Universidade de Aveiro
2008

Departamento de Electrónica, Telecomunicações e Informática

Pedro Jorge Pereira Lopes

Integração de serviços para extracção de conhecimento

Universidade de Aveiro
2008

Departamento de Electrónica, Telecomunicações e Informática

Pedro Jorge Pereira Lopes

Integração de serviços para extracção de conhecimento

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Engenharia de Computadores e Telemática, realizada sob a orientação científica do Professor Doutor José Luís Guimarães Oliveira, Professor Associado do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro

Para a minha Mãe, a quem devo tudo, e para a Andreia, por algo que não posso exprimir em palavras.

o júri

presidente

Doutor Joaquim Arnaldo Carvalho Martins
Professor Catedrático da Universidade de Aveiro

Doutor Rui Pedro Sanches de Castro Lopes
Professor Coordenador do Departamento de Informática e Comunicação da Escola Superior de Tecnologia e Gestão do Instituto Politécnico de Bragança

Doutor José Luís Oliveira
Professor Associado da Universidade de Aveiro

agradecimentos

Ao Professor José Luís, pelos sábios conselhos e opiniões durante o trabalho; aos meus amigos e família, pela força dada para concluir mais esta etapa; aos colegas do grupo de Bioinformática, por me aceitarem na sua casa; e por último um agradecimento especial ao Joel Arrais, principal mentor deste projecto, pelo apoio, palavras, críticas e sugestões que tornaram possível o trabalho apresentado nesta dissertação.

palavras-chave

Web2.0, web, interface, base de dados, javascript, xml, xpath, xsl, internet, integração, extracção, ajax, serviços, css, html, workflow, informação, extracção, conhecimento, json, web services

resumo

Com a descoberta do genoma humano completa, uma enorme quantidade de dados foi gerada, criando grandes desafios na compreensão do valor de toda esta informação. Este cenário trouxe a necessidade de novas ferramentas para extracção automática de informação de bases de dados biológicas.
Esta dissertação apresenta um sistema web baseado em princípios Web2.0 desenhado para agregar serviços e dados, melhorando a extracção de conhecimento de bases de dados biológicas.
A arquitectura segue uma aproximação ágil e nova, usando fluxos de informação, que podem ser construidos dinamicamente pelo utilizador. O workflow permite a criação de protocolos de obtenção de informação específicos, permitindo que o utilizador aceda a recursos distribuídos numa interface única e centralizada.

keywords

Web2.0, interface, web, database, integration, extraction, javascript, xml, xsl, xpath, ajax, css, web services, html, services, workflow, information, extraction, knowledge, json

abstract

With the advent of human genome disclosure an enormous quantity of data was generated putting great challenges in understanding the value of all this data. This scenario has risen the necessity to have new tools for automatic extraction of knowledge from biological databases.
This dissertation presents a web system based on web2.0 principles designed to aggregate services and data and to enhance knowledge extraction from biological databases.
The architecture follows a new and agile solution using workflows which may be dynamically built by the user. The workflow allows building specific protocols for information retrieval enabling the user to access distributed resources in a centralized and unique interface.

Índice

Índice de Tabelas

Índice de Figuras

Lista de Acrónimos

AJAX	*Assynchronous Javascript and XML*
API	*Application Programming Interface*
CSS	*Cascade Style Sheet*
DB	*Database*
DBMS	*DataBase Management System*
DOM	*Document Object Model*
FAQ	*Frequently Asked Questions*
FTP	*File Transfer Protocol*
HTML	*HyperText Markup Language*
HTTP	*HyperText Transfer Protocol*
JSON	*JavaScript Object Notation*
PDA	*Personal Digital Assitant*
RSS	*Really Simple Sindication*
SGBD	*Sistema de Gestão de Bases de Dados*
SOA	*Service Oriented Architecture*
SOAP	*Simple Object Access Protocal*
UDDI	*Universal Description, Discovery and Integration*
URL	*Uniform Resource Locator*
WSDL	*Web Services Description Language*
XML	*eXtensible Markup Language*
Xpath	*XML Path Language*
XSL	*eXtensible Stylesheet Language*

1 Introdução

1.1 Enquadramento

A incessante sede de conhecimento do Homem aliada à evolução tecnológica, levou à criação de grandes quantidades de informação, que deve de ser armazenada, processada e divulgada ao público. O armazenamento de dados de forma estruturada foi a necessidade seguinte e que levou à criação de diversos sistemas de tratamento e divulgação de dados. Estes sistemas podem seguir diversos modelos e permitem uma disponibilização rápida e eficiente da informação. Um dos sistemas tradicionais para guardar a informação é o sistema de bases de dados. De entre os vários Sistemas de Gestão de Bases de Dados existentes, temos diversos esquemas de armazenamento de dados – tabelas, relações – diferentes e não relacionáveis entre si, mesmo que pertençam a áreas semelhantes e tenham sido criados no mesmo Sistema de Gestão de Bases de Dados.

Estas diferenças não se prendem apenas ao nível da apresentação e estruturação da informação, mas também ao nível das tecnologias usadas para tratamento dos dados. No caso mais comum, temos um acesso directo, através de apenas um entre vários métodos, à informação guardada nas bases de dados. Aceder apenas a um local de armazenamento de dados é demasiado restritivo, limitando a informação que podemos obter e processar. A integração de dados surge neste contexto: é necessário juntar os dados distribuídos por diversas bases de dados para que a informação extraída seja em maior quantidade, mais coerente, mais consistente e, sobretudo, de melhor qualidade. O trabalho apresentado nesta dissertação situa-se na área de desenvolvimento das interfaces de utilizador que permitem o uso de todas as potencialidades inerentes à integração de dados e serviços. Aplicando conceitos existentes e reutilizando algumas tecnologias, podemos criar interfaces com mecanismos que permitam uma gestão melhorada do acesso aos dados e que providenciam ao utilizador uma maior imersão no controlo da obtenção da informação pretendida.

