁸:

Quadro 6: Biblioteca e Frameworks Ajax

Biblioteca/Framework	Descrição
Prototype	Framework Javascript que agiliza o desenvolvimento de aplicações Web de interação sofisticada com base em Ajax. http://prototypejs.org/
jQuery	Biblioteca JavaScript que simplifica a comunicação de HTML, gestão de eventos, animação e interações Ajax. http://jquery.com/
Ext	Biblioteca Javascript <i>cross-browsing</i> para o desenvolvimento de aplicações RIA. http://extjs.com/
Script.aculo.us	Utilizada em conjunto com a Framework Prototype, principalmente para o desenvolvimento de animações e interfaces com o utilizador. http://script.aculo.us/
Mootools	Framework JavaScript compacta e modular utilizada frequentemente para efeitos visuais e transições. http://mootools.net/
Dojo Toolkit	Biblioteca modular de JavaScript, desenhada para acelerar o desenvolvimento de aplicações Web, <i>cross-platform</i> , com base em Ajax. http://dojotoolkit.org/

Segundo o mesmo estudo, entre as linguagens de programação mais utilizadas em conjunto com o Ajax, encontra-se o PHP e o JAVA.

3.5 Síntese e conclusões

Este capítulo apresenta uma síntese sobre o universo Ajax e “ecossistema tecnológico” em este está inserido. Para além de apresentar uma descrição detalhada sobre as tecnologias que constituem e que influenciam uma aplicação Ajax, aponta também os desafios e oportunidades que esta tecnologia terá de resolver ao longo do seu processo de maturação e disseminação. As perspectivas apresentam-se optimistas devido ao respeito pelos standards e alinhamento com as tendências e convenções globais para a Web. Do ponto de vista tecnológico e do impacto para os utilizadores, o Ajax é uma tecnologia necessária e apetecível.

⁵⁸ Burton Group annual survey - December 2007, http://www.surveymonkey.com/sr.aspx?sm=fXLIKcnKID6cO5bRe961aBB6NCCWytRyY3rParAYmwa_3d, Acesso: 2009-03-23

4 Padrões Ajax

Neste capítulo iremos introduzir a temática - Padrões – e tentar descrever de que forma podem e estão a ser utilizados na indústria da Engenharia de Software e das mais-valias que representam.

Ainda no âmbito deste capítulo será apresentada uma visão macro sobre a arquitectura Ajax do ponto de vista dos Padrões que contribuem para essa arquitectura. Por fim serão identificados os Padrões Ajax que mais contribuem para as inovações proporcionadas pelo Ajax ao nível da interacção com o utilizador.

Antes de entrar em mais pormenores, aproveito para desmistificar o que é um Padrão:

Os padrões são uma descrição textual de uma solução genérica para um problema recorrente num determinado contexto.

No âmbito deste trabalho o contexto a considerar é:

Desenvolvimento de aplicações Web recorrendo a Ajax, com especial enfoque na Interacção com o Utilizador.

A noção de Padrão tem origem no trabalho do arquitecto Cristopher Alexander⁵⁹. Durante 10 anos, Cristopher recolheu e documentou exemplos de soluções genéricas para problemas recorrentes na arquitectura.

Em 1997 publicou o livro “A *Pattern Language*” [50], onde apresenta 253 Padrões para problemas recorrentes no desenho e construção de edifícios.

“(...) It describes exact methods for constructing practical, safe and attractive designs at every scale, from entire regions, through cities, neighborhoods, gardens, buildings, rooms, built-in furniture, and fixtures down to the level of doorknobs. (...)” [50]

Em seguida apresentamos um exemplo curioso (“83. *Master and Apprentices*”) retirado do livro “A *Pattern Language*”, como demonstração simplificada do conceito de Padrão:

- Problema
 - “The fundamental learning situation is one in which a person learns by helping someone who really knows what he is doing.” [50]
- Solução
 - “Arrange the work in every workgroup, industry, and office, in such a way that work and learning go forward hand in hand. Treat every piece of work as an

⁵⁹ Cristopher Alexandre, Wikipedia, http://en.wikipedia.org/wiki/Christopher_Alexander, Acesso: 2009-03-14

opportunity for learning. To this end, organize work around a tradition of masters and apprentices: and support this form of social organization with a division of the workspace into spacial clusters - one for each master and his apprentices - where they can work and meet together.” [50]

4.1 Padrões de Software

“Um padrão de software descreve um problema recorrente num contexto específico e uma solução genérica comprovadamente boa para o resolver. A solução descreve os componentes que a constituem, as suas responsabilidades, as associações e os mecanismos de colaboração entre eles” [51]

Apoiado na definição anterior, Ademar Aguiar [49] acrescenta que os padrões de software representam por isso

“Uma forma de partilhar experiência”, uma vez que proporcionam uma forma consistente e uniformizada de troca de informação entre profissionais de engenharia de Software. E se não podemos *aprender trabalhando com os mestres* (“83. *Master and Apprentices*”), podemos no mínimo aprender com o conhecimento (problemas, decisões e possíveis soluções) encapsulado num padrão.

“Documentam soluções de qualidade para problemas típicos”, se alguém se dá ao trabalho de escrever um padrão de software é porque investiu o tempo necessário na sistematização de um problema. Quando nos deparamos com um problema em Engenharia de Software muito provavelmente já outros o sentiram. Se assim é, porque não beneficiar com as soluções e esforço já realizado por outros na tentativa de encontrar uma solução para esse problema.

“O património actual é bastante grande e continua a aumentar.” Da mesma forma que num dicionário são adicionadas novas palavras sempre que adquirem importância para tal, o universo de Padrões de Software continua a crescer à medida que novos problemas e soluções vão sendo elencados. A possibilidade de compilar este conhecimento num formato consistente e uniformizado para todo o mundo, potência de uma forma incrível a circulação e disseminação de conhecimento.

“Da curiosidade à familiaridade”, após desmistificada a utilização de Padrões de Software, os profissionais da Engenharia de Software começam agora a recorrer com maior frequência às bibliotecas de Padrões. Não se trata de curiosidade mas sim de **produtividade**. Como profissional invisto o meu tempo a tomar conhecimento de soluções propostas pelos outros e, a partir daí, desenho a minha própria solução. Ganho tempo e melhora a qualidade das soluções que proponho e desenvolvo.

Como e quando aplicar Padrões?

Em Engenharia de Software os Padrões estão organizados basicamente em 3 grupos [52]:

- **Architectural Patterns**, descrevem a organização da estrutura fundamental de um sistema de software. Identificam os subsistemas existentes, especificam as suas responsabilidades e descrevem como as boas práticas na organização das relações entre eles.
- **Design Patterns**, descrevem estruturas recorrentes de comunicação entre componentes de um sistema de software, na tentativa de resolução de um determinado problema, num determinado contexto.
- **Idioms**, representam padrões, com um grau de abstracção mais baixo, específicos de uma determinada linguagem de programação.

A principal diferença entre os 3 grupos de Padrões apresentados é o nível de abstracção e detalhe que descrevem. Os *Architectural Patterns* descrevem estratégias de elevado grau de abstracção que correspondem à visão macro do componente ou sistema. Os problemas e soluções indicados estão tipicamente relacionados com a organização e esqueleto de um sistema.

Por outro lado, os *Design Patterns* apresentam um grau de abstracção intermédio, uma vez que se centram na resolução de problemas relacionados com o comportamento de determinados componentes do sistema, ou nas relações entre componentes. Estes padrões definem *micro-arquitecturas* que resolvem problemas específicos, mas não influenciam a arquitectura global do sistema.

Por fim, os *Idioms* representam técnicas específicas de determinadas linguagens para resolução de problemas específicos que influenciam o comportamento interno de um componente do sistema. São indicados por isso como padrões com baixo nível de abstracção.

Utilização de Padrões

Como referido no ponto anterior, os padrões apresentam diferentes graus de abstracção. Da mesma forma, fornecem soluções em abstracto para determinados problemas. Nesse sentido, quem recorre aos Padrões de Software, deve ter presente que:

- Os Padrões de Software têm de ser *adaptados e instanciados ao contexto* específico em que vão ser aplicados;
- Com a utilização de um Padrão, normalmente recorre-se a outros padrões para complementar a solução pretendida;

Da mesma forma que apresentam uma visão pormenorizada de um problema, os Padrões facilitam também a avaliação de compromissos relativamente à adopção de um Padrão:

- Expõem uma solução, demonstram as forças envolvidas e de que forma se podem contornar ou potenciar;
- Auxiliam a tomar decisões de desenho e de implementação através de alertas para os compromissos inerentes à adopção de cada Padrão;

Mais-valias

Como referido anteriormente, os Padrões beneficiam a indústria do Software uma vez que:

- *Veiculam sabedoria entre quem desenvolve software*, através da documentação e partilha de soluções comprovadas para problemas típicos da Engenharia de Software.
- *Vocabulário comum*, permitindo melhorar a eficiência da comunicação entre profissionais da indústria do Software, através da criação de um vocabulário partilhado por todos. Através do nome de um Padrão, os profissionais conseguem *descrever um problema* e/ou sugerir uma *solução específica*.
- *Uma forma mais rápida de aprender na prática*, uma vez que os utilizadores menos experientes têm acesso a conhecimento sistematizado por utilizadores mais experientes. Todavia, não se recomenda a utilização cega das soluções indicadas. O programador, por maior ou menor experiência que tenha, deverá sempre questionar sobre qual a solução que melhor se adequa ao seu problema.

4.2 Padrões Ajax

Os Padrões Ajax pretendem descrever a forma e princípios com que os utilizadores têm utilizado esta tecnologia no desenvolvimento de aplicações Web. Pode até parecer estranho existirem tantos Padrões sobre um fenómeno tecnológico relativamente recente. De facto já antes de existir o termo Ajax para descrever esta arquitectura, existiam ideias e implementações dos típicos *early adopters*, que ajudaram a crescer e a transformar o Ajax numa arquitectura bem definida e consistente.

Com mais de 60 Padrões Ajax conhecidos, torna-se fundamental a sua organização hierárquica. Nesse sentido, com base nos trabalhos: *Ajax Design patterns: Creating Web 2.0 Sites with Programming and Usability Patterns* [3] e *Ajax Patterns and Best Practices* [4]; apresentamos uma organização com base em 4 áreas de especificidade:

- **Fundamentos Tecnológicos**, enquadra os Padrões estruturantes que diferenciam o Ajax das abordagens tradicionais para desenvolvimento de aplicações Web;
- **Programação**, enquadra Padrões que descrevem soluções de código e arquitectura que permitem a implementação dos princípios indicados pelos *Fundamentos Tecnológicos* (e.g. gerir fluxos de informação entre o servidor e o browser, actualizar a estrutura DOM quando chega a resposta do servidor),
- **Funcionalidade e Usabilidade**, apresenta Padrões com impacto directo nas interfaces gráficas e paradigmas de interacção com o utilizador (e.g. *drag and drop*, *direct manipulation*, efeitos visuais);
- **Desenvolvimento**, enquadra Padrões a adoptar no processo de desenvolvimento. Ao contrário das áreas anteriores que pretendem representar aspectos que fazem parte da solução Ajax, nesta área são sugeridas regras e boas práticas para o desenvolvimento de aplicações Ajax (e.g. diagnóstico de problemas, desenho e execução de testes)

Desta forma, as 3 áreas iniciais apresentam Padrões que inserem sobre a **aplicação Ajax** em si, enquanto a última área incide sobre Padrões relevantes para o **processo de desenvolvimento** dessas aplicações.

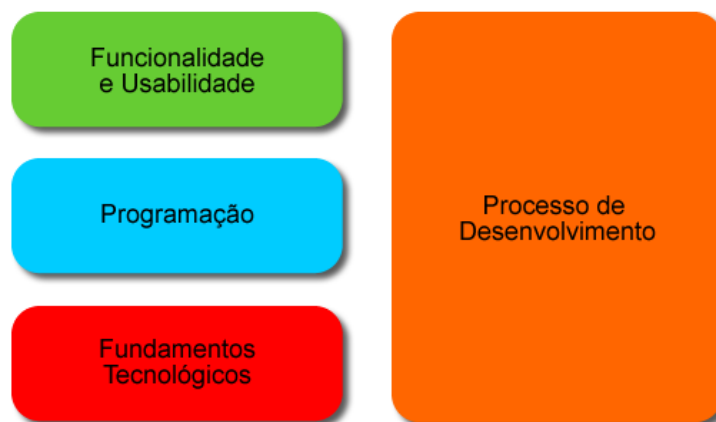


Figura 27: Padrões Ajax: Áreas de especialização

A Figura 27 apresenta uma representação visual sobre as 4 áreas de especificidade dos Padrões Ajax. Os Padrões em Fundamentos Tecnológicos tentam descrever a utilização das tecnologias e mecanismos na essência do Ajax (e.g. DOM e XMLHttpRequest) e estão na base de qualquer aplicação Ajax. A um nível intermédio temos os Padrões de Programação que apresentam soluções sobre como utilizar as tecnologias Ajax. No nível superior temos os Padrões relativos às soluções com impacto na interacção com o utilizador.

No lado oposto (lado direito da Figura 27), surgem então as regras e boas práticas sugeridas para o desenvolvimento de aplicações Ajax.

Cada área de especialidade contém um conjunto de Padrões Ajax, dos quais iremos seleccionar, descrever e exemplificar os mais relevantes na base das inovações introduzidas pelo Ajax, no que respeito diz à **Interação com o Utilizador** em aplicações Web.

4.3 Padrões Ajax centrados na Interação com o Utilizador

Da análise dos Padrões Ajax conhecidos, resultou a identificação dos Padrões que mais contribuem para as tendências de convergência entre aplicações Web e Desktop, com especial enfoque nos Padrões com impacto na Interação com o Utilizador.

De seguida é apresentada a lista de Padrões que iremos detalhar com maior pormenor no próximo capítulo.

Quadro 7: Padrões AJAX centrados na interação com o utilizador

PADRÃO	SUMÁRIO
Display Morphing	Manipulação do aspecto visual de elementos na estrutura DOM, com impacto imediato na representação do Browser
Page Rearrangement	Manipulação directa da estrutura DOM com impacto imediato dos elementos apresentados pelo Browser
XMLHttpRequest Call	Mecanismo de comunicação (envio e recepção de dados) com o servidor, invisível e não bloqueante da interação com utilizador.
User action	Detecção e captura de acções do utilizador, realizadas através do teclado, do rato ou outro dispositivo de posicionamento do cursor.
Scheduling	Agendamento automático da realização de tarefas, sendo possível a indicação sobre a recorrência da tarefa.
XML Message	Estrutura da informação veiculada nas mensagens entre Servidor e Browser e vice-versa.
Submission Throttling	Gestão de uma lista de pedidos gerados pelas acções do utilizador, com vista à garantia da performance do servidor.
Progress Indicator	Feedback ao utilizador sobre um processo de comunicação em curso entre Browser e Servidor.

Rich Text Editor	Ferramenta de apoio ao utilizador na introdução e edição de conteúdos, permitindo a aplicação de estilos e visualização imediata do impacto do formato no conteúdo.
Drag and Drop	Possibilidade de mudar a posição de um elemento na página, podendo com isso mudar as relações ou estado do elemento.
Malleable Content	Mecanismo que permite a criação e edição de um conteúdo directamente numa página de consulta, sem necessitar por isso, de uma página específica para o efeito.
Microlink	Aparecimento suave numa página de informação adicional sobre um conteúdo seleccionado pelo utilizador.
Highlight	Destaque de elementos presentes na página, no sentido de indicar uma mudança de estado ou selecção para aplicação de determinada acção.