1.2 Objectivos

O principal objectivo deste trabalho consiste na construção de uma aplicação *web* que forneça uma interface inovadora para gestão de fluxos de informação, com a finalidade de extrair conhecimento através da integração de dados e serviços dispersos.

Para atingir os objectivos, o utilizador da aplicação *web* tem de construir o *workflow* de modo a obter a informação desejada. Um *workflow* consiste numa lista de módulos de processamento ordenados, fechados e independentes entre si; estes adaptadores apenas comunicam ao receberem como dados de entrada, os dados de saída do adaptador anterior. A aplicação deve disponibilizar informação sobre todas as funcionalidades passíveis de serem usadas no *workflow*. O utilizador deverá depois usar estas funcionalidades pré-definidas para construir dinamicamente um *workflow* que lhe permita obter a informação que necessita. Informação sobre os diversos passos do fluxo de informação deve ser fornecida ao utilizador e a apresentação da informação deve variar de acordo com o tipo de informação a ser mostrada.

Sucintamente, os principais objectivos da dissertação foram os seguintes:

- estudar os novos modelos de interfaces de aplicações de integração de dados existentes;
- gerir dinamicamente o acesso a funcionalidades de diversas bases de dados;
- criar uma interface *web* ágil para criação e manutenção de mecanismos de extracção e agregação de informação – *workflows*, procurando-se oferecer uma interacção semelhante à que pode ser dada por uma aplicação *Desktop*.

1.3 Estrutura

Esta dissertação encontra-se dividida em cinco capítulos.

No capítulo 2 é apresentado o estado da arte relativo a metodologias e arquitecturas de integração de dados e serviços. O foco é dado à camada aplicacional de onde fará parte a interface de que trata a dissertação. Neste âmbito são referidas

as novas aplicações Web2.0, as suas características e também as tecnologias que permitem este novo tipo de aplicações e interfaces. São também referidos alguns serviços *web* usam estas tecnologias e se baseiam em *workflows* para integração de dados.

No capítulo 3 é apresentada a análise de requisitos, a modelação do sistema e uma proposta para a arquitectura da aplicação.

O quarto capítulo descreve a estrutura da aplicação criada, as escolhas efectuadas para responder aos requisitos definidos e os problemas encontrados durante o desenvolvimento da aplicação. É ainda feita uma contextualização da aplicação dentro da área da bioinformática.

No capítulo final faz-se um balanço do trabalho realizado, são apresentadas as conclusões do trabalho e apontados caminhos para o futuro da arquitectura.

2 Integração de dados

A integração de dados é uma área cada vez mais importante no tratamento da informação, principalmente devido ao facto desta se encontrar cada vez mais dispersa. A informação encontra-se distribuída das mais variadas maneiras e a integração desta informação nem sempre se revela uma tarefa simples.

Fontes de dados heterogéneas causam problemas na integração a 3 níveis [1]:

- problemas no estrutura dos dados, pois podemos ter ficheiros, tabelas ou outros formatos, sem qualquer tipo de relação ou mapeamento directo;
- problemas no esquema de armazenamento dos dados, isto porque, mesmo com uma estrutura semelhante, conceitos semelhantes podem ser guardados com representações diferentes, representações semelhantes podem modelar conceitos diferentes e ainda porque existem diversas formas de representar um mesmo conceito. Este problema também não é ultrapassado mesmo que se siga um esquema semelhante na estrutura de armazenamento – modelo relacional na 3.ª forma normal por exemplo;
- problemas na instanciação dos dados, pois podemos ter o mesmo objecto representado em fontes de dados diferentes e de maneiras distintas. Assim, a integração pode ser feita de diversas formas e envolve geralmente mais que uma camada de abstracção na arquitectura de acesso aos dados.

Um dos métodos de acesso a dados, *brokerless*, envolve apenas configurações ao nível do nosso Sistema de Gestão de Bases de Dados (SGBD) (Figura 1). O SGBD usa geralmente a *Structured Query Language* (SQL) para permitir o acesso à informação. Usando um SGBD, podemos configurá-lo de forma a que o acesso a diferentes bases de dados seja transparente para o utilizador desse SGBD. O utilizador pode então efectuar as respectivas *queries* SQL e obter respostas de variadas fontes de informação de uma forma simples e sem se preocupar com

problemas de programação. O SGBD tem toda a carga de trabalhos o que pode reduzir a eficiência da aplicação devido às latências que o processamento do SQL causaria.

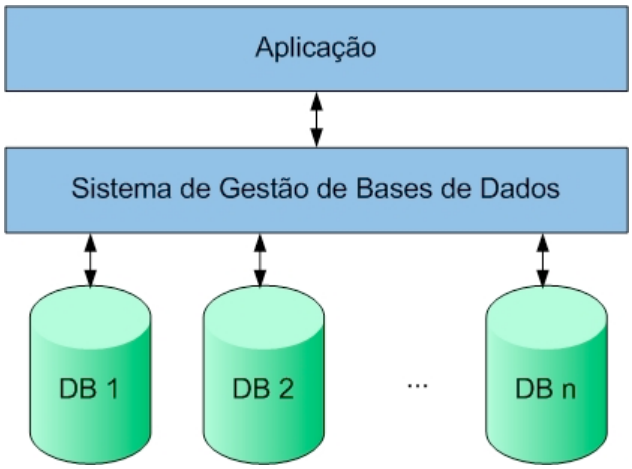


Figura 1 : Integração de Dados a nível de SGBD

Nesta situação, há também que considerar todos os problemas inerentes ao uso de Bases de Dados distribuídas e as suas implicações na arquitectura do sistema a implementar, fazendo com que esta solução seja usada poucas vezes.

O segundo caso para integração de dados envolve a criação e desenvolvimento de um *broker* (Figura 2). Este *broker* terá de ser programado para fornecer à camada de abstracção superior um acesso transparente aos dados.

Tanto o acesso aos dados como o retorno de dados para a camada de abstracção superior podem ser feitos de diversas formas.

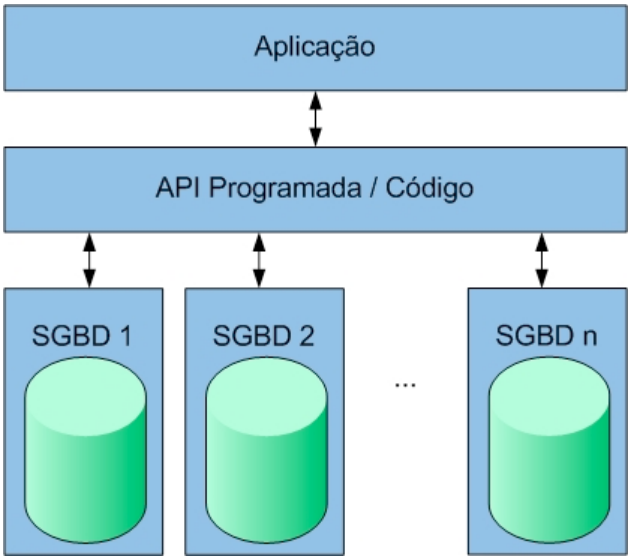


Figura 2 : Integração de Dados a nível de código

Para aceder aos dados existentes podemos programar a nossa aplicação para usar uma das várias implementações de SGBD existentes, usando as suas respectivas potencialidades. Podemos também aceder aos dados através de *WebServices*, tecnologia cada vez mais em voga na disponibilização de acesso a informação e que oferece potencialidades semelhantes a uma arquitectura distribuída. Muitas bases de dados de onde queremos extrair informação possuem *Application Programming Interfaces* (APIs) próprias. Deste modo, temos de adaptar o nosso *broker* para poder usar estas APIs. O uso de APIs ou *Web Services* é a opção cada vez mais usada, visto limitar o acesso à informação existente, bem como esconder e preservar muito do trabalho feito em termos de modelação e estruturação da base de dados.

No fornecimento de dados à camada aplicacional superior podemos seguir diversos caminhos. Podemos usar estruturas de dados internas à nossa aplicação, usável caso estejamos a construir uma aplicação proprietária e caso o nosso código não seja alvo de divulgação pública. Em situações em que pretendemos que o acesso ao que construímos seja usado por mais que uma aplicação, tendo estas fins distintos e arquitecturas distintas, é necessário optar por métodos que facilitem o acesso à informação que conseguimos obter. Tal como os proprietários das bases de dados, podemos optar por criar e disponibilizar *Web Services* que permitem o acesso à informação, podemos criar uma API própria para acesso à nossa estrutura ou então disponibilizar os dados num formato específico. Escolhendo esta última hipótese, o

XML surge como o formato ideal pela sua ampla divulgação e pela facilidade com que outras aplicações podem aceder aos dados neste formato.

Depois de resolvido o acesso aos dados é importante trabalhar sobre as possibilidades de tratamento e apresentação. É neste nível de abstracção que surge o trabalho apresentado nesta dissertação. Este nível pode ser desenvolvido de diversas maneiras: podemos optar por criar uma aplicação típica *Desktop*, podemos construir uma aplicação para correr num PDA, podemos construir uma aplicação para correr num tipo de *hardware* específico ou podemos desenvolver ainda aplicações para ambiente *web* (Figura 3).

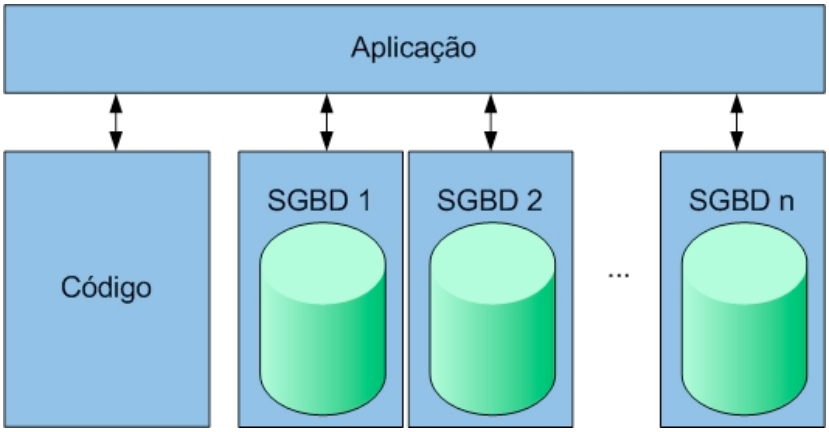


Figura 3 : Integração de Dados a nível da aplicação

2.1 Aplicações Web para Integração de Dados

A integração de dados começa a surgir cada vez mais associada a aplicações Web e não só a aplicações *Desktop*. Desde o turismo à saúde, do entretenimento à educação, a integração de dados como acesso a informação pode ser aplicada a todas as áreas existentes e, apesar de ainda serem questionáveis pelos mais puristas, as aplicações *web* surgem cada vez mais na linha da frente nas áreas em que a digitalização da informação é fulcral.