Os Padrões indicados foram seleccionados pelo autor de um universo de 60 Padrões Ajax disponíveis na bibliografia e referências na Web indicadas neste trabalho. Para a selecção dos Padrões indicados, recorreremos aos seguintes critérios:

- **Evidência do Padrão**, disseminação da utilização do Padrão em aplicações actuais;
- **Impacto na interacção com o utilizador**, alterações introduzidas pelo padrão visíveis para o utilizador;
- **Alinhamento com tendências de convergência**, migração de aspectos típicos de aplicações Desktop e respectiva adaptação para a Web.

4.4 Anatomia de um Padrão

Antes de avançarmos directamente para o detalhe dos Padrões seleccionados, será descrita a anatomia de um Padrão.

A anatomia de um Padrão representa uma estrutura base de aspectos a abordar em cada Padrão. Alguns desses aspectos podem ser descartados caso não sejam aplicáveis ao Padrão em questão.

Em seguida será apresentada e explicada a estrutura base, utilizada neste trabalho, para a descrição dos Padrões Ajax com centrados na Interacção com o Utilizador.

Evidência

Apresenta o grau de disseminação pela Web do Padrão em questão. Tipicamente é apresentada com recurso a uma escala visual de 1 a 3 valores, em que o nível 1 representa pouca evidência e 3 maior evidência.

Ainda que representa uma visão subjectiva do autor, os Padrões descritos neste trabalho, de acordo com os critérios de selecção, apresentam todos o nível 3 de evidência, isto é, encontram-se largamente disseminados nas aplicações Ajax disponíveis actualmente.

Palavras-chave

Conceitos e tópicos chave utilizados na descrição do Padrão. Auxiliam na pesquisa e na percepção do contexto global do Padrão.

Esboço

Representação visual com os aspectos mais relevantes do Padrão. Tipicamente serão utilizados os elementos visuais presentes na Figura 28.

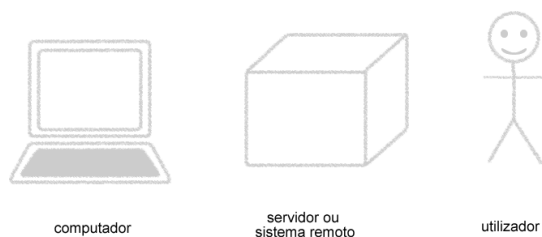


Figura 28: Elementos presentes nos esboços dos padrões

Narrativa

Descrição de um cenário típico, incluindo os principais intervenientes e acções, cuja utilização do Padrão poderá introduzir benefícios e resolver problemas de implementação.

Problema

O bloqueio ou problema em concreto que se pretende resolver

Forças

Aspectos relevantes para balizamento e caracterização do problema.

Solução

A descrição técnica da solução para o problema apresentado.

Decisões

Desafios e compromissos inerentes à adopção de determinado Padrão.

Exemplos

Exemplos de aplicações em funcionamento na Web actualmente.

Exemplos de código

Apresentação de código chave para a implementação do Padrão.

Alternativas

Indicação de Padrões que possibilitam a resolução do mesmo problema, ou semelhante, com diferentes soluções.

Padrões Relacionados

Padrões relacionados com o actual, que complementam ou resolvem determinados compromissos inerentes à utilização de um determinado Padrão.

Metáfora

Um descrição ou alusão que permita a memorização da essência do Padrão.

Saber Mais

Referências ao material original onde foi recolhida informação para o respectivo padrão.

4.5 Padrão: Aplicação Ajax

Evidência

3 – Amplamente utilizado

Palavras-chave

Ajax, Rich Internet Application, Web App, Web 2.0, XMLHttpRequest, Javascript

Esboço

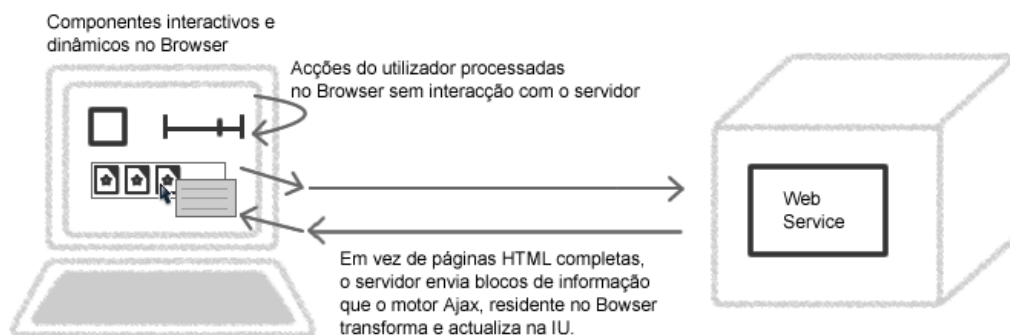


Figura 29: Padrão: Aplicação Ajax

Narrativa

Um utilizador de teste de aplicação Web, começa a testar uma nova aplicação desenvolvida em Ajax. Fica então agradavelmente surpreendida ao verificar que a introdução se realiza com maior eficácia: a informação é validada à medida que vai preenchendo os campos, à medida que vai preenchendo determinados campos são-lhe apresentadas sugestões que contêm as letras já indicadas e os restantes campos do formulário vão-se adaptando de acordo com a informação introduzida em campos anteriores.

Problema

Como criar uma aplicação com interactividade e mecanismos de interacção sofisticados?

Forças

- Utilizadores sentem-se confortáveis com a utilização do Browser e, cada vez mais, disponibilizam-se para introduzir dados directamente nas aplicações

Web. Todavia, as interfaces tradicionais implicam vários cliques e tempos de espera entre acções simples.

- Empresas que investiram em aplicações Web devido à sua facilidade de disponibilização para os utilizadores finais, continuam com dificuldades em transportar experiência de utilização das versões Desktop para as respectivas versões na Web.
- Programadores confortáveis e conhecedores da arquitectura Web, pretendem contribuir para a evolução Web e da sofisticação das aplicações aí existentes.
- A crescente largura de banda e maior capacidade de servidores e infraestrutura física disponível (e.g. capacidade de processamento e armazenamento de dados) impelem o desenvolvimento de aplicações impraticáveis anos anteriores.

Solução

Desenvolver uma aplicação Ajax que corra dentro de um Browser tradicional. O utilizador poderá aceder à aplicação Ajax através da introdução do respectivo Url no Browser, através de um link ou *bookmark*. A aplicação Ajax é então descarregada para o Browser, onde o utilizador pode começar a utilizar imediatamente, não havendo necessidade de aguardar que toda a informação seja descarregada.

As melhorias introduzidas por uma aplicação Ajax reflectem-se principalmente na interacção com o utilizador. Benefícios alinhados com a necessidade e vontade dos utilizadores por uma melhorada experiência de utilização em aplicações Web. Cada vez mais os longos tempos de carregamento de uma página, a perda de dados introduzidos em formulários de preenchimento demorado, representam motivos pelos quais os utilizadores não aderem à utilização destas aplicações. Existem vantagens em continuar a utilizar o Browser, como referido nos capítulos anteriores, no entanto, têm de ser introduzidas inovações ao nível da interacção com o utilizador, no sentido de potenciar o sucesso das aplicações Web.

O Ajax pretende ser um factor potenciador dessa melhorada experiência de utilização, alinhada com standards e constrangimentos dos Browsers actuais. O cumprimento dos standards permite a compatibilidade das aplicações em diferentes Browsers e dispositivo, bem como, fomenta a consistência entre aplicações Web. No final da linha de valor está o utilizador, que beneficia directamente desta consistência.

Tecnologias que tendem a não respeitar os standards quebram a consistência. A utilização de *plugins* como Flash ou JavaApplets deve ser considerada uma solução apenas em situações para os quais não existem soluções baseadas em standards.

As tecnologias referidas como standard incorporadas no Ajax foram apresentadas no capítulo anterior.

Decisões

“Ajax Lite” ou “Ajax Deluxe”?

No que toca ao desenvolvimento de aplicações Ajax existem dois tipos de abordagens distintas.

- **Ajax Deluxe**, representam as aplicações que tiram o máximo partido da tecnologia Ajax. Devido à sua sofisticada interacção com o servidor e com o utilizador, assemelham-se bastante às tradicionais aplicações Desktop. As actualizações completas de toda a página são reduzidas ao mínimo. Evita-se a necessidade de sessões do lado do servidor, uma vez que a informação de *estado* fica no motor JavaScript do lado do Browser. O servidor precisa apenas de disponibilizar informação (e.g. *Web Services*) e a lógica de apresentação do lado do cliente transforma a informação em elementos visuais.
- **Ajax Lite**, representam aplicações mais semelhantes às tradicionais aplicações Web, em que determinadas funcionalidades proporcionam uma experiência de interacção mais sofisticada derivada da implementação de pequenos componentes e mecanismos Ajax. Podemos tomar como exemplo a validação de campos de formulários antes da submissão do formulário, apresentação de alertas, notificações ou ecrãs de apoio às acções do utilizador,

Qual a atitude certa a adoptar? Tudo depende do objectivo e da experiência dos profissionais envolvidos. Uma aplicação *Deluxe* requer uma maior experiência no desenvolvimento com Ajax, conhecimento mais profundo sobre bibliotecas e ferramentas de desenvolvimento apropriadas. De uma forma geral a aplicação Web resultante apresenta uma experiência de utilização mais sofisticada e consistente com o que os utilizadores estão habituados no Desktop. Todavia, a experiência de utilização encontra-se fortemente dependente da largura banda e qualidade da comunicação entre o servidor e o browser.

Uma aplicação Lite, apresenta uma abordagem mais conservadora, utilizando o Ajax apenas para melhorar a interactividade de componentes específicos (e.g. formulários, calendários, efeitos visuais). Costumam ser uma abordagem mais pragmática quando se pretende que a aplicação funcione em Browsers mais antigos.

Quais os requisitos necessários no Browser para correr a aplicação?

Como referido no ponto anterior, uma aplicação Ajax implica que o Browser contemple alguns requisitos de base. Desde 2001 que a maioria dos Browsers apresenta implementações consistentes dos standards utilizados no Ajax. Tipicamente o problema poderá incidir sobre o objecto XMLHttpRequest cuja implementação que pode variar entre Browsers. Mais importante do que investir o desenvolvimento para funcionar em todos os Browsers é definir estrategicamente os Browsers em que a aplicação tem obrigatoriamente de funcionar.

Quantidade de processamento do Browser?

De facto algumas das primeiras aplicações Ajax foram desenvolvidas assumindo que todo o processamento seria realizado no lado do cliente – no Browser. Infelizmente, e uma vez que não estavam preparados para esta utilização intensiva, o motor de interpretação de JavaScript, na maior parte das versões actuais dos Browsers mais representativos, é demasiado lenta, especialmente quando o utilizador tem mais do que uma janela do mesmo Browser aberta.

Ainda assim, as empresas responsáveis pelos Browsers estão atentas a esta evolução e preparam já versões melhoradas dos motores de interpretação de JavaScript. Uma dessas empresas é a Google que referiu explicitamente no desenvolvimento do seu Browser – Chrome; que uma das suas novidades seria um motor de JavaScript mais rápido e capaz de suportar processamento do lado do cliente.

Capacidade de comunicação de rede?

Por outro lado temos a largura de banda da rede (a quantidade de dados que é possível comunicar por segundo) e a respectiva latência (o tempo entre o envio de pedido do Browser até que este chega ao servidor e vice-versa).

Efectivamente através de Ajax é possível reduzir a quantidade de dados a comunicar entre o cliente e o servidor. No entanto, mais importante do que a quantidade de informação transmitida, é o tempo de latência entre o envio de pedido e a sua recepção. Tenhamos em consideração funcionalidades como o Google Suggest que envia um novo pedido ao servidor sempre que o utilizador carrega numa tecla.

Exemplos

Consultar o capítulo 6, deste trabalho, onde se encontra sistematizada a análise e descrição de 5 aplicações Web emergentes que usam Ajax para incrementar significativamente a sua experiência de utilização.

Alternativas

Aplicação Web Convencional, com interactividade limitada.

- **Vantagens Ajax:** menores tempos de espera, mais produtivo menos frustrante para o utilizador; conteúdos das páginas actualizados sem necessário a actualização de toda a página; melhoramento do feedback à introdução de dados; melhoramento global da interacção e interactividade, resultando numa melhor experiencia de utilização.
- **Vantagens Ap. Web tradicional:** os utilizadores estão mais familiarizados; os programadores dominam as técnicas e conhecimentos necessários; compatibilidade com Browsers mais antigos.

Aplicação Flash

Para efeito da análise comparativa, consideremos apenas aplicações completamente Ajax (*Ajax Deluxe*) ou completamente Flash. Actualmente o Flash e o Ajax são alternativas muito próximas, estando a principal diferença no facto de o Flash necessitar da instalação de um *plugin* no Browser e o Ajax requer apenas um Browser que implemente os standards de tecnologias amplamente disseminadas. Seguem-se outras diferenças entre aplicações Ajax e aplicações Flash:

- **Vantagens Ajax:** portabilidade entre Browsers e dispositivos; mais familiares ao utilizadores uma vez que são consistentes com standards disseminados entre os utilizadores; maior abertura para comunicação com outros sistemas ou aplicações; não dependem de tecnologia e *plugin* proprietário; código fonte aberto o que resulta numa mais rápida evolução e disseminação.
- **Vantagens Flash:** suporte de gráficos avançados e vídeo; captura, reprodução e controlo de sinais de áudio; armazenamento local de dados; programação avançada de animações de elementos visuais interactivos;

Java Applet

Eventualmente, para alguns programadores Java, o Ajax represente um retorno a um fenómeno que já viram no passado. Quando apareceram as JavaApplets, foi anunciado o fim das aplicações Desktop ou pelo menos remetido para funções secundárias como apresentações de slides. Efectivamente essa perspectiva não se concretizou por vários motivos e as JavaApplets são muito raras na Web e quando existem pretendem resolver problemas e tarefas bastante específicas. De seguida, elencamos algumas das vantagens comparativas entre ambas as tecnologias:

- **Vantagens Ajax:** carregamento mais rápido; apresentam um *look and feel* consistente com as aplicações Web; componentes Ajax podem ser acrescentados a uma aplicação, enquanto JavaApplets requerem desenvolvimento de raiz de toda a aplicação; dependem de software instalado na máquina cliente (Java Virtual Machine).
- **Vantagens JavaApplet:** em muitas aplicações o Java é a linguagem de programação do lado do servidor, de modo que a sua integração com as JavaApplets é garantida a 100%; muitos programadores estão familiarizados com a arquitectura inerente ao Java, logo não têm de adquirir novas competências; existe um alargado suporte para desenvolvimento de interfaces gráficas típicas de Desktop.

Recentemente a Sun introduziu o *Java Web Start* como uma ponte entre a Web e Desktop. Aplicações Web Start correm no Desktop mas podem ser iniciadas através de um link numa página Web. Os tempos de carregamento foram reduzidos, uma vez que grande parte das Bibliotecas estão armazenadas localmente (partilhadas entre aplicações). Ainda que se apresente como uma boa alternativa, este tipo de soluções não apresenta uma grande expressão na Web actual.

Aplicação Desktop

A derradeira questão que se coloca a quem vai começar a desenvolver uma aplicação – Web ou Desktop. Ao longo deste trabalho fomos abordando diferentes perspectivas e pontos de comparação entre ambos os mundos. De seguida, apresentamos um breve resumo das vantagens comparativas de cada tipo de aplicações.

- **Vantagens Ajax:** não necessita de instalação e está disponível em vários computadores; o utilizador pode aceder à aplicação a partir de qualquer ponto no mundo com acesso à Web; a informação está disponível a partir de qualquer ponto de acesso, uma vez que é armazenada no servidor; actualizações sem necessidade de intervenção do utilizador; a linguagem de programação utilizada no desenvolvimento da aplicação não tem implicações nem restringe o dispositivo ou plataforma em que a aplicação será utilizada.
- **Vantagens Desktop:** permitem o armazenamento local de dados; utilização de som e exploração das potencialidades das placas de som; gráficos avançados tirando partido completo das placas gráficas; atalhos de teclado; acesso directo ao hardware; possibilidade de utilização de outros protocolos de comunicação para além do HTTP; interacção directa com o sistema operativo (e.g. alertas e notificações); maior e mais rápida capacidade de processamento de dados.