Sendo a integração de dados um dos principais pontos do trabalho mostrado nesta tese, importa referir o que já existe feito pelo grupo de Bioinformática da

Universidade de Aveiro numa área semelhante à que será abordada nesta dissertação. A aplicação DiseaseCard é um "*portal Web público que integra em tempo real informação genómica e médica de bases de dados heterogéneas e distribuídas, e apresenta-a num paradigma visual familiar*" [2]. Esta aplicação permite recolher informação em tempo real de diversas fontes de dados disponíveis na Internet acerca de um determinado item de pesquisa relacionado com doenças raras. Estas fontes de dados encontram-se dispersas e são desconexas entre si, tornando a integração de dados destas um problema complexo e aliciante. O GeneBrowser, "*um novo serviço Web (...), que combina diversas fontes de dados e métodos de visualização para melhorar a interpretação biológica e a extracção de conhecimento de um grupo de genes*" [3], envolve uma arquitectura de integração de dados semelhante ao DiseaseCard. Esta aplicação serviu de exemplo e de ponto de partida para o presente trabalho.

2.2 Web2.0

Actualmente, as aplicações *Desktop* dominam o mercado, principalmente devido ao facto de serem muito mais ricas e interactivas que as aplicações *web*. As aplicações *Desktop* conseguem ser mais eficientes, mais rápidas e claramente melhores na interacção com o utilizador uma vez que podem usar capacidades avançadas do *hardware*, tanto a nível gráfico, como ao nível de capacidades de armazenamento e memória, algo que nunca poderá ser igualado num *browser*. Obviamente, a estrutura complexa das aplicações *Desktop* traz desvantagens: é sempre necessária uma instalação, existe geralmente uma dependência do sistema operativo, as actualizações podem ter de ser despoletados pelo utilizador e envolvem na maioria das vezes uma ligação à Internet.

São estes problemas que as aplicações *web* tentam usar a seu favor para conquistar o seu espaço. Uma aplicação *web* pode sofrer actualizações inteligentes e mudanças profundas na arquitectura da aplicação de uma forma completamente transparente e inócua para o utilizador. Permite que se poupe tempo e dinheiro, pois não necessita de instalação, e, muitas vezes, ultrapassa as restrições impostas por

normas de empresas que não permitem que os seus funcionários instalem *software* nas suas máquinas. As aplicações *web* têm também uma grande vantagem em termos de alcance: as aplicações desenvolvidas seguindo paradigma *web* podem correr em qualquer *browser*, em qualquer sistema operativo e a partir de qualquer parte do globo em qualquer hora do dia.

O programador, no início do desenvolvimento da sua aplicação, terá de optar por uma aplicação "*rich or reach*" [4], ou seja, por uma aplicação *Desktop* tipicamente mais rica a todos os níveis, ou por uma aplicação *web* em que o alcance de utilizadores finais é muito maior. A escolha é sem dúvida complicada, pois existirá sempre uma fronteira entre estes dois tipos de aplicação.

É na diminuição das fronteiras entre as tecnologias *Desktop* e *Web* que têm surgido conceitos inovadores: *frameworks* como o AJAX aliadas a um sustentado desenvolvimento permitem que as aplicações *web* sejam tão eficientes e ricas a nível gráfico como uma aplicação *Desktop*, melhorando a interacção com o utilizador em todos os aspectos.

A Web2.0 "surgiu" como forma de denominar os novos conceitos de interacção com o utilizador e a nova vaga de aplicações *web* cada vez mais semelhantes, em todos os níveis, das aplicações *desktop*.

2.2.1 Origens

O termo Web2.0 não é usado no contexto de mudança e/ou criação de protocolos ou linguagens para melhorar a forma de comunicar com o utilizador. É sim uma mudança no uso que se dá a essas tecnologias e, acima de tudo, uma mudança na mentalidade de aproximação ao utilizador final. Web2.0 é um regresso às ideias originais da Internet, em que o conteúdo deve ser criado e partilhado entre todos os utilizadores.

Com a Web2.0 o poder é dado ao utilizador. O poder de criar, o poder de alterar e o poder de destruir também. O conteúdo na Internet deixou de estar exposto para consulta em *sites* fechados e mantidos por equipas de técnicos de informática, para passar a ser acedido, criado, mantido e alterado por utilizadores com cada vez

menos conhecimento técnico de como os diversos elementos tecnológicos da Internet se interligam.

O termo Web2.0 foi introduzido por Tim O'Reilly na primeira conferência sobre Web2.0 em 2004 [5]. Nessa conferência foram defendidos os princípios sobre os quais assentava esta nova Internet: a *Web* como uma plataforma, dados como sendo o motor da Internet, facilidade de adopção de novos utilizadores e manutenção dos antigos, novos modelos de negócio orientados ao conteúdo, uma arquitectura participativa e o contínuo desenvolvimento de aplicações, estando estas continuamente numa fase *beta*. Deste último aspecto nasceu uma das marcas gráficas da Web2.0: o símbolo *beta* omnipresente durante largos períodos de tempo em todos os sites que se auto-intitulam de Web2.0. Mas não foi só esta introdução que propulsionou as mudanças no aspecto das páginas. A crescente facilidade de criação de páginas HTML, o CSS, o AJAX e a popularidade de diversos *softwares* de edição de imagem levaram a uma *Web* mais apelativa e agradável à vista. Acabaram as páginas com fundo branco, tipo de letra *Times New Roman* a preto e ligações simples a azul. A Web2.0 traz-nos um bom gosto e uma preocupação acrescida no *look n' feel* que se transmite aos utilizadores.