Aplicação Mobile

As aplicações para dispositivos móveis estão a evoluir num mundo paralelo ao Ajax. No entanto, a perspectiva de se poder utilizar o Ajax para o desenvolvimento de aplicações Mobile, começa a ganhar força através de projectos como o Opera Platform⁶⁰ - “AJAX gone mobile: Opera Platform brings AJAX applications to the mobile phone”.

Acreditamos também que a importância destas aplicações vá crescer no futuro. Segundo Steve Brat na apresentação sobre “W3C - One Web: Going Mobile ” [53], Steve destaque a plataforma Mobile como estando ao nível dos restantes grupos de interesse actuais no W3C. Como é possível verificar através da Figura 30, os principais grupos de interesse do W3C são:

- **Aplicações Web:** XHTML, XForms, CSS, SVG, SMIL, etc.;
- **Mobile:** XHTML básico, Mobile SVG, XForms básico, CSS Mobile, etc.;
- **Voz:** VoiceXML, SRGS, EMMA, etc.;
- **Web Services:** SOAP, WSDL, MTOM, etc.;
- **Web Semântica:** OWL, SKOS, RDFa, etc.;
- **Privacidade e segurança:** P3P, APPEL, XML Sig, etc.;

⁶⁰ Opera Platform, <http://www.opera.com/press/releases/2005/11/15/>, Acesso: 2009-03-17

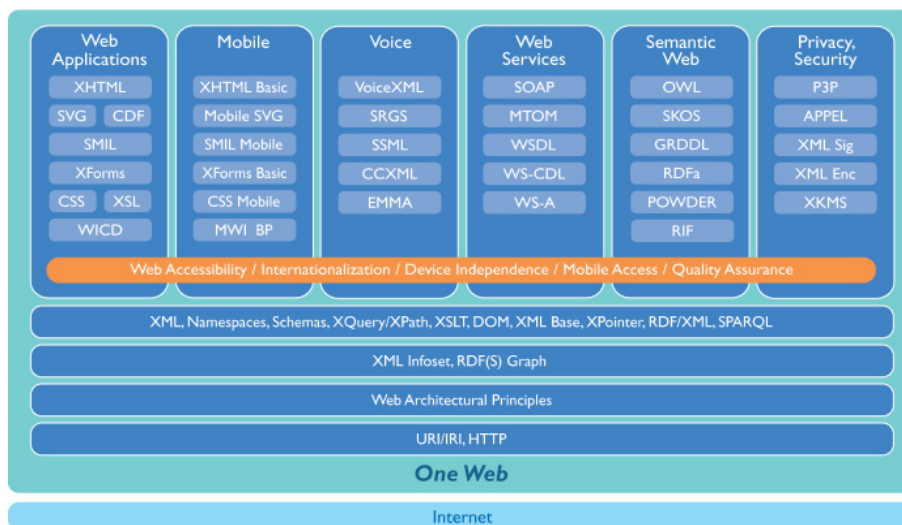


Figura 30: W3C - Grupos de Interesse actuais

Fica por isso presente a forte convicção de que nos próximos anos uma das áreas em forte explosão será o acesso à Web através de dispositivos móveis. E ainda que não seja necessário reinventar todo um conjunto de tecnologias (e.g. XHTML, CSS, SVG, etc), é de vital importância que estas sejam adaptadas às características técnicas dos dispositivos em vista.



Figura 31: Variedade de dispositivos móveis

Padrões Relacionados

Todos os padrões apresentados no capítulo seguinte estão de alguma forma relacionados com o Padrão Aplicação Ajax. De facto, o Padrão Aplicação Ajax é o padrão essencial dos *Fundamentos Tecnológicos* da arquitectura Ajax.

4.6 Síntese e conclusões

Este capítulo apresenta uma breve introdução ao tema dos Padrões e de que forma estes têm sido utilizados na Engenharia de Software para solucionar e melhorar o processo de desenvolvimento de aplicações. Foi também descrita e clarificada a estrutura de uma Padrão e o tipo de informação que deve ser sistematizada para demonstração desse Padrão. Foi também descrita uma proposta de organização dos Padrões Ajax de acordo com a essência e nível de abstracção do problema e solução sistematizada por cada Padrão.

Resulta por isso uma visão objectivo sobre o papel dos Padrões na Engenharia de Software e, neste caso em particular, ajudar a perceber a essência de uma aplicação Ajax.

5 Padrões Ajax centrados na interacção com o Utilizador

Após descrita a estrutura de um Padrão e exemplificado com um Padrão de alto nível “**Aplicação Ajax**”, passamos a padrões mais específicos. Padrões estes que, como referido ao longo deste trabalho, acreditamos estarem na base de um conjunto de melhorias significativas na interactividade e interacção em aplicações Web.

Ainda que estes Padrões estejam amplamente disseminados nas aplicações Desktop e que a sua transposição para a Web seja encarado como um fenómeno natural e evidente da maturação da plataforma Web, ao longo do trabalho fomos defendendo a teoria de que o Ajax desempenha um papel fundamental para a consumação das tendências de convergência apontadas.

Nos próximos pontos iremos descrever e demonstrar de que forma o Ajax torna possível a implementação de paradigmas de interacção típicos de aplicações Desktop, numa plataforma com diferentes pressupostos e constrangimentos tecnológicos – a **Web**.

Os Padrões apresentados neste capítulo, surgem como um refinamento do trabalho realizado por Michael Mahemoff, [3].

A abordagem presente neste trabalho, difere da realizada pelo autor, uma vez apresenta especial enfoque no contributo desses mesmos Padrões para a materialização das tendências de convergência verificadas actualmente nos modelos de interacção com o utilizador em aplicações Web e aplicações Desktop.

5.1 Padrão – Display Morphing

Evidência

3 – Amplamente utilizado

Palavras-chave

Ecrã, DOM, Gráfico, Interface Gráfica para o utilizador (GUI), Transformação, Página, interactividade

Esboço

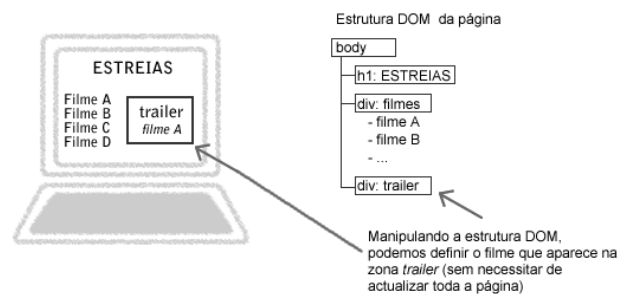


Figura 32: Padrão: Display Morphing

Narrativa

Um apostador entra no seu site favorito de apostas. Entra num evento desportivo a acontecer em tempo real. É-lhe apresentado um cronómetro com o tempo real de jogo sempre a contar, comentários em directo sobre os aspectos mais marcantes a acontecer e continuamente actualizada a lista de apostas disponíveis, onde aparecem assinaladas com cor verde ou vermelho as apostas cuja valor subiu ou desceu.

Problema

Como é possível actualizar dinamicamente os elementos gráficos numa página? (sem proceder à actualização da mesma)

Forças

- É necessário apresentar ao utilizador informação que é constantemente actualizada;
- À medida que o utilizador modifica a informação, a informação anterior é pertinente ou pode ser descartada;

- Em aplicações Web tradicionais, a informação descartável só é removida da página, através da actualização completa da página.
- Submeter a informação para o servidor e receber uma nova página pode quebrar o fluxo de actividade do utilizador.

Solução

Alterar estilos e informação visual (e.g. cor e texto) dos elementos presentes na estrutura DOM da página actual.

A estrutura DOM representa os elementos presentes numa página Web a qualquer instante. Uma alteração nesta estrutura, através de JavaScript, implica a actualização imediata do Browser da representação visual do elemento em causa. Para a alterar um elemento na estrutura DOM basta conhecer uma de duas referências: *id* ou *class*. O *id* identifica univocamente um elemento na página enquanto uma *class* pode identificar um conjunto de elementos. Desta forma, se pretende alterar o aspecto visual de um elemento em particular referencie o seu *Id*. Caso pretenda alterar a informação visual de um conjunto de elementos utilize a referenciação através da *class*.

Desta forma conseguirá manipular o aspecto visual de qualquer elemento presente na página actual (e.g. div, headings, images, etc.).

Decisões

Que tipo de propriedades podem ser manipuladas?

- *Cor (fontColor)*, possibilita a definição da cor a aplicar ao texto presente num determinado elemento;
- *Fundo (backgroundColor, backgroundImage)*, possibilita a definição do tipo de fundo pretendido para um elemento (e.g cor sólida ou imagem);
- *Border (borderWidth, borderColor, borderStyle)*, possibilidade de adicionar e alterar o aspecto visual do contorno aplicado a um elemento;
- *Texto (fontSize, fontWeight, fontStyle)*, possibilidade de alterar o aspecto visual do texto presente num elemento da página;
- *Conteúdo (innerHTML)*, possibilidade de alterar o conteúdo dentro de um determinado elemento da página;
- *Imagens (src)*, possibilidade de alterar a imagem apresentada através da modificação da propriedade *src*;

Exemplo numa aplicação Web

Corrector ortográfico Google Mail

O Gmail possibilita a verificação ortográfica da mensagem escrita. Quando detecta um erro, assinala a palavra com uma cor diferente. Permite também após seleccionar uma palavra assinalada, e verificar as sugestões de correcções apresentadas.

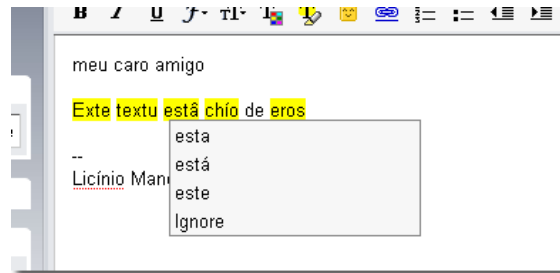


Figura 33: Google Mail: Corrector ortográfico

Padrões Relacionados

Page Rearrangement, está focado na manipulação global dos elementos na estrutura da página, enquanto o *Display Morphing* (Padrão actual) se centra na alteração do aspecto visual de um elemento. Em conjunto os dois Padrões abordam todos os aspectos relacionados com a manipulação dos elementos visuais de uma página.

Metáfora

Display Morphing, é como utilizar uma varinha mágica que nos permite mudar os aspectos de cada elemento na página Web.

Saber Mais

- innerHTML VS DOM, discussão entre Mr Alex Russell e Mr Tim Scarfe⁶¹, sobre as vantagens e desvantagens da utilização do innerHTML em vez dos métodos disponibilizados pela estrutura DOM.

⁶¹ innerHTML VS DOM, <http://www.developer-x.com/content/innerHTML/>, Acesso: 2009-03-17

5.2 Padrão – Page Rearrangement

Evidencia

3 – Amplamente utilizado

Palavras-chave

Manipular, Adicionar, Remover, Alterar, Mover, DOM, Sobrepor, Reposicionar, Reestruturar

Esboço

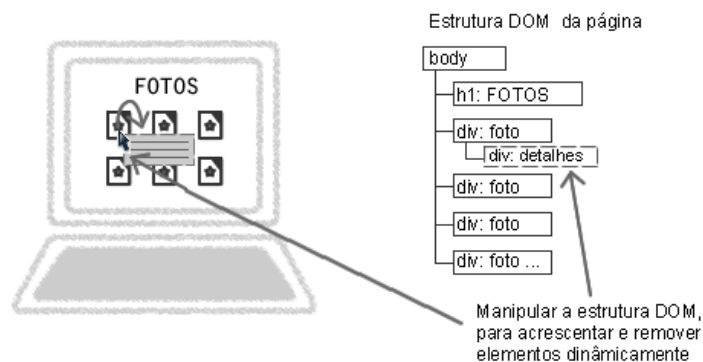


Figura 34: Padrão: Page Rearrangement

Narrativa

Um viajante aproveita para ver o álbum online da sua última viagem. Assim que chega ao álbum são-lhe apresentadas as fotografias tiradas e quando passa o rato sobre cada fotografia é-lhe apresentada um painel data hora e local onde foi tirada. Ainda no painel de detalhes da foto o utilizador pode acrescentar e alterar a informação existente.

Problema

Como é possível manipular de forma dinâmica estrutura da página Web?

Forças

- As acções do utilizador implicam a apresentação de nova informação.
- As alterações à estrutura da página devem acontecer de forma suave, consistente e coerente.

- Actualizar toda a página quebra a continuidade do fluxo de utilização, bem como, a informação de estado do JavaScript.
- A actualização completa da página implica alterações bruscas da posição ou visibilidade dos elementos, não permite transições suaves entre estados e posições.

Solução

Adicionar, remover, mover e sobrepor elementos através da manipulação da estrutura DOM. A estrutura DOM representa a estrutura hierárquica dos elementos presentes numa página Web. Nest estrutura todos os elementos estão relacionados uns com os outros (e.g. *node*, *parentNode*, *childNodes*). Através da manipulação da estrutura DOM, é possível alterar a visibilidade e posicionamento de elementos na página Web.

- **Adicionar**, permite criar dinamicamente um novo elemento filho (*child*), inserido no contexto de um elemento já existente (*parent*). A acção contrária, remover elemento, é igualmente possível.
- **Display**, utilizado para fazer um elemento, que já existe na estrutura DOM, aparecer e desaparecer. Permite também definir a forma como o elemento deve ser representado pelo Browser (e.g. *block*, *inline*, *none*, etc.).
- **Mover**, permite mudar a posição hierárquica de um elemento na estrutura DOM, bem como, alterar o posicionamento do mesmo na janela do Browser.
- **Sobrepor (overlay)**, através de CSS é possível definir a profundidade de um elemento (*zIndex*). Entenda-se por profundidade a posição de elemento quando sobreposto aos restantes elementos presentes na página.

Decisões

Que estratégia de posicionamento de elementos utilizar?

Uma aplicação pode utilizar diferentes estratégias de posicionamento de elementos na página Web. O tipo de posicionamento por defeito é – *Static* – e na maior parte dos casos é suficiente. Outros tipos de posicionamento são normalmente utilizados quando se pretende a utilização de elementos flutuantes. Para este tipo de solução o tipo de posicionamento recomendado é – *Relative* – que permite o posicionamento de um elemento dentro de um *container* (elemento na estrutura DOM que serve de contexto para a existência de outro).

Como prevenir a falta de memória (RAM)?

O acto de adicionar e remover continuamente elementos pode resultar numa gestão pouco eficiente da memória do sistema, podendo, em casos muito críticos, causar erros por falta de memória. Efectivamente o JavaScript tem mecanismos de *garbage*

collection (automatismos da linguagem de programação que eliminam de memória variáveis e objectos que deixaram de ser referenciados). Estes mecanismos nem sempre funcionam da forma esperada, devido ao motor JavaScript de cada Browser. Ficam algumas sugestões para evitar o problema da falta de memória:

- Evitar variáveis globais.
- Para eliminar uma variável, deve atribui-lhe o valor *null*.
- Evitar ou destruir as referências circulares.
- Testar, testar e testar.

Exemplo numa aplicação Web

Kiko Calendar

Após adicionar um compromisso ao calendário, ao passar com o rato sobre o compromisso é-nos apresentada uma nova área (na Figura 35, área com fundo verde) com os detalhes desse compromisso. O utilizador pode também reajustar o compromisso aumentando a sua duração, bem como, a hora de início e o dia de realização.