A Web2.0 não trouxe consigo a criação de uma nova tecnologia revolucionária. Surgiu apenas uma redescoberta das tecnologias já existentes, associando-as a novos conceitos, o que permitiu que a Web2.0 crescesse sem grandes desenvolvimentos a nível de programação.

2.2.2 Javascript

Javascript (ECMAScript) [6] é a linguagem de *script* mais amplamente usada para enriquecimento de aplicações *web* no lado do cliente. A sua principal consiste na possibilidade de ser usada como complemento ao código HTML de forma a permitir um aumento da interactividade entre o utilizador e a página *web*. Isto acontece já que Javascript corre localmente no *browser* do utilizador, o que possibilita que as aplicações *web* sejam mais interactivas.

O poder do Javascript reside no facto de trabalhar directamente com o *Document Object Model* (DOM) [7] que compõe a página *web*. Interagindo com este

modelo, o Javascript consegue modificar o aspecto das páginas, alterando o seu conteúdo de acordo com as especificações da aplicação. Isto é possível, pois o acesso ao DOM permite também identificar operações realizadas pelo utilizador e despoletar acções de acordo com essas operações.

As típicas janelas *pop-up* ou a identificação do *browser* e do sistema operativo que estamos a usar são algumas das funcionalidades típicas obtidas pelo Javascript.

```
GetListHtml : function()
{
    var webRequest = new Sys.Net.WebRequest();
    webRequest.set_url("Modules.aspx?op=validation");
    webRequest.add_completed(Dataset.ValidateGenelist);
    webRequest.invoke();
},

GetValidationHTML : function()
{
    if(typeof(obj) != undefined)
        Dataset.GetListHtml();
    else
    {
        setTimeout("Dataset.GetValidationHTML();", 5000);
    }
},
```

Figura 4 : Exemplo de código Javascript

2.2.3 XML

O formato *eXtensible Markup Language* (XML) [8] é uma metalinguagem que nos possibilita definir as nossas próprias *markup languages* de forma estruturada. Uma *markup language* é um mecanismo que permite identificar de uma forma sintética as diversas partes de um documento, normalmente recorrendo ao uso de *tags*. Podemos, com o XML, "criar" a nossa própria linguagem e guardar informação hierarquizada e complexa de uma forma legível, sem recorrer a qualquer *software* específico. A ideia primordial para o desenvolvimento deste *standard*, foi a criação de uma norma que permitisse que documentos especificados de uma forma estruturada pudessem ser partilhados através da Internet.

Os documentos XML são estruturados com um prólogo seguido de elementos. Cada um destes elementos pode conter mais elementos e atributos dentro dele,

providenciando uma hierarquia de regiões bem delimitadas cada uma delas com um propósito específico.

```
<?xml version="1.0" encoding="utf-8" ?>
<persons>
  <person id="1">
    <firstName>Pedro</firstName>
    <lastName>Lopes</lastName>
    <email>pedrolopes@ua.pt</email>
  </person>
  <person id="2">
    <firstName>José</firstName>
    <lastName>Oliveira</lastName>
    <email>jlo@ua.pt</email>
  </person>
  <person id="3">
    <firstName>Joel</firstName>
    <lastName>Arrais</lastName>
    <email>jpa@ua.pt</email>
  </person>
</persons>
```

Figura 5 : Extracto de um ficheiro XML

O uso de XML traz diversos benefícios, de entre os quais há que destacar os seguintes:

- o facto de ser livre e independente de programas ou plataformas, ou seja, podemos manipular ficheiros XML em Mac OS X que serão igualmente lidos e interpretados em Windows, caso a codificação dos ficheiros seja compatível;
- o XML é, como o nome indica, extensível, ou seja, podemos criar as nossas próprias *tags* e fazer actualizações à nossa "linguagem" conforme necessário e mantendo a compatibilidade com versões anteriores;
- a representação dos dados está normalizada e quase todas as linguagens de programação já têm um suporte construído para leitura e escrita de ficheiros XML, o programador não necessita, neste caso, de estar a "redescobrir a roda";
- é legível para o utilizador comum, qualquer *browser* permite que sejam abertos ficheiros *.xml* e estes podem ser lidos e compreendidos facilmente.

Associado ao XML surgiram outras normas como a *eXtensible Stylesheet Language Transformations* (XSLT) [9] e a *XML Path Language* (XPath) [10].

A XPath é uma linguagem que permite a selecção de elementos de um determinado documento XML de acordo com uma expressão específica. A XPath lê o documento XML como uma árvore de elementos e relações entre eles; deste modo podemos, aceder ao conteúdo de qualquer elemento de um ficheiro XML com uma simples expressão XPath. Atendendo ao exemplo de ficheiro XML apresentado anteriormente (Figura 5), avaliando sobre ele a expressão XPath "*persons/person*" obteremos todos os elementos "*person*" que são filhos de "*persons*".