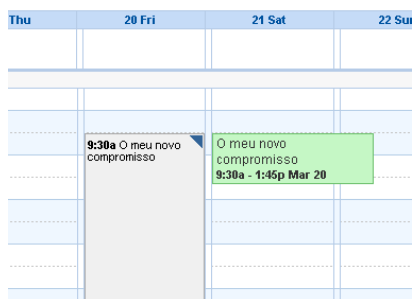


Figura 35: Kiko Calendar: detalhes de entrada no calendário

Padrões Relacionados

Padrão Display Morphing

O *Padrão Display Morphing*, centra-se no aspecto visual dos elementos enquanto o *Page Rearrangement* foca-se no posicionamento e visibilidade dos elementos. Como referido anteriormente, em conjunto os dois Padrões abordam todos os aspectos relacionados com a manipulação dos elementos visuais de uma página.

Metáfora

Page Rearrangement é como adicionar, remover e organizar a disposição de um conjunto de *post-its* num quadro.

5.3 Padrão – XMLHttpRequest Call

Evidencia

3 – Amplamente utilizado

Palavras-chave

Chamada, *Callback*, *Download*, *Live*, Interrupção, *Remoting*, Scripting Remoto, Sincronismo, Sincronizar, *Upload*, *XMLHttpRequest*

Esboço

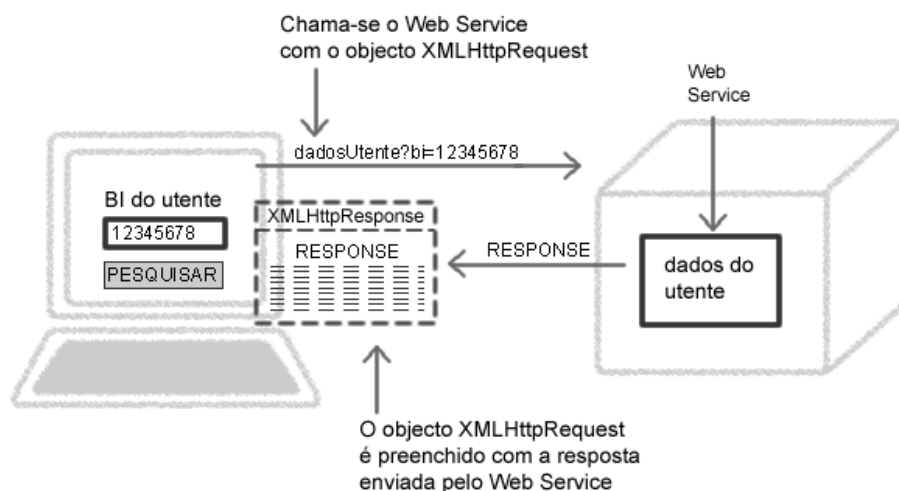


Figura 36: Padrão: XMLHttpRequest Call

Narrativa

Uma dona de casa está a fazer compras no super mercado online. De cada vez que adiciona um item ao seu carrinho de compras, o Web site envia um XMLHttpRequest com a última informação recolhida. O item aparece instantaneamente no carrinho de compras sem ser necessário submeter qualquer formulário, o que ajuda a dona de casa a ter feedback imediato da sua acção e sem ver a interacção interrompida por uma actualização completa da página.

Problema

Como é possível comunicar com o servidor de forma transparente para o utilizador?

Forças

- As aplicações Ajax implicam comunicação com o servidor. A informação gerada pelo utilizador tem de ser enviada para o servidor e a respectiva resposta adicionada à página Web.
- A aplicação Ajax deverá proporcionar ao utilizador uma interacção contínua sem interrupções para actualização completa de toda a página.
- Uma aplicação Ajax deve apresentar feedback imediato ao utilizador, de modo que a transferência de dados entre o servidor e o browser deve ser reduzida ao mínimo.
- A disponibilidade da rede varia, de modo que uma aplicação Ajax deverá permitir que o utilizador continue a interagir com a aplicação enquanto aguarda pela resposta do servidor.

Solução

Utilize o objecto XMLHttpRequest para comunicação entre o Browser e o Servidor. Por restrições de segurança (que provavelmente se irão manter), o JavaScript não dispõem de objectos para estabelecer comunicações através da rede. Graças ao objecto XMLHttpRequest, disponível na maioria dos Browsers actuais, é possível recorrer ao JavaScript para implementar chamadas HTTP ao servidor original e recolher a resposta do mesmo.

Desta forma, torna-se possível estabelecer ligação ao servidor (e.g. enviar informação, pedir informação) de forma transparente para o utilizador, quebrando o modelo tradicional em aplicações Web – pedido e resposta.

Decisões

Em que formato vêm as respostas do servidor?

Tipicamente as respostas do servidor poderão ser baseadas em XML. No entanto existem outros formatos de informação que podem ser passados através do XMLHttpRequest (e.g. HTML, JSON ou texto simples). O requisito essencial é que seja possível o processamento da informação através de JavaScript.

Como deverá ser gerida a *cache* do Browser?

É possível que algumas das chamadas XMLHttpRequest fiquem na *cache* do Browser. Por vezes é o comportamento pretendido mas outras vezes não, de forma que é necessário implementar mecanismos que permitam gerir esta situação.

Quanto se estabelece uma ligação com o servidor através de XMLHttpRequest é necessário indicar o tipo de ligação - GET ou POST. As ligações do tipo GET podem ficar na cache do Browser, o mesmo não acontece com as ligações POST.

Todavia quando a solução passa por uma ligação do tipo GET, deverá garantir que URL chamado nunca é único, isto é, não é igual a qualquer outro chamado previamente. Isto é possível de implementar através da introdução de um parâmetro no URL cujo valor será sempre diferente das chamadas anteriores. Tipicamente recorre-se a um *timestamp* preenchido com o valor da data/hora actual. Simples mas robusto e consistente.

Exemplo numa aplicação Web

Integrada na aplicação Google Mail, está um componente que permite conversação em tempo real entre utilizadores deste serviço. Conforme é possível ver na Figura 37, este mecanismo é composto por um painel onde se encontram os utilizadores online com possibilidade de chat e do lado direito, da mesma figura, está um painel com uma conversa a decorrer. A troca de informação no chat e a actualização do estado dos utilizadores é realizada através de chamadas XMLHttpRequest.

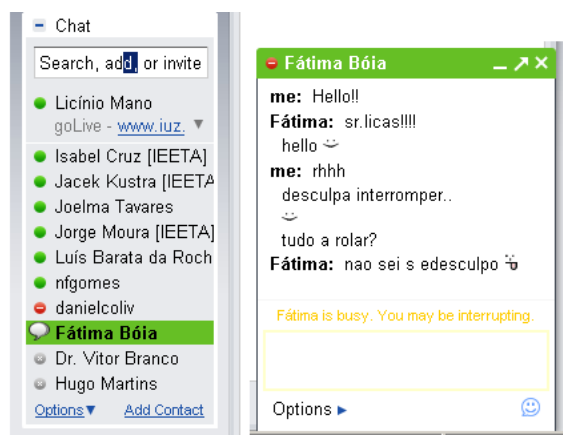


Figura 37: Gmail: conversação em tempo real

Alternativas

Os Padrões apresentados como alternativos ao XMLHttpRequest não serão abordados no âmbito deste trabalho. Ficam no entanto identificados para auxiliar em futuras pesquisas:

- Page Refresh, IFrame Call, HTTP Streaming, Richer Plugin, On-Demand JavaScript, Image-Cookie Call, Stylesheet Call, 204 Response, Import XML Document

Metáfora

Uma chamada XMLHttpRequest corresponde à uma conversa entre o Browser e o servidor enquanto, o Browser, mantém a conversa principal com o Utilizador.

5.4 Padrão – User Action

Até agora abordamos Padrões de comunicação com o servidor. Iremos agora abordar alguns Padrões que permitem despoletar essas chamadas.

Evidencia

3 – Amplamente utilizado

Palavras-chave

Ação, Clique, DHTML, DOM, Eventos, Rato, Teclado

Esboço

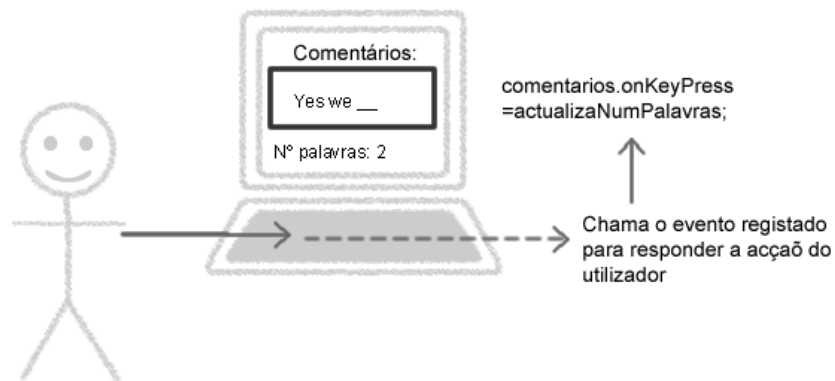


Figura 38: Padrão: User Action

Narrativa

Uma empresária está planejar as suas férias recorrendo para isso ao site de uma agência de viagens. Quando clica no campo destino, surge uma lista de cidades com interesse turístico, das quais escolhe Roma. A lista de cidades desapareceu dando lugar a uma lista de hotéis da cidade de Roma. Após escolher o hotel pretendido a lista desaparece também.

Problema

Como fazer com que a aplicação responda às acções do utilizador?

Forças

- Uma aplicação interactiva permite que o utilizador realize acções de diversas formas, tipicamente através do teclado e do rato.
- O feedback às acções do utilizador deve ser o mais imediato possível, de modo a manter a atenção do utilizador e ajuda-lo a perceber o modelo de interacção disponível na aplicação.
- A utilização de formulários como única forma de interacção com o utilizador limita a interactividade e diminui a experiência de utilização.

Solução

Captar as acções do utilizador recorrendo a JavaScript e aos seus mecanismos de gestão de eventos. A essência do Ajax é possibilitar uma maior interactividade e capacidade de interacção com o utilizador no contexto das aplicações Web. Para captar as acções do utilizador o JavaScript está “atento” aos eventos realizados sobre os elementos da estrutura DOM. A estrutura DOM consegue registar *handlers*, isto é, funções que são notificadas quando ocorre um evento.

Sempre que uma função é notificada sobre um clique de rato sobre um determinado elemento ou uma determinada tecla pressionada, é despoletada a acção previamente preparada.

Decisões

Que eventos estão disponíveis?

Existem já vários tipos de eventos disponíveis para JavaScript, mas a lista continua a aumentar à medida que vão surgindo novas versões dos Browsers. De seguida apresentamos os eventos mais disseminados e consistentes entre Browsers:

- *Tecla pressionada (onKeyPress, onKeydown, onKeyUp)*, é despoletado quando as teclas são pressionadas e posteriormente libertadas.
- *Focus do cursor (onBlur, onFocus)*:
 - *onBlur*, representa que o focus se perdeu, indicando tipicamente que um campo acabou de ser editado;
 - *onFocus*, representa o campo activo, tipicamente o campo onde o utilizador começou a introduzir informação;
- *Clique do rato (onmouseup, onmousedown, onclick, ondblclick)*, permite despoletar funções para os vários tipos de acções despoletadas pelo rato.
- *Movimento do Rato (onmouseover, onmouseout)*, permite despoletar um evento quando o ponteiro do rato entre ou sai da área ocupada por um elemento.

- *Seleção (onselect)*, despoleta um evento sempre que é realizada uma selecção.
- *Alteração de valor (onchange)*, despoleta um evento sempre que o valor de um campo do formulário é alterado.

Que atributos dos eventos vão ser comunicados?

O objecto do Evento contém vários pedaços de informação – atributos - os quais poderá aceder e utilizar para implementar determinados mecanismos de interacção. Tenha no entanto presente que alguns destes atributos podem estar presentes em Eventos nos quais não estava a contar.

Ficam os atributos mais frequentemente utilizados:

- *Elemento (target (Firefox), srcElement(IE))*, indicam o elemento sobre o qual aconteceu o evento.
- *Tipo de Evento (type)*, indica o tipo de acção que despoletou o evento (e.g. *onmouseover*, *onkeypress*).
- *Tecla (which (Firefox), keyCode(IE))*, indica o valor Unicode da tecla pressionada.
- *Teclas atalho (altKey, ctrlKey, shiftKey)*, ainda que não seja um atributo muito consistente entre Browsers, tipicamente serve para verificar se determinada tecla estava a ser pressionada no momento do evento.
- *Botão do rato (button)*, permite determinar qual o botão do rato que despoletou o evento (e.g. esquerdo, central, direito).
- *Localização do ponteiro do rato (clientX, clientY)*, indica o as coordenadas da localização do ponteiro do rato na janela do Browser.

Exemplo numa aplicação Web

O Google Maps permite a utilização de vários botões do rato, bem como cliques do rato e teclas pressionadas para controlar e manipular a visualização do mapa.

Alternativas

“Clic ‘n’ wait”

Padrão tradicional em aplicações Web, presente desde 1990, em que o utilizador sempre que activa um link tem de aguardar pelo envio do pedido e resposta do servidor com uma nova página. Da mesma forma, um formulário deve ser submetido, ficando o utilizador a aguardar pela validação do lado do servidor, seguido de uma resposta com uma nova página a confirmar que o processamento foi realizado com sucesso ou indicar eventuais erros.

Richer forms

Formulários mais interactivos do que os baseados HTML puro, mas menos do que os formulários implementados com Ajax. Isto é, algumas das validações são realizadas do lado do Browser evitando erros frustrantes para o utilizador (e.g. número de caracteres possíveis num campo, mascara de caracteres aceites num determinado campo).

Padrões Relacionados

- *Display Morphing, Page Rearrangement*, abordados em pontos anteriores, tipicamente são despoletados por *User Actions*
- *XMLHttpRequest Call, IFrame Call*, tipicamente despoletados por acções do utilizador (User Action)

5.5 Padrão – Scheduling

Evidencia

3 – Amplamente utilizado

Palavras-chave

Cron Job, Repetitivo, Periódico, Agendado, Sequência, Contagem decrescente

Esboço

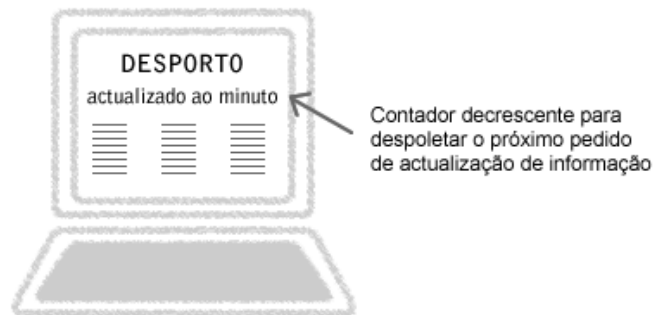


Figura 39: Padrão: Scheduling

Narrativa

Um segurança vigia as instalações através de uma aplicação online que lhe permite ver em tempo real as várias zonas das instalações. Para garantir que verifica todas as secções com a regularidade pretendida, a aplicação assinala a amarelo as secções que não foram investigadas nos últimos 5 e a vermelhos as que não foram investigadas nos últimos 10 minutos.

Problema

Como se podem agendar acções no futuro e num modo repetitivo?

Forças

- Por vezes uma aplicação tem de repetidamente executar uma tarefa (e.g. solicitar informação actualizada ao servidor, verificar o estado da aplicação).
- Por vezes uma aplicação precisa de agendar uma acção para acontecer passado um determinado período de tempo.

Solução

Utilizar contadores JavaScript para agendar acções. Através destes contadores JavaScript poderá agendar a realização pontual ou repetitiva de determinadas acções. Em ambos os casos é necessário definir a acção pretendida e o tempo até acontecer.

Exemplo numa aplicação Web

Tanto o Google Docs como o Google Mail apresentam um mecanismo de gravação automática que, de acordo com um determinado período de tempo, envia para o servidor a informação actualizada.

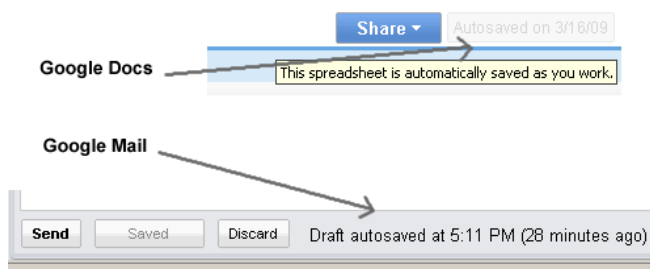


Figura 40: Google Docs e Mail: Gravação automática

Alternativas

HTTP Meta Refresh

A *meta tag* HTTP Meta Refresh agenda um período no final do qual o Browser deverá pedir novamente ao servidor uma versão da página actual – actualização completa da página. Tradicionalmente é utilizada por portais noticiosos para actualizar as notícias numa determinada página. No entanto é bastante limitada pois não permite ao utilizador qualquer controlo sobre a situação, bem como perde toda a informação de estado da aplicação do lado do Browser.

Metáfora

Os alarmes despertadores são programados para se activarem passado algum tempo. Alguns alarmes têm também a capacidade de repetirem esta acção por algum tempo até serem desligados.

5.6 Padrão – XML Message

Evidencia

3 – Amplamente utilizado

Palavras-chave

Documento, Formato, Hierarquia, Estrutura, Semântica, XML

Esboço

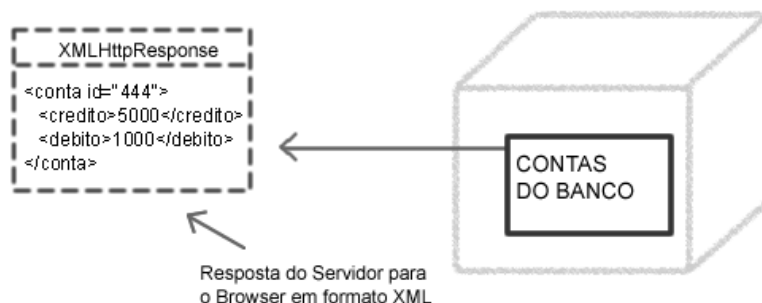


Figura 41: Padrão: XML Message

Narrativa

Um programador está a desenvolver o motor Ajax para uma aplicação web que está a desenvolver. A resposta do servidor aos seus pedidos XMLHttpRequest vem no formato XML. O motor que está a desenvolver deverá estar por isso preparado para processar blocos de informação em formato XML.

Problema

Como comunicar informação entre o servidor e o Browser?

Forças

- As aplicações Ajax requerem a comunicação nos dois sentidos Servidor-Browser e vice-versa.
- De ambos os lados Browser e Servidor deverão existir mecanismos de processamento das mensagens trocadas. Tipicamente as mensagens devem

ser trocadas num formato facilmente processável por JavaScript e pelas linguagens de servidor.

Solução

Enviar mensagens no formato XML entre o servidor e o Browser. É possível enviar mensagens XML em ambos os sentidos da comunicação. Do lado do servidor, é frequente a utilização deste tipo de mensagens para comunicar informação textual. Posteriormente do lado do Browser, as aplicações recebem através de pedidos XMLHttpRequest e fazem o processamento que entenderem. Por outro lado, quando o Browser precisa de enviar para o servidor informação complexa, também este pode elaborar uma estrutura XML e enviar através de um pedido XMLHttpRequest.

O XML é de facto a forma mais universal para comunicação de informação entre sistema e aplicações, uma vez que é independente da plataforma de origem e de destino. Por isso mesmo, grande parte das linguagens de programação apresentam as bibliotecas necessárias para elaborar e processar mensagens XML.

Decisões

Como será gerado o XML do lado do servidor?

Do lado do servidor esta tarefa pode ser realizada de várias formas:

- Através de código próprio que cria um texto XML.
- Através da criação de uma estrutura DOM.
- Utilizando Frameworks que auxiliam a converter estruturas de dados em XML.
- Utilizando ficheiros XML residentes no sistema de ficheiros do servidor.

Haverá algum DTD ou Schema para o XML a gerar?

Para garantir que uma determinada mensagem XML é compreensível no destino, a sua elaboração deve respeitar formato previamente combinado. O procedimento correcto e standard é a descrição num ficheiro separado da mensagem XML a comunicar (e.g. DTD – *Document Type Definition*)

Como irá o Browser transformar a mensagem XML em informação visual?

Nem sempre uma mensagem recebida tem de ser transformada em informação visual a ser apresentada pelo Browser. No entanto, em situações em que esse comportamento seja o pretendido, a solução passa por converter o XML em HTML e inserir directamente na estrutura DOM da página actual. A converção de XML em HTML pode ser conseguida através de: conversão manual através de JavaScript, XSLT ou através da utilização de *templates*.

Exemplo numa aplicação Web

Google Maps é talvez o mais famoso exemplo da utilização de mensagens XML. As mensagens são descarregadas em XML e através apresentadas no Browser com recurso a XSLT.



Figura 42: Google Maps

Alternativas

- **Mensagens de texto simples**, em situações em que a informação a comunicar é simplificada, uma mensagem de texto pode ser suficiente, evitando todo o processo de conversão do XML.
- **Mensagens JSON**, assim como uma mensagem XML, uma mensagem JSON consiste numa representação estrutura de dados. Uma vantagem para o XML é que uma mensagem JSON é imediatamente interpretada como um objecto JavaScript.

5.7 Padrão – Submission Throttling

Evidencia

3 – Amplamente utilizado

Palavras-chave

Buffer, Performance, Lista de Espera, Gestão de listas de espera

Esboço

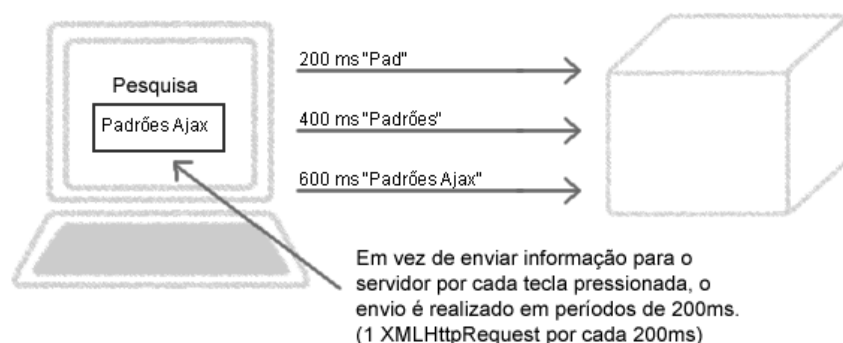


Figura 43: Padrão: Submission Throttling

Narrativa

Um programador que está a desenvolver um chat em Ajax pretende transmitir texto à medida que o utilizador vai escrevendo. No entanto, ele sabe que muitas vezes os utilizadores escrevem mais rápido do que a velocidade do sistema em copiar e enviar a informação. Então implementou um mecanismo de gestão de listas de espera (*throttling*) que garante o envio de apenas uma mensagem por cada 200 milissegundos.

Problema

Como podemos controlar o envio de informação para o servidor?

Forças

- O envio de informação pode acontecer de forma exagerada em aplicações de chat quando um utilizador se entusiasma por um assunto.
- O servidor vai tendo maior dificuldade em receber e processar mensagens à medida que o número de mensagens por segundo aumenta.

- Os Browsers por seu lado apresentam também limitações ao número de pedidos XMLHttpRequest que conseguem manter em aberto (aguardar resposta do servidor).
- Cada mensagem contém meta *tags*, para além da mensagem em si, que requerem processamento do lado servidor e do Browser.

Solução

Em vez de submeter um pedido por cada evento JavaScript (e.g. *keyPress*), guarde a informação num *buffer* do lado do Browser e faça o seu envio automático após um período de tempo definido. Numa grande parte das aplicações que implementam mecanismos de *Buffer* ou *Throttling*, o objectivo é assegurar dois aspectos fundamentais: performance (software) e recursos (hardware). Em alguns casos poderia até ser útil despoletar acção sempre que uma tecla é pressionada ou o rato é deslocado. No entanto, devido a limitação de largura de banda e latência das comunicações em rede, tal não é viável.

Decisões

Como deverá o servidor gerir a chegada dos pedidos?

Os mecanismos de *throttling* apresentam alguns riscos de consistência. Enquanto um pedido vai e regressa do servidor o utilizador pode já ter executado outras acções. Nesse sentido é importante precaver como deverá o sistema reagir quando uma resposta recebida já não faz sentido no contexto de utilização actual.

Como criar um *buffer* de pedidos?

A solução, ainda que simplista, mais recorrente é a utilização de cronómetros (*timers*) que são reiniciados quando determinada acção é executada. Após serem iniciados os *timers* continuam activos até receberem instruções para parar o reiniciar.

Quantos buffers são necessários por aplicação?

Não há nenhuma restrição em particular para a existência de apenas um *timer* na aplicação. É por isso possível activar vários *timers*. Importa no entanto ter em atenção, que a ordem de chegada de respostas a pedidos despoletados por *timers*, pode não corresponder exactamente à ordem pela qual os pedidos foram comunicados.

Exemplo numa aplicação Web

Funcionalidade disponível no Google Suggest e no Google Mail. No GMail, para escolher um destinatário, ao escrever a primeira letra do respectivo nome, é apresentada uma lista com os emails de destinatários com nomes correspondentes.

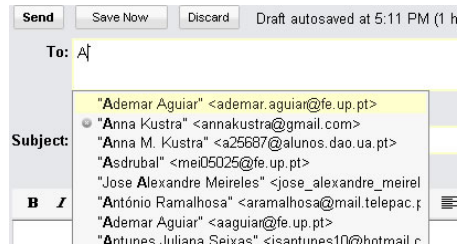


Figura 44: Google Mail: Escolha de destinatários

Padrões Relacionados

- **Periodic Refresh**, é a acção contrária à *Submission Throttling*. Em vez de enviar periodicamente informação para o servidor, é solicitado periodicamente o envio de uma nova versão da página actual.

Metáfora

Imagine que está a fazer um ditado para uma criança que está a aprender escrever. Deverá reservar algum tempo entre cada palavra ou conjunto de palavras para que a criança tenha tempo de interpretar e escrever as palavras que acabou de ditar.

5.8 Padrão – Progress Indicator

Evidencia

3 – Amplamente utilizado

Palavras-chave

Actividade, *Feedback*, Progresso, Aguardar

Esboço

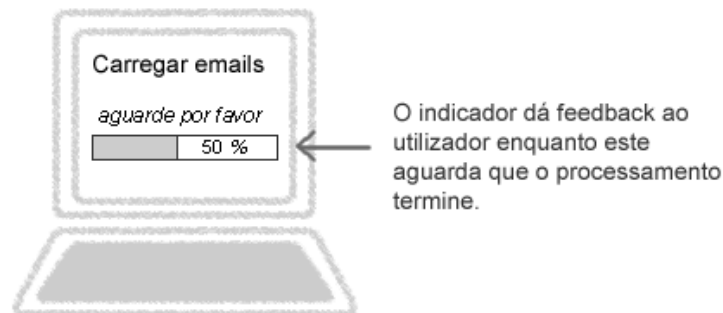


Figura 45: Padrão: Progress Indicator

Narrativa

O utilizador está a aceder à sua caixa de correio online. Enquanto aguarda que o email seja descarregado, a aplicação apresenta feedback sobre a quantidade de mensagens ou tempo estimado até conclusão da tarefa.

Problema

Como apresentar feedback ao utilizador enquanto este aguarda pela resposta do servidor?

Forças

- As aplicações Ajax utilizam tipicamente chamadas XMLHttpRequest para comunicar com o servidor. Como este processo de comunicação acontece de forma invisível para o utilizador, em determinadas situações pode ser importante apresentar feedback sobre o estado da comunicação.
- Os elementos que estão a ser actualizados devem estar devidamente assinalados.

- Os utilizadores gostam de estar no controlo da aplicação. Enquanto estão a aguardar essa sensação perde-se. Nesse sentido é importante minimizar o atraso e tempo de espera introduzido pelas comunicações com o servidor.

Solução

Indicar o progresso da comunicação com o servidor. Nem sempre é possível reduzir o atraso introduzido por uma comunicação com o servidor. No entanto, a presença de um indicador de progresso da comunicação (*Progress Indicator*), ajuda a manter a atenção do utilizador, uma vez que lhe permite tomar conhecimento do modo de funcionamento do sistema. Ao mesmo tempo o indicador de progresso possibilita a percepção ao utilizador sobre se a aplicação bloqueou ou está simplesmente a processar.

A utilização deste tipo de mecanismos é possível recorrendo à verificação do estado de uma chamada XMLHttpRequest.

Decisões

Que tipo de indicador de progresso utilizar?

Para informações específicas sobre este assunto, consultar as boas práticas sugeridas por Jakob Nielsen, no seu trabalho "*Usability Engineering*"⁶².

Que tipo de feedback utilizar para tempos de espera maiores?

Para longos períodos de espera é necessário dar a indicação ao utilizador sobre a percentagem de processamento que já foi realizada e o que ainda falta fazer.

Exemplo numa aplicação Web

Após indicarmos os dados de utilizador no ecrã de autenticação no Google Mail, é apresentado um indicador de progresso sobre o processo de carregamento das mensagens da respectiva conta de email.

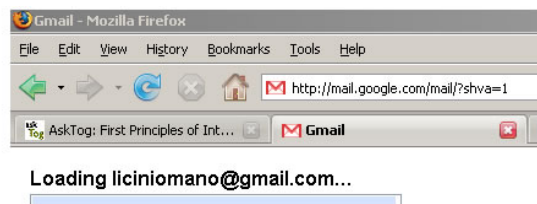


Figura 46: Google Mail – Carregamento de dados

⁶² Jakob Nielsen, Response Times: The Three Important Limits, <http://www.useit.com/papers/responsetime.html>, Acesso: 2009-03-18

Padrões Relacionados

- **Status Area** ou **Popup**, tipicamente um indicador de progresso surge numa área específica sobre o estado da aplicação ou num *Popup* que se sobrepõem a toda a aplicação.
- **Guesstimate**, por vezes não é possível determinar o tempo que a tarefa irá demorar, nem mesmo apresentar a percentagem de trabalho realizado. Nestes casos apresentar uma estimativa desse tempo é melhor do que não apresentar qualquer informação ao utilizador.
- **Distributed Events**, quando recebe a resposta de uma chamada, é necessário dar a indicação para retirar o indicador de Progresso. Nesta situação pode surgir alguma confusão, principalmente se a instrução de retirar o indicador de progresso estiver incluída no processamento da informação recebida.

Metáfora

Um serviço de atendimento público costuma normalmente apresentar um sistema de senhas para gerir a fila de pessoas a atender. Tipicamente existe um ecrã a indicar a senha actual (pessoa a ser atendida).

Saber Mais

No seu trabalho “First Principles of Interaction Design⁶³”, Bruce Tognazzini apresenta algumas recomendações sobre como gerir os momentos em que o utilizador tem de aguardar pelo sistema.