A XPath serve de base às funcionalidades trazidas pelas XSLT. Esta segue uma aproximação funcional na transformação de ficheiros XML. Usando XPath podemos fazer uma selecção de elementos e aplicar-lhes as transformações que pretendemos. A associação destas duas normas permite-nos efectuar uma transformação parametrizada e personalizável de documentos XML noutro tipo de documentos ou num documento XML estruturado de maneira diferente. As XSLTs permitem capacitar a interoperabilidade entre diversas aplicações: o mesmo ficheiro XML pode ser transformado de formas diferentes de acordo com as necessidades da aplicação, facultando que o mesmo formato de dados seja partilhado entre várias aplicações com objectivos diferentes.

O XML "*está rapidamente a tornar-se a norma para representação e troca de dados. Fornece um formato comum para expressar tanto estruturas de dados como conteúdos. Assim, pode ajudar na integração de dados estruturados, semi-estruturados ou não estruturados na Web*" [1]. Além de ser um formato essencial na Web2.0, o XML é também um formato importante na integração de dados pois, através dele, podemos definir normas e esquemas para facilitar a integração de dados e serviços, principalmente nas aplicações Web.

2.2.4 JSON

O Javascript Object Notation (JSON) [11] é um formato de tratamento de dados bastante leve e cada vez mais usado na troca de informação entre aplicações. Estabelecendo uma comparação directa com o XML, o JSON é vantajoso em termos de tamanho e de *overhead* na estrutura de dados e também no facto de ser mais

facilmente gerado e tratado pelas aplicações. Tal como o XML, é um formato baseado em texto.

As estruturas dos objectos em JSON são universais e suportadas por todas as modernas linguagens de programação: temos pares chave – valor ou então listas ordenadas de valores.

No contexto deste trabalho, os objectos JSON podem ser considerados de significativa importância. Isto porque, tal como o nome indica, o formato JSON vem da linguagem Javascript, a linguagem utilizada para tratar os dados do lado do cliente e a sua integração nesta linguagem é natural e sem problemas: "*no muss or fuss*" [11]. Esta integração com o Javascript é benéfica no que diz respeito à criação de interfaces, pois podemos tratar o objecto JSON directamente, em vez de efectuar o *parsing* do XML para obter a informação pretendida.

```
var myJSONObject = {"persons": [
        {"firstName": "Pedro", "lastName": "Lopes", "email": "pedrolopes@ua.pt"},
        {"firstName": "José", "lastName": "Oliveira", "email": "jlo@ua.pt"},
        {"firstName": "Joel", "lastName": "Arrais", "email": "jpa@ua.pt"}
    ]
};
```

Figura 6 : Exemplo de objecto em JSON

Apesar de "*Web Services serem praticamente sinónimo de XML*" [12], o formato JSON está numa crescente conquista de espaço devido às suas principais características: leveza, simplicidade e integração natural com o Javascript.

2.2.5 *Web Services*

Um *Web Service* [13] é meramente uma interface de programação distribuída sobre a *web*, o seu método de acesso é através da Internet e as operações são executadas num servidor remoto. *Web Services* são o mais comum exemplo de *Service Oriented Architecture* (SOA).

Os *Web Services* funcionam sobre HTML e XML, sendo o XML a base de comunicação entre o cliente e o servidor. No entanto, as respostas do servidor podem estar estruturadas de acordo com diversos formatos: HTML, texto, JSON. As

mensagens XML são formatadas de acordo com a norma *Simple Object Access Protocol* (SOAP) [14] e os *Web Services* têm de estar descritos de acordo com a *Web Services Description Language* (WSDL) [15] e registados (para poderem ser encontrados) de acordo com a norma *Universal Description, Definition and Integration* (UDDI) [16]. Estas três normas juntas (SOAP, WSDL e UDDI) permitem a fácil interacção entre duas aplicações comunicando sobre Http, ultrapassando as barreiras trazidas pelos *proxies* e pelas *firewalls*.

Linguagens de programação como o C# [17] têm já uma implementação de *Web Services* de grande qualidade. Nesta linguagem, as referências a *Web Services* são tratadas como qualquer outra referência local, possibilitando que a programação de aplicações com C# que usem *Web Services* seja extremamente simples. O programador não necessita de ter um conhecimento profundo de como os *Web Services* funcionam para poder usá-los no desenvolvimento das suas aplicações.

O uso de *Web Services* está a tornar-se cada vez mais vulgar para quem pretende fornecer uma API de acesso de alto nível a serviços que detém. Os *Web Services* são talvez uma das formas mais simples de integrar várias aplicações *web* numa arquitectura distribuída. Usando *Web Services* podemos estabelecer formas de contacto entre diversas aplicações *web* sem que qualquer uma delas tenha conhecimento da estrutura ou do conteúdo programático da outra.

2.2.6 CSS

As *Cascading Style Sheets* (CSS) folhas de estilo são um "*um mecanismo simples para adicionar estilo (ex.: tipos de letra, cores, espaçamento) a documentos Web*" [18]. Usando folhas de estilo podemos personalizar a aparência da nossa aplicação Web sem termos de recorrer à especificação particular de cada componente.

```
body
{
    font: 80% Verdana, Arial, Helvetica, sans-serif;
    background: #FFF;
    margin: 0;
    padding: 0;
    text-align: center;
    color: #333333;
}
#container
{
    width: 100%;
    background: #FFFFFF;
    margin: 0 auto;
    text-align: left;
}
```

Figura 7 : Extracto de um ficheiro CSS

Ao associar as CSS à funcionalidade do Javascript, é-nos oferecido um leque mais alargado de possibilidades para a criação de interfaces mais apelativas que as tradicionais, uma vez que alterações dinâmicas ao estilo da nossa página podem ser feitas enquanto navegamos. Estas permitem uma melhor ilustração de determinada funcionalidade ou operação executada. As folhas de estilo são um dos componentes da aproximação AJAX, descrita de seguida.