⁶³ Bruce Tognazzini, First Principles of Interaction Design,
<http://www.asktog.com/basics/firstPrinciples.html#latencyReduction>, Acesso: 2009-03-18

5.9 Padrão – Ritch Text Editor

Evidencia

3 – Amplamente utilizado

Palavras-chave

Editor, Formatação, Tipos de letra, estilos de texto, Área de texto, Barra de ferramentas, Escrever, WYSIWYG

Esboço

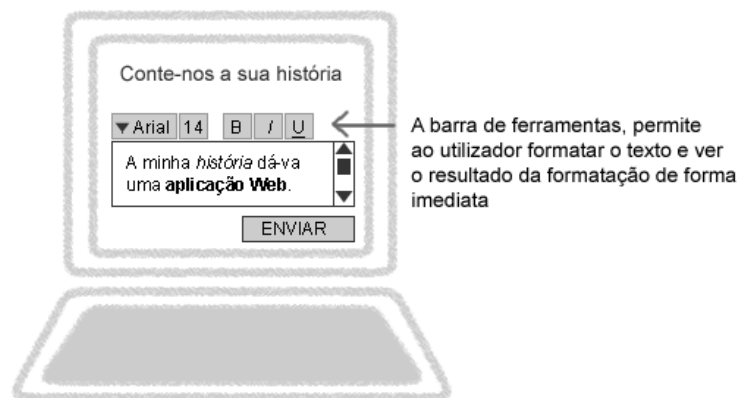


Figura 47: Padrão: Rich Text editor

Narrativa

Um gestor de conteúdos está a introduzir uma notícia no portal da sua empresa. Para a tarefa em questão ele poderá escrever o texto directamente na aplicação Web, aplicando a barra de ferramenta disponíveis para adicionar estilos e formatações ao texto que está a introduzir. Começa pelo título onde define o tipo e tamanho de letra e ao longo do corpo da notícia vai adicionando negritos e itálicos nas expressões mais relevantes.

Problema

Como podem os utilizadores criar e actualizar texto com formatação?

Forças

- Algumas aplicações permitem aos utilizadores introduzir e editar blocos extensos de informação textual.

- A formatação de conteúdos deve tirar o máximo partido de HTML e CSS.
- A grande maioria dos utilizadores não tem conhecimentos em HTML.

Solução

Incorporar um Editor nas áreas de texto da aplicação, que permita a formatação dos conteúdos e visualização imediata do resultado das formatações aplicadas. Tipicamente, este componente nas aplicações Web assemelha-se a um editor de texto simplificado, colocado na parte superior das áreas de texto que possibilitam edição e formatação. A área de edição de texto pode ser uma *div* ou uma *textarea* tradicional.

Exemplo numa aplicação Web

Na aplicação Google Docs pode trabalhar diferentes tipos de documentos (e.g. documentos de texto, folhas de cálculo e apresentações). De acordo com o tipo de documento em que estamos a trabalhar é apresentado um editor que ajuda o utilizador a formatar e manipular os conteúdos que à medida que os vai introduzindo.

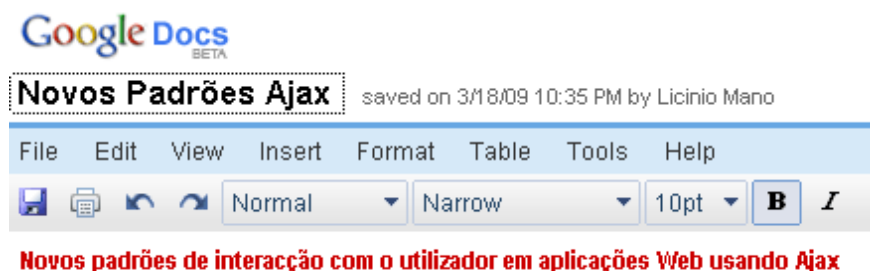


Figura 48: Google Docs – Editor de texto

Padrões Relacionados

- **Progress Indicator**, ajuda a manter a atenção do utilizador na altura de gravação da informação do lado do servidor. Por outro lado o carregamento destes componentes pode demorar um bocado, de forma que informa o utilizador que há informação que está a ser carregada.
- **Status Area**, preenchida com informação que ajuda o utilizador a compreender o que está acontecer.

Metáfora

Aplicação Desktop de edição e processamento de texto.

5.10 Padrão – Drag and Drop

Evidencia

3 – Amplamente utilizado

Palavras-chave

Arrastar, Largar, Mover, colocar numa nova posição

Esboço

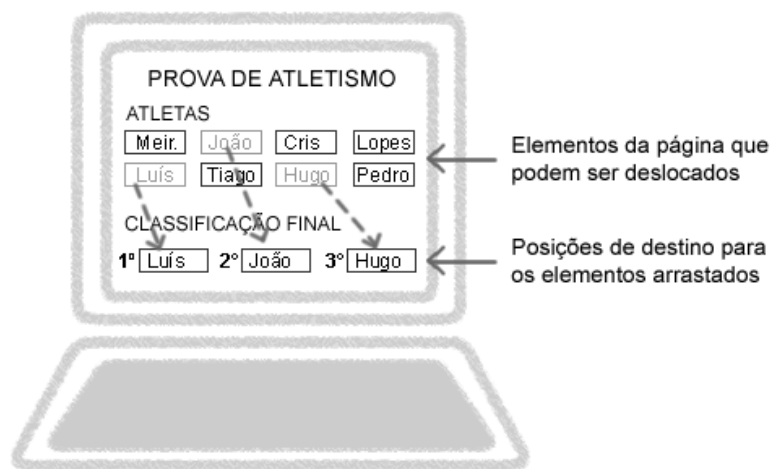


Figura 49: Padrão: Drag and Drop

Narrativa

Um leitor está a comprar livros numa livraria online. Sempre que pretende comprar um livro tem de o seleccionar (1 clique do rato) e arrastar para o respectivo carrinho de compras.

Problema

Como podem os utilizadores mudar a posição de um elemento na página?

Forças

- As relações entre elementos são importantes e por vezes a tarefa do utilizador consiste em alterar as relações entre esses elementos.

- As aplicações Ajax permitem a representação visual dessas relações, bem como, propiciam feedback para o utilizador quando este as está a alterar.

Solução

Possibilitar aos utilizadores um mecanismo para seleccionar, arrastar e largar (*drag and drop*) elementos de uma página. O mecanismo de interacção *drag and drop* já demonstrou o seu potencial nas aplicações Desktop. Até agora a sua utilização em aplicações Web era impraticável, situação agora alterada com o aparecimento do Ajax.

Do ponto de vista do utilizador, trata-se de um mecanismo bastantes simples. Selecciona, com um clique do rato, o elemento que quer manipular. Mantendo o botão do rato pressionado, desloca o elemento pela área disponível. Ao chegar à área de destino pretendido, liberta o botão do rato e é despoletada uma acção. Seguem-se algumas utilizações possíveis para este Padrão Ajax:

- *Organização de listas*, ordenar de elementos, adicionar e remover elementos.
- *Reposicionar elementos*, alterar o posicionamento de determinados elementos na página.
- *Criar conjuntos de elementos*, por exemplo arrastar produtos de uma loja online para o respectivo carrinho de compras.
- *Acções*, despoletar uma acção apenas por arrastar um determinado elemento para áreas específicas do ecrã poder (e.g. arrastar um elemento para a reciclagem, apaga o elemento da página).

Decisões

Quais os constrangimentos e restrições a implementar no mecanismo?

É fundamental dar a entender ao utilizador que é possível arrastar um determinado elemento. Invariavelmente, este tipo de acção apenas pode ser realizado dentro de uma área predefinida, logo, também esta área deve estar bem definida.

Os utilizadores não estão habituados a este tipo de interacção em aplicações Web?

Ainda que esteja amplamente disseminada em aplicações Desktop, a utilização deste mecanismo de interacção em aplicações Web sofre do síndrome: “Não sabia que era possível fazer isso.”. Efectivamente este mecanismo apenas recentemente chegou às aplicações Web e por isso mesmo os utilizadores não estão à espera de poderem utilizar as aplicações dessa forma. Como referido, numa primeira fase é importante demonstrar ao utilizador o tipo de interacção que este pode realizar sobre determinados elementos numa página.

Exemplo numa aplicação Web

No iGoogle é possível reposicionar os vários painéis disponíveis na página. Para tal basta seleccionar um painel e arrastar para uma nova posição. Quando arrastamos um painel, as posições que ele pode assumir vão sendo assinaladas com linhas a tracejado.

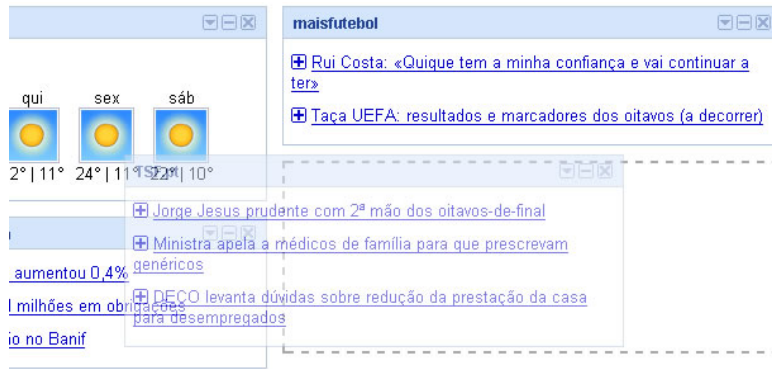


Figura 50: iGoogle – Reorganização de painéis

Padrões Relacionados

- **Slider**, é um tipo especial de *drag and drop*, onde o indicador do valor é arrastado ao longo de um eixo de valores.

Metáfora

Imagine-se num jogo de xadrez e é a sua vez de jogar. Terá por isso de apanhar uma peça, desloca-la e largar a mesma numa nova posição.

5.11 Padrão – Malleable Content

Evidencia

3 – Amplamente utilizado

Palavras-chave

Affordance, Colaboração, Edição, Introdução de dados, Selecção

Esboço

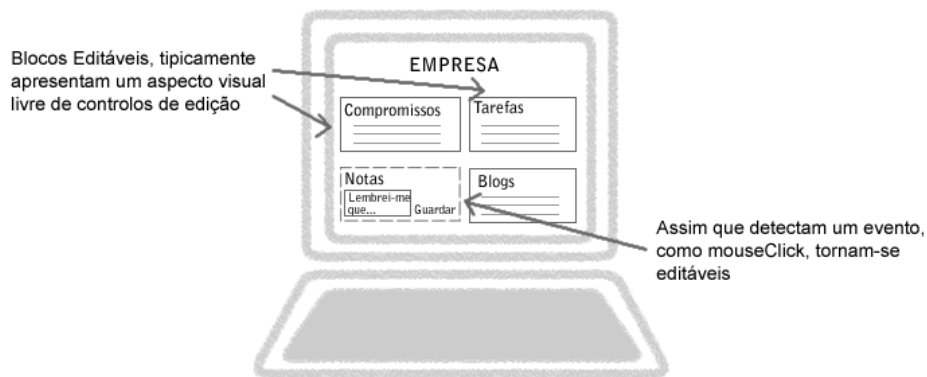


Figura 51: Padrão: Malleable Content

Narrativa

Um editor está a validar a próxima edição da página inicial do portal da revista. A informação é-lhe apresentada como se um leitor comum se tratasse. À medida que pretende realizar alterações basta carregar no conteúdo que este transforma-se num formulário que permite a edição do conteúdo.

Problema

Como permitir aos utilizadores editar o conteúdo?

Forças

- Cada vez mais para além de consultar informação os utilizadores introduzem informação. Administradores e utilizadores regulares investem cada vez mais tempo na escrita e edição de conteúdos directamente online.
- Independente do perfil de utilizador, é importante indicar qual a informação que este pode alterar.

- Para facilitar a colaboração de mais utilizadores e melhorar a experiência de utilização na introdução e edição de conteúdos, devem ser eliminados constrangimentos e restrições.
- Tipicamente, as páginas de gestão de conteúdos são menos apelativas visualmente porque estão carregadas com controlos dos formulários. Um utilizador em tarefa de edição tem de imaginar como o mesmo vai ficar quando disponível para consulta, isto é, não há uma relação directa entre o aspecto de edição e o aspecto final para consulta pelos utilizadores.

Solução

Desenvolver páginas com blocos de informação que podem ser alterados e directamente editados na página actual. Para que este Padrão seja exequível a informação deve estar agrupada em pequenos contextos de informação. Cada um destes contextos permite a edição dos seus conteúdos directamente, bastando apenas um clique para activar o modo de edição.

Existem alguns problemas de consistência na utilização deste Padrão em páginas que o utilizador actualiza constantemente através das funcionalidades *refresh* ou *reload*. Daí que, invariavelmente sejam utilizadas páginas específicas para edição de conteúdos com vários campos de informação. O problema desta solução está na dificuldade de percepção dos utilizadores sobre qual será o aspecto, no modo de consulta, da informação que acabou de introduzir.

Decisões

Como podem os utilizadores localizar os blocos de informação editável?

Uma página pode conter vários blocos de informação. Alguns deles podem ser editados pelos utilizadores e outros não. É fundamental apresentar informação visual sobre quais os elementos que podem ou não ser alterados directamente na página.

Qual a dimensão dos blocos de informação editáveis?

Qualquer zona de informação editável é passível de conter o Padrão Malleable Content. No entanto, importa ter em atenção se o bloco de informação que está a ser editado tem impacto ou está dependente de outros blocos de informação. Caso existam restrições a este nível, a solução mais adequada passará pela possibilidade de o utilizador poder editar toda a estrutura dos conteúdos de uma só vez.

Como deverá o utilizador proceder para gravar as alterações que realizou?

Após ter iniciado a edição de conteúdo, o utilizador vai ter de indicar que conclui a edição e que o conteúdo deverá voltar ao modo de visualização. Para tal, é possível utilizar duas estratégias:

- Controlos com a indicação de acção (e.g. guardar)

- Após detectar um evento de `onBlur`, a aplicação sabe que o utilizador terminou a introdução de informação, num determinado campo, voltando a colocar o conteúdo no estado de visualização.

Exemplo numa aplicação Web

Na aplicação iGoogle pode editar directamente a informação presente em cada painel. Depois de alterar, terá apenas de seleccionar um das acções possíveis (e.g. Guardar, Cancelar)



Figura 52: iGoogle – Malleable Content

Alternativas

- **Compound Edit**, possibilidade de editar todos os conteúdos presentes numa página, através de uma página específica de edição de conteúdo. Ainda que seja uma solução menos imediata que o Malleable Content, pode ser uma solução mais eficaz para longos períodos de edição de conteúdos.

Padrões Relacionados

- **Rich Text Editor**, um Malleable Content, poderá apresentar um editor de texto para auxiliary na formatação de conteúdos.
- **Microlink**, a possibilidade de edição do conteúdo por surgir através de um Microlink.
- **One-Second visual effects**, os efeitos visuais de curta duração, representam um mecanismo simples para auxiliar na edição de conteúdo (e.g. fadeIn, fadeOut, piscar, ou tremer).
- **Highlight**, os conteúdos editáveis (Malleable Content), podem ser destacados visualmente, de forma a facilitar a sua identificação por parte dos utilizadores.

Metáfora

Aplicações com Blocos Editáveis são como um quadro de informação que podemos consultar, mas facilmente podemos alterar.

5.12 Padrão – MicroLink

Evidencia

3 – Amplamente utilizado

Palavras-chave

Surgir, deslizar, Emergir, Link, Introduzir, Esconder

Esboço

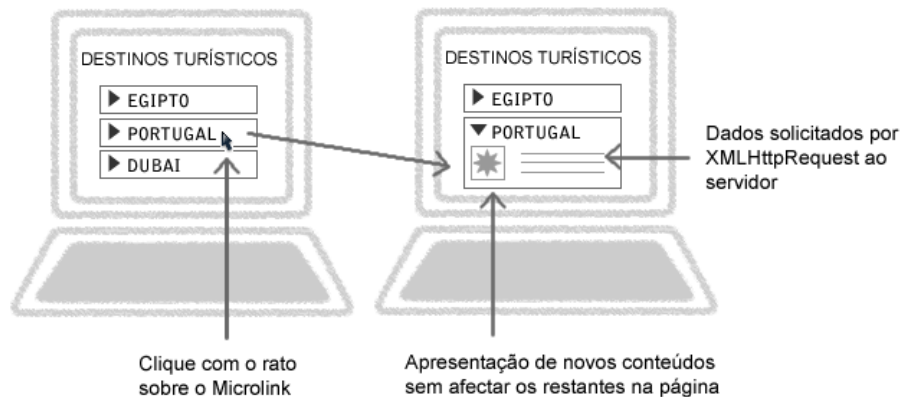


Figura 53: Padrão: MicroLink

Narrativa

Um gestor de projecto entra na plataforma de trabalho colaborativo da empresa para verificar as tarefas realizadas pelos seus colaboradores. Após seleccionar o projecto, é-lhe apresentada uma lista com o nome e fotografia de cada colaborador. Quando selecciona um colaborador a respectiva área cresce o suficiente para apresentar um resumo das actividades do programador e um conjunto de links para outras informações.