2.2.7 AJAX

O *Assynchronous Javascript and XML* (AJAX) não é, por si só, uma nova tecnologia, mas sim, tal como o nome parcialmente indica, uma aliança formada entre tecnologias já existentes: o Javascript, o XML, as CSS e o DOM. Sucintamente, o AJAX consiste em usar Javascript e XML para efectuar pedidos a um servidor dinamicamente e, com a resposta, mudar a aparência das páginas em tempo real. Diferem das aplicações estáticas antigas no que diz respeito ao tratamento de pedidos. Estas, ao efectuarem um pedido, obrigavam a uma limpeza da janela de *browsing* seguida de um tempo de espera e só depois apareciam os resultados requeridos. Com AJAX, podemos actualizar apenas as partes da página que pretendemos e oferecer ao utilizador um novo tipo de experiência: mais eficiente, mais rápida, mais usável, mais interactiva e mais funcional no uso de aplicações *web-based*.

O conceito essencial do AJAX é o seu assincronismo, possível devido aos objectos *XMLHttpRequest*. O significado de assincronismo nesta área refere-se ao facto de serem efectuados pedidos ao servidor e as suas respostas serem tratadas sem qualquer interferência visível para o utilizador: as trocas acontecem nos "bastidores". Para efectuar pedidos ao servidor é usado o objecto *XMLHttpRequest*. Após a criação e configuração deste mesmo objecto, pode-se proceder a uma chamada a uma página, basta depois definir o que fazer com os dados obtidos para que estes sejam tratados. O tipo de objectos devolvidos pode ser um documento XML, um documento HTML, um documento de texto simples ou um objecto JSON.

A Figura 8 representa uma interacção AJAX simples. Os passos de um pedido AJAX encontram-se ordenados numericamente. Depois de carregada a página, o Javascript fica à escuta de um evento, como o clique do rato ou a pressão numa tecla. A esse evento está associada uma função que será chamada aquando da sua ocorrência (Figura 8 – 1). Nessa função é criado e configurado um objecto do tipo *XMLHttpRequest*. De seguida, o objecto *XMLHttpRequest* executa um pedido ao servidor (Figura 8 – 2), este pedido é depois processado pelo servidor que devolve a resposta ao cliente (tal como um pedido Http normal) (Figura 8 – 3). Em caso de sucesso na obtenção de uma resposta, o *XMLHttpRequest* passa-a à função desejada que depois irá tratá-la como pretendido, geralmente alterando alguma parte do DOM no *browser* (Figura 8 – 4).

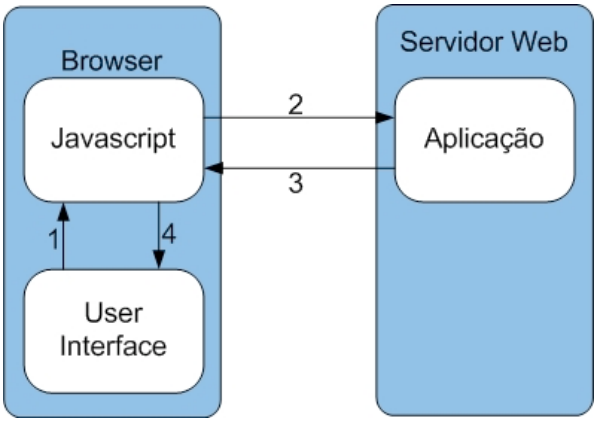


Figura 8 : Interacção cliente/servidor AJAX

Este tipo de interacção permite a existência de pequenas funcionalidades que o utilizador comum não se apercebe, mas que envolvem alguma complexidade. De entre essas funcionalidades há que destacar as seguintes:

- formulários auto-completáveis, em que os dados são automaticamente preenchidos de acordo com a informação disponível;
- submissão parcial, em que dados de formulários são enviados e processados pelo servidor;
- validação de dados em tempo real, os dados de formulários que requerem validação por parte do servidor podem ser validados sem interferência do utilizador;
- *mashups*, as páginas podem ter componentes provenientes de partes distintas da *web*;
- pré-carregamento de dados, as aplicações *web* podem basear-se nos eventos despoletados pelo cliente e numa previsão dos seus passos seguintes para carregar dados, acelerando o *download* da página;
- efeitos e controlos de interface mais rápidos e sofisticados.

Este tipo de funcionalidades, possíveis com recurso a AJAX, possibilita mais e melhores interacções com os utilizadores das aplicações Web. O utilizador tem mais controlo sobre as suas acções na aplicação e a aplicação é muito mais rica e funcional, aproximando-se das aplicações Desktop e dando ao utilizador, total controlo sobre a aplicação: um dos ideais Web2.0.

A utilização de AJAX tem uma importante vantagem que consiste na separação entre dados, estilo, formatação e funcionalidade (Figura 9). Pensando na utilização do AJAX desde a primeira fase de concepção da aplicação, o desenvolvimento da aplicação pode incluir uma separação funcional benéfica. Os dados podem ser incluídos num formato próprio, num ficheiro XML, como simples texto ou tratá-los directamente num formato próprio para bases de dados. É possível formatar a página HTML de modo a estar preparada para as diversas possibilidades de interacção com o DOM. Pode-se configurar o aspecto da nossa página com recurso a uma ou mais folhas de estilo CSS, que permitem definir todos os elementos intervenientes na aparência final da nossa página.