Problema

Como pode o utilizador navegar rapidamente em conteúdos extensos?

Forças

- As aplicações Ajax respondem as acções do utilizador disponibilizando novos conteúdos.

- Para melhorar a experiência de utilização e eficiência da mesma, a apresentação de novos conteúdos deve acontecer de forma fluida, rápida e sem elementos de distração.
- Actualizações completas da página diminuem a eficácia da interacção e quebram a concentração do utilizador, bem como, não permitem acompanhar as modificações que acontecem na página.

Solução

Disponibilizar Microlinks que abrem novas áreas de informação na página, sem necessitar da actualização completa da página actual. Um Microlink é uma sofisticação, introduzida pelo Ajax, nas tradicionais hiperligações. Na grande maioria das vezes consiste numa chamada XMLHttpRequest, em que é passado um identificador e a resposta do servidor com os detalhes desse identificador são introduzidos directamente na estrutura DOM da página.

Tipicamente o conteúdo em questão deverá ser de dimensão reduzida. Caso se trate de um conteúdo de maiores dimensões a solução mais indicada será uma nova página para disponibilizar o conteúdo.

Decisões

O que acontece ao restante conteúdo?

Tipicamente um Microlink adiciona conteúdo numa página. No entanto a introdução de novo conteúdo implica a disponibilização de espaço visual para esse conteúdo. É por isso importante perceber como vão reagir os elementos influenciados para inclusão de novos conteúdos na página.

Por outro lado, é importante remover um Microlink quando o utilizador activa um novo Microlink. Desta forma, consegue controlar a complexidade e quantidade de elementos na página.

Como são apresentados visualmente os Microlinks?

A questão coloca-se pela confusão que pode causar a diferença de comportamentos entre um link simples e um Microlink. Para evitar que o utilizador se sinta confuso ou na dúvida sobre qual o resultado de uma acção, importa assinalar de forma diferente, elementos que proporcionam diferentes mecanismos de interacção.

Exemplo numa aplicação Web

Numa aplicação como o Google Mail, ao carregar no topo de uma mensagem é apresentado o conteúdo da mesma. A utilização do Microlink é especialmente notada, quando um email existem vários emails de resposta. Para ver o conteúdo de cada uma dessas mensagens basta carregar no respectivo título. Apesar de ver o conteúdo

da mensagem continua a manter a noção de ordem na sequência de mensagens pois consegue ver o título das mensagens anteriores e seguintes.

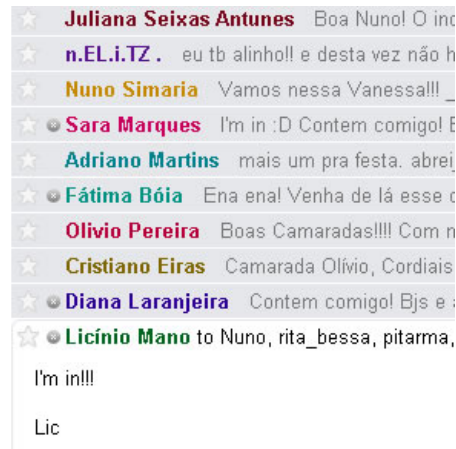


Figura 54: Google Mail – Lista de mensagens encadeadas

Padrões Relacionados

- **Malleable Content**, juntamente com o Microlink os dois Padrões modelo um estilo de interacção muito usual. Tipicamente, os Malleable Content podem ser despoletados por Microlinks.
- **One-Second visual effects**, ajudam a passar a informação ao utilizador sobre uma alteração em curso.

Metáfora

Um Microlink é como uma gaveta com a etiqueta brinquedos. Temos de abrir a gaveta para saber quais os brinquedos que lá se encontram.

5.13 Padrão – Highlight

O Padrão Highlight apresenta várias soluções alternativas e relacionadas. Nas respectivas secções serão abordados, com especial destaque, os Padrões que se enquadram no conceito de animações de curta duração (*One-Second Visual Effects*).

Evidencia

3 – Amplamente utilizado

Palavras-chave

Actualização automática, Sincronismo, Sincronizar, Tempo-real

Esboço

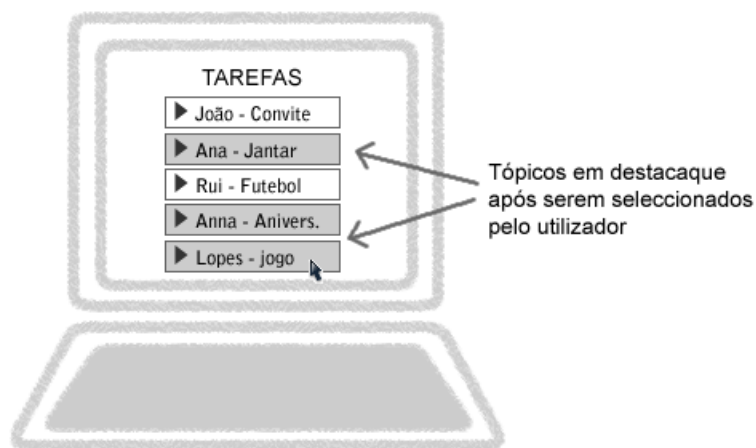


Figura 55: Padrão Highlight

Narrativa

Um editor de *blog* verifica que existam os comentários menos próprios num *post* recentemente publicado. Após entrar na área de edição de conteúdos, vai clicando nos referidos *post* que ficam assinalados com cor de fundo amarela. No final carrega na opção bloquear para que todos os *post* assinalados deixem de estar visíveis para o público.

Problema

Como assinalar/destacar determinados elementos numa página Web?

Forças

- Uma aplicação Ajax pode apresentar um conjunto extenso de informação de uma só vez. Por seu lado, cada conjunto de informação pode ter um estado próprio que importa demonstrar ao utilizador.
- Tipicamente, o utilizador encontra-se a trabalhar sobre um bloco de informação que é necessário destacar.
- Adicionalmente, o utilizador poderá pretender seleccionar um conjunto de blocos de informação para depois aplicar uma determinada acção. Os elementos seleccionados devem estar destacados dos restantes elementos.

Solução

Destacar elementos através da alteração do seu aspecto visual, de forma consistente e suficiente para chamar a atenção do utilizador. Este Padrão já era utilizado antes da introdução do Ajax, de qualquer forma a sua presença neste conjunto de Padrões deve-se à importância que este desempenha em mecanismos de interacção mais sofisticados. Seguem-se algumas propostas de utilização do Padrão Highlight:

- Assinalar o elemento em que está o *focus* do cursor.
- Indicar quais os elementos seleccionados
- Assinalar as acções disponíveis, quando o utilizador coloca o ponteiro do rato sobre um determinado elemento.
- Indicar a importância, ou falta dele, de determinados elementos na página.
- Indicar que um elemento está em actualização

Decisões

Quais as alterações visuais num elemento destacado?

O Highlight deve ser um elemento de destaque (e.g. alteração de cor, tamanho ou margens do elemento), mas não deve funcionar como perturbador da atenção do utilizador.

Como deve ser adicionada e removida a informação de destaque?

O primeiro aspecto a salientar sobre o Highlight é que existem apenas dois estados, está ou não assinalado. No entanto é possível suavizar a transição entre estados com animações suaves (e.g. *fadeIn* ou *fadeOut*)

Exemplo numa aplicação Web

Conforme é possível verificar na Figura 56, o Google mail permite aos seus utilizadores seleccionarem um conjunto de mensagens para posteriormente aplicarem uma acção. As mensagens em estado normal apresentam uma estrela sem preenchimento, enquanto as mensagens assinaladas apresenta uma estrela preenchida.



Figura 56: Google Mail – Mensagens assinaladas

Padrões Relacionados

- **Status Area**, é a área indicada para explicar o motivo de um elemento estar destacado.
- **One-Second visual effects**, importantes para suavizar a transição entre estados.

Metáfora

Utilização de um marcador fluorescente para assinalar um artigo de jornal com particular interesse.

5.14 Síntese e Conclusões

Neste capítulo foram descritos detalhadamente treze Padrões Ajax com impacto directo na experiência de utilização de aplicações Web. Esta descrição permitiu tomar conhecimento sobre a solução para problemas comuns de quem desenvolve ou quer desenvolver aplicações Ajax, bem como, elencar os compromissos e questões que devem ter em conta na adopção desta tecnologia.

Constitui por isso num bom manual, para quem pretende tomar conhecimento de quais os Padrões Ajax mais relevantes para o desenvolvimento de aplicações cuja sofisticação da experiência de utilização se aproxime ou, pelo menos, seja convergente com as aplicações desktop.

6 Aplicações baseadas na Internet

Em seguida iremos sistematizar alguns dos exemplos de aplicações reais que foram sendo indicados ao longo do trabalho. Iremos apresentar uma descrição genérica de cada aplicação e para a mesma identificar se tem presente os padrões considerados neste trabalho como fundamentais para a evolução da experiência de utilização de aplicações Web em consequência da cada vez mais evidente convergência com as aplicações Desktop.

De notar que, 4 das 5 aplicações apresentadas pertencem ao universo da Google. Não foi intencional o destaque desta empresa, nem queremos com isso passar a mensagem de que a Google domina o universo das aplicações Web Ajax.

No entanto, temos de assumir que os melhores exemplos encontrados para aplicações com uma utilização expressiva (número de utilizadores e frequência de utilização relevante), são efectivamente algumas das aplicações lançadas pela Google.

Admitimos que devido à extensão do universo Web e da rápida disseminação do Ajax, o número de aplicações com base nesta tecnologia continue a aumentar e o número de bons exemplos aumente também. Admitimos também que na actualidade possam já existir outros e melhores exemplos da utilização real da tecnologia Ajax.

No âmbito deste trabalho, estas foram as aplicações mais vezes referidas nas pesquisas realizadas pelo autor, contributos de diversos autores nas referências bibliográficas e Web, bem como, opiniões de utilizadores Web que depressa identificaram as inovações indicadas nas aplicações que mais costumam utilizar.

6.1 Aplicações Web tradicionais

Antes de avançar directamente para as aplicações Web Ajax, iremos referir alguns exemplos de aplicações Web tradicionais.

Um exemplo claro de aplicações Web tradicionais são as soluções de *home banking*. Talvez por questões de segurança ou complexidade, as soluções de home banking teimam em não adoptar paradigmas de interacção mais sofisticados, ficando intimamente posicionadas ao Padrão tradicional, *click and wait*. Por outro lado, temos clientes de email como SquirrelMail, que ainda não deram o passo para o Ajax, sendo que a concorrência (Google Mail) adoptou esta tecnologia com enorme sucesso.

De facto os exemplos de aplicações em componentes Ajax começam a escassear. Se por um por um lado esta conclusão pode indiciar uma pesquisa pouco profunda, o facto é que a grande maioria de Aplicações Web tem sabido gerir e incorporar a

inovação introduzida pelo Ajax. E ainda que possam não representar exemplos completos de aplicações Ajax(*Ajax Deluxe*), na sua maioria apresentam componentes (*Widgets*) que melhoram a experiência de utilização (e.g. LinkedIn, Hi5, Meebo).

6.2 Aplicações Web baseadas em Ajax

6.2.1 Google Maps

O Google Maps é uma aplicação de pesquisa e consulta de mapas e imagens de satélite do Planeta Terra, disponível gratuitamente na Web pela Google. Para além da consulta de mapas, permite a definição de rotas entre diferentes pontos.

A sua popularidade deve-se não só ao facto de ter sido a primeira aplicação do género disponível na Web, mas também pela flexível e extensa API que disponibiliza para toda a comunidade de programadores que deseje colocar um mapa interactivo na sua aplicação.

No que toca a Padrões Ajax, os mais prementes nesta aplicação são:

Drag and Drop, Display Morphing, XMLHttpRequest Call e XML Message

6.2.2 Kiko Calendar

Calendário online para gestão de agenda de compromissos. Para além da sua interface altamente interactiva, o Kiko Calendar apresenta um conjunto de serviços online como alertas, gestão de contactos e leitor de RSS. Apresenta também uma API para integrar em aplicações desenvolvidas por terceiros.

Os Padrões Ajax em destaque são:

Page Rearrangement, XMLHttpRequest Call, User action, Drag and Drop, Highlight

6.2.3 Google Docs

O Google Docs é um editor de: texto, folhas de cálculo e apresentações; baseado na Web de utilização gratuita. Disponibilizado pela Google, o Google Docs permite a criação e edição online de documentos, bem como, a realização de trabalho colaborativo, em tempo real, sobre esses documentos. O Google Docs resulta da aglutinação de três aplicações distintas: *Writely*, *Spreadsheets* e *Presentations*.

Os principais Padrões Ajax presentes são:

Display Morphing, Page Rearrangement, Scheduling, Rich Text Editor, Drag and Drop

6.2.4 iGoogle

O iGoogle ou, de acordo com a anterior designação, Página Pessoal no Google, é um serviço apresentado pela Google que consiste numa versão personalizável, graças a

componentes Ajax, da página inicial da Google. Deste modo, para além do tradicional de pesquisa, o utilizador pode definir diferentes painéis onde lhe serão apresentadas notícias e informações de acordo com as suas preferências.

Principais Padrões Ajax detectados:

Malleable Content, Microlink, Submission Throttling, Page Rearrangement

6.2.5 Google Mail

Cliente de email online de utilização gratuita, disponibilizado pela Google. A sua integração de serviços (e.g. conversação em tempo real, formatação avançada de mensagens), bem como, a sua interface com o utilizador revolucionária, potenciaram a popularidade desta ferramenta de gestão de emails.

Principais Padrões Ajax detectados:

XMLHttpRequest Call, XML Message, Rich Text Editor, Microlink, Submission Throttling

6.3 Mapeamento de Padrões Ajax em aplicações Web

O Quadro 8 sistematiza a utilização dos Padrões apontados como fundamentais para o melhoramento da interactividade e interacção em aplicações Web usando Ajax, nas aplicações de referência abordadas neste trabalho.

Através do referido quadro podemos verificar que nas 5 aplicações seleccionadas é possível identificar a utilização dos Padrões indicados neste trabalho como os mais representativos da evolução introduzida pelo Ajax. Por si só esta conclusão pode ser dúbia no sentido em que o universo de aplicações seleccionadas como demonstrador pode ter sido intencionalmente manipulado para que este fosse o resultado.

No entanto, mais importante do que questionar o universo seleccionar o universo de aplicações indicado é o facto da importância e preponderância que estas aplicações têm vindo a assumir no dia-a-dia dos utilizadores da Web e tentar encontrar uma ligação com os novos Padrões de interacção que estas apresentam.

Este trabalho defende por isso, que o sucesso e notoriedade de cada uma das aplicações indicadas está intimamente relacionado com a evolução e inovação da experiência de utilização que proporcionam aos seus utilizadores. Experiência de utilização possível apenas, em aplicações Web, após o surgimento e maturação do Ajax e respectivos Padrões de utilização.