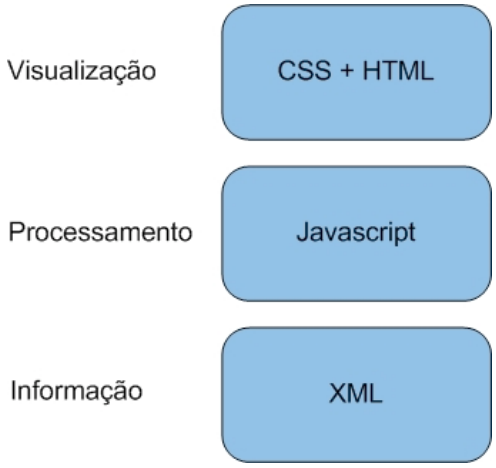


Figura 9 : Organização modular do AJAX

Claro que existe o reverso da medalha. A dependência do Javascript pode revelar-se problemática pois os diversos *browsers* existentes não suportam as mesmas funcionalidades da linguagem e nem tratam eventos semelhantes da mesma forma. O AJAX constitui também uma barreira para a optimização perante motores de busca e para as ferramentas de análise de *sites*. Isto porque, sendo o conteúdo gerado quase na sua totalidade dinamicamente e dependendo do Javascript, os dados mostrados não se encontram disponíveis através de um endereço particular e os motores de busca não executam instruções de Javascript, porque estas dependem, na sua maioria, de eventos despoletados pelo utilizador. Existem também alguns problemas no que toca à latência provocada pelos diversos pedidos ao servidor, que podem levar a que o utilizador fique à espera que aconteça algo sem ter qualquer *feedback* por parte da aplicação. Mas a grande desvantagem prende-se com a integração do AJAX no *browser*. Sendo quase todo o conteúdo criado dinamicamente, não podemos efectuar a simples operação de "retroceder" no *browser*, pois este não sabe o estado em que a aplicação se encontrava nem como chegar de novo a esse estado. Esta característica também não permite a gestão de favoritos da aplicação, uma vez que o que será guardado é o URL da barra de endereço do *browser* e não uma representação do estado em que a aplicação se encontra.

Contudo, o desenvolvimento para colmatar estas falhas tem feito grandes progressos e o AJAX começa a ser cada vez mais usado em aplicações Web de alguma complexidade.

2.3 Exemplos de aplicações Web2.0

A Web2.0 trouxe uma nova vaga de aplicações. O valor e a funcionalidade da maioria destas aplicações pode ser questionado. No entanto, vieram modificar o quotidiano dos utilizadores da Internet – a nível profissional ou de entretenimento, mesmo que estes não se apercebam desse facto.

De seguida é feita uma pequena apresentação das mais relevantes aplicações que usam as ideias trazidas pela web2.0.

2.3.1 *Folksonomy*

A grande mudança na maneira de pensar a Internet surgiu com a introdução da denominada *folksonomy*. Sem qualquer tradução possível para Português, este termo reflecte o aspecto mais importante da nova Web: as *tags*, ou etiquetas. Poder categorizar o conteúdo do nosso trabalho, das nossas ligações, dos nossos vídeos, dos nossos artigos, permite um melhor acesso à informação e o aparecimento de algoritmos de procura muito mais eficazes a extrair o que realmente se pretende ao carregar no botão "Procurar".

A grande vantagem no uso de *tags* prende-se com o facto do papel da categorização ser atribuído ao Homem e não a máquinas e seus algoritmos. Podemos associar melhor a respectiva *tag*, ou conjunto de *tags*, ao contexto do que está a ser categorizado. Isto providencia melhores resultados, tempos de pesquisa mais rápidos e maior satisfação por parte do utilizador final. Como é óbvio existem problemas inerentes ao uso de alguns vocábulos, palavras homónimas são o melhor exemplo disso: "Banco" como instituição financeira ou "banco" para sentar? "Canto" de cantar ou "canto" da sala? O próprio acto de atribuir as *tags* pode não ser muito seguro, pois as pessoas tendem a usar um conjunto restrito de significados, diminuindo a pluralidade do que pode ser dito num mesmo contexto.

O uso de *folksonomies* pode levar à tão desejada Web semântica: cada página com as suas *tags* permitiria uma precisão melhor dos motores de pesquisa. A

complicação que surge neste aspecto é na criação de uma norma para definição de *tags*. O seu grande poder é também a sua grande perda. O facto de serem criadas pela mão do Homem pode levar a más intenções – *spam* com *tags* falsas, por exemplo – e a subjectividade inerente ao mesmo nunca poderá ser recriada no ambiente mecânico da Web.

A um nível mais técnico, o uso de *folksonomies* levou a pequenas mudanças na estrutura das aplicações. Nas páginas Web, o uso de etiquetagem revela-se cada vez mais importante e as etiquetas podem ser declaras na sua própria *tag* dentro do código HTML da página. Uma aplicação Web mais complexa, que trabalhe com bases de dados por exemplo, terá apenas de prever nas bases de dados uma estrutura com tabelas e relações entre elas que suporte as várias *tags* e a ligação das *tags* com o resto do conteúdo.

2.3.2 *Social Bookmarking*

O *social bookmarking* consiste, sucintamente, em partilhar os nossos favoritos através da Internet. Ao invés dos favoritos permanecerem alojados no *browser*, são partilhados através de diversas aplicações Web (Digg, Reddit, del.icio.us) com todos os outros utilizadores que visitam os mesmos *sites*. Este tipo de aplicações é tão