Quadro 8: Mapeamento de Padrões Ajax em aplicações actuais

	Display Morphing	Page Rearrangement	XMLHttpRequest Call	User action	Scheduling	XML Message	Submission Throttling	Progress Indicator	Rich Text Editor	Drag and Drop	Malleable Content	Microlink	Highlight
Google Maps	●	●	●	●		●		●		●		●	●
Kiko Calendar	●	●	●	●		●		●		●	●	●	●
Google Docs	●	●	●	●	●	●		●	●	●	●	●	●
iGoogle	●	●	●	●		●	●	●		●	●	●	●
Google Mail	●	●	●	●	●	●	●	●	●	●	●	●	●

6.4 Síntese e Conclusões

Este capítulo consistiu na sistematização das aplicações Web que foram utilizadas recorrentemente ao longo do trabalho para demonstrar a forma como a utilização do Ajax potenciou a experiência de utilização em aplicações Web. É no entanto realizada uma breve referência sobre aplicações Web tradicionais e o caminho que estas estão a seguir rumo ao Ajax ou pelo menos, na adopção de componentes Ajax que melhorem significativamente os paradigmas de interacção que disponibilizam aos utilizadores. Por fim é apresentado um mapeamento entre os Padrões Ajax, identificados como de maior impacto na experiência de utilização em aplicações Web, e as aplicações Ajax com maior notoriedade na actualidade.

Este mapeamento permitiu concluir que os Padrões identificados se encontram amplamente disseminados e que, provavelmente, estão na base do sucesso e popularidade das referidas aplicações. As tradicionais aplicações Web tenderão a convergir e adoptar os respectivos Padrões, ainda que o possam fazer de forma mais global (*Ajax Deluxe*) ou localizada (*Ajax Lite*).

7 Conclusões

Na base do trabalho realizado encontrava-se dúvidas e perguntas sobre um fenómeno de evolução em curso. Relembro a essência dessas perguntas:

- O que está a mudar na Web e qual o papel do Ajax nesse processo evolutivo?

Ao longo deste trabalho abordamos um conjunto alargado de assuntos que acreditamos contribuir de forma directa para a necessidade e sucesso de uma tecnologia ou, mais precisamente, conjunto de tecnologias e arquitectura de utilização, sistematizadas no **Padrão Aplicação Ajax**.

Convergência entre Web e Desktop

As grandes conclusões, ou pelo menos, argumentos defendidos neste trabalho sugerem que a Web, mais concretamente, o conceito de aplicação Web está a evoluir. Está a evoluir convergindo, dentro das suas limitações e especificidade, com o mundo Desktop. Desta forma, constatamos a transposição de diversos paradigmas e modelos de interacção típicos em aplicações Desktop, para uma nova geração de aplicações Web potenciadas pela utilização da arquitectura Ajax.

À tradicional pergunta sobre se a Web ou o Desktop se irão anular no futuro, o trabalho defende e sustenta a teoria que não, que esse fenómeno não deverá acontecer. Efectivamente, ao longo deste trabalho, foram identificados pressupostos e restrições de base nas tecnologias Web e Desktop conforme as conhecemos actualmente, que garantem que a fusão num único mundo não seja muito provável.

Sustenta no entanto, um cenário provável onde a convergência continue a evoluir limando, cada vez mais e melhor, as arestas que diferenciam as aplicações Web e Desktop, até que o resultado para o utilizador seja uma barreira invisível entre aplicações dos dois tipos. A consistência e coerência entre os paradigmas de utilização será de tal ordem, que o utilizador não terá de se preocupar com o facto de estar a trabalhar numa aplicação Web ou Desktop.

Tecnologia Ajax

Quanto ao Ajax e às perspectivas de evolução desta tecnologia no ecossistema de tecnologias Web, o facto de alicerçar a sua forma de funcionamento em tecnologias standard, convencionadas e amplamente disseminadas, representam um factor distintivo de sucesso comparativamente às restantes tecnologias.

Tirando partido das mais-valias da Web como: ubiquidade, universalidade e abertura à comunidade; não se prevêem grandes ameaças naturais ao Ajax. Pelo contrário, a sua

proximidade e respeito pelos standards fomenta a sua disseminação por todo um universo da indústria de software cujos únicos factores de estabilidade no futuro são criatividade e alinhamento com os standards lançados por grupos do trabalho pluridisciplinares e multi-facetados, como o W3C.

Interacção com o utilizador

Beneficiando directamente com as tendências de convergência entre Web e Desktop, assim como, do forte impacto do Ajax na nova geração de aplicações Web que começam a chegar ao *maistream*, a sofisticação da experiência de utilização destas aplicações tem crescido de forma significativa.

Ao longo do trabalho apresentamos e justificamos argumentos que defendem que a melhoria da experiência de utilização nas aplicações Web se deve principalmente a dois factores:

- Tendências de convergência entre Web e Desktop.
- Maturação e disseminação da tecnologia Ajax.

Para aqueles que consideram estas conclusões uma interpretação óbvia ou subjectiva de um fenómeno em curso, o trabalho capta a essência desta evolução através da identificação de um conjunto particular de Padrões de interacção com o utilizador, que comprovam a transposição para a Web de paradigmas de interacção, típicos de aplicações Desktop, mas impraticáveis em aplicações Web antes do advento do Ajax.

Considerações Finais

A extensão do trabalho realizado permitiu-me concretizar um estudo alargado sobre os modelos e paradigmas de interacção com o utilizador em aplicações de software, assim como, compreender o processo evolutivo dos mesmos tendo em conta o passado histórico, o presente e os cenários mais prováveis para o futuro.

E ainda que os pressupostos base conhecidos actualmente para ambos os mundos - Web e Desktop – possam mudar, a noção histórica e conhecimento do presente indicam o **Utilizador** como a origem da necessidade e consequentes inovações tecnológicas.

Nesse sentido, no que toca aos modelos cognitivos de interacção com o computador, surgem, do meu ponto de vista, duas direcções:

- Convergência e Consistência, como motores da **Evolução Natural**.
- Ruptura total e completa com os modelos actuais, e estaremos a falar de **Revolução**.

Fica então a pergunta sobre qual o cenário mais provável para o futuro dos paradigmas de interacção entre utilizadores e aplicações, sejam elas Web ou Desktop.

Referências bibliográficas

- [1] Moritz, Florian, ***Rich Internet Applications (RIA): A Convergence of User Interface Paradigms of Web and Desktop Exemplified by JavaFX***, University of Applied Science Kaiserslautern, Deutschland, 2008
- [2] Welie, Martijn, ***Patterns in Interaction Design***, 2008, <http://www.welie.com/patterns/>, Acesso: 2009-03-03
- [3] Mahemoff, Michael, ***Ajax Design patterns: Creating Web 2.0 Sites with Programming and Usability Patterns***, USA, O'Reilly, 2006, ISBN: 0-596-10180-5
- [4] Gross, Christian, ***Ajax Patterns and Best Practices***, USA, Apress, 2006, ISBN: 1-59059-616-1
- [5] Sambells, Jeffrey, ***DOM Scripting: Dynamic Web Design Techniques***, USA, Friendsoft, 2007, ISBN: 1-59059-856-3
- [6] Snook, Jonathan, ***Accelerated DOM Scripting with Ajax, APIs, and Libraries***, USA, Apress, 2007, ISBN: 1-59059-764-8
- [7] 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: 2009-03-04
- [8] ***Web 2.0 Conference***, 2004, <http://www.web2con.com/web2con/>, Acesso: 2009-03-06
- [9] ***International standards for HCI and usability ISSO 942-11: Guidance and Usability (1998)***, http://www.usabilitynet.org/tools/r_international.htm#9241-11, Acesso: 2009-03-06
- [10] Nielsen, Jakob, ***Tem Usability Heuristics***, 2005, http://www.useit.com/papers/heuristic/heuristic_list.html, Acesso: 2009-03-06
- [11] Nielsen, Jakob, ***Usability Engineering: Generations of User interfaces*** (Capítulo 3), USA, Morgan Kaufmann, 1993, ISBN: 0125184069
- [12] Berners-Lee, ***The World Wide Web: Past, Present and Future***, 1996, <http://www.w3.org/People/Berners-Lee/1996/ppf.html> , Acesso: 2009-03-06
- [13] ***History of Web***, Oxford Brookes University, 2002, <http://www.w3c.rl.ac.uk/primers/history/origins.htm>, Acess: 2009-03-06
- [14] ***Browser Wars***, http://en.wikipedia.org/wiki/Browser_wars , Acesso: 2009-03-07
- [15] ***The Browser Wars***, <http://www.cssnolanche.com.br/the-browser-wars/> , Acesso: 2009-03-07
- [16] A Guerra dos navegadores, <http://videolog.uol.com.br/video.php?id=366620> , Acesso: 2009-03-07
- [17] ***Mozilla Foundation***, <http://www.mozilla.org/foundation/>, Acesso: 2009-03-07
- [18] ***Gecko***, <https://developer.mozilla.org/en/Gecko>, Acesso: 2009-03-07
- [19] ***Konqueror***, <http://www.konqueror.org/> , Acesso: 2009-03-07
- [20] ***Chrome***, <http://www.google.com/chrome> , Acesso: 2009-03-07
- [21] ***A história por trás do Google Chrome***, <http://www.youtube.com/watch?v=ARII4t8-JGk>

- , Acesso: 2009-03-07
- [22] **Browser Statistics Month by Mont**, w3schools.com, http://www.w3schools.com/browsers/browsers_stats.asp, Acesso: 2009-03-07
- [23] **Web 1.0**, Wikipedia, http://en.wikipedia.org/wiki/Web_1.0, Acesso: 2009-03-07
- [24] O'Reilly, Tim, **Web 2.0 Compact Definition: Trying Again**, (2006), <http://radar.oreilly.com/archives/2006/12/web-20-compact.html>, Acesso: 2009-03-07
- [25] Garret, Jesse James, **Ajax: A New Approach to Web Applications**, <http://www.adaptivepath.com/ideas/essays/archives/000385.php>, Acesso: 2009-03-10
- [26] **Web 3.0**, <http://www.webtuga.com/web-30/>, Acesso: 2009-03-10
- [27] **Web 3.0, the official definition**, <http://calacanis.com/2007/10/03/web-3-0-the-official-definition/>, Acesso: 2009-03-10
- [28] **Porquê ignorar a Web 3.0 e apoiar a Web 2.0**, <http://webinsider.uol.com.br/index.php/2008/08/12/porque-ignorar-a-web-3-0-e-apoiar-a-web-2-0/>, Acesso: 2009-03-10
- [29] Allaire, Jeremy, **Macromedia Flash MX - A next-generation rich client**, <http://www.adobe.com/devnet/flash/whitepapers/richclient.pdf>, Acesso: 2009-03-10
- [30] Innerphaze, **Ajax Office Review**, <http://innerphaze.homelinux.com/blog/?p=28>, Acesso: 2009-03-10
- [31] **Rich Internet Application**, Wikipedia, http://en.wikipedia.org/wiki/Rich_internet_application, Acesso: 2009-03-10
- [32] Morris, Simon, **A Rose By Any Other Name**, http://weblogs.java.net/blog/javakiddy/archive/2007/06/a_rose_by_any_o.html, Acesso: 2009-03-10
- [33] Gantz, John F., An Updated Forecast of Worldwide Information Growth Through 2011, IDC, 2008, <http://www.emc.com/collateral/analyst-reports/diverse-exploding-digital-universe.pdf>, Acesso: 2009-03-11
- [34] Sauermann, Leo, **Overview and Outlook on the Semantic Desktop**, German Research Center for Artificial Intelligence DFKI GmbH, Germany, 2005, <http://www.dfki.uni-kl.de/~sauermann/papers/Sauermann+2005d.pdf>, Acesso: 2009-03-11
- [35] Clancy, Tom, **The Infancy of the Information Digital Age**, <http://www.scribd.com/doc/9234242/Growing-Need-for-Information-Storage-Management-The-Infancy-of-the-Information-Digital-Age>, Acesso: 2009-03-11
- [36] O'Reilly, Tim, **Web 2.0 Is About Controlling Data**, Revista Wired, 2007, http://www.wired.com/techbiz/people/news/2007/04/timoreilly_0413?currentPage=all, Acesso: 2009-03-11
- [37] Garner, Bob, **SEO 2.0 And The Pageless Web: The RIA Search Conundrum**, 2006, http://www.mediapost.com/publications/?fa=Articles.showArticle&art_aid=52868, Acesso: 2009-03-11
- [38] Stewart, Ryan, **Desktop Rich Internet Application versus Web Rich Internet Application**, 2006, <http://blogs.zdnet.com/Stewart/?p=153&tag=rbxccnbzd1>, Acesso: 2009-03-11
- [39] Muller-Prove, Matthias, **Dueling Interaction Models of Personal-Computing and Web-Computing**, Klagenfurt University, Austria, 2007, <http://www.mprove.de/script/07/medichi/paper.html>, Acesso: 2009-03-11

- [40] **Direct manipulation interface**, Wikipedia, http://en.wikipedia.org/wiki/Direct_manipulation, Acesso: 2009-03-12
- [41] **Drag and drop**, http://en.wikipedia.org/wiki/Drag_and_drop, Wikipedia, Acesso: 2009-03-12
- [42] Pop, Paul, **Comparing Web Applications with Desktop Applications: An empirical Study**, Linköping University, Sweden, 2000, <http://www.ida.liu.se/labs/eslab/publications/pap/db/hci.slides.pdf>, Acesso: 2009-03-12
- [43] **Usabilidade**, Wikipedia, <http://pt.wikipedia.org/wiki/Usabilidade>, Acesso: 2009-03-12
- [44] Nielsen, **Jacob, Designing Web Usability: The Practice of Simplicity**, Peachpit, 1999, ISBN 156205810X
- [45] Tidwell, Jenifer, **Designing Interfaces**, O'Reilly Media, 2005, ISBN: 0596008031
- [46] HTTP State Management Mechanism, Network Working Group, 1997, <http://www.ietf.org/rfc/rfc2109.txt>, Acesso: 2009-03-14
- [47] W3C, **XForms 1.1: W3C Candidate Recommendation**, 2007, <http://www.w3.org/TR/xforms11/>, Acesso: 2009-03-14
- [48] Eich, Brendan, JavaScript 1, 2, and in between, 2005, <http://weblogs.mozillazine.org/roadmap/archives/008325.html>, Acesso: 2009-03-14
- [49] Aguiar, Ademar, **Reutilização de Software, Padrões e Frameworks**, Arquitectura de Sistemas de Software, FEUP, 2007 (Slides de Aulas)
- [50] Alexander, Cristopher, **A Pattern Language**, USA, Oxford University Press 1977, ISBN: 0195019199
- [51] Buschmann, Frank, **Pattern-Oriented Software Architecture Volume 1: A System of Patterns**, Wiley, 1996, ISBN-10: 0471958697
- [52] Appleton, Brad, **Patterns and Software: Essential Concepts and Terminology**, 2000, <http://www.cmcrossroads.com/bradapp/docs/patterns-intro.html#Origins>, Acesso: 2009-03-14
- [53] Bratt, Steve, **W3C – One Web: Going Mobile**, World Wide Web Consortium, 2006, [http://www.w3.org/2006/Talks/0404-sb-ctia-mwi/Overview.html#\(1\)](http://www.w3.org/2006/Talks/0404-sb-ctia-mwi/Overview.html#(1)), Acesso: 2009-03-17
- [54] Sauermann, Leo, **The Semantic Desktop - a Basis for Personal Knowledge Management**, University of Kaiserslautern, 2005, <http://www.dfki.uni-kl.de/~sauermann/papers/Sauermann2005b.pdf>, Acesso: 2009-03-22

Anexos

Anexo A – Lista de motores de busca

- [Alexa Internet](#)
- [Ask.com](#) (formerly Ask Jeeves)
- [Baidu](#) (Chinese)
- [Cuii](#)
- [Exalead](#) (French)
- [Google](#)
- [Live Search](#) (formerly MSN Search)
- [Sogou](#) (Chinese)
- [Sohu](#) (Chinese)
- [Wikia Search](#)
- [Yahoo! Search](#)

Fonte: Wikipedia, http://en.wikipedia.org/wiki/List_of_search_engines, Acesso: 2009-03-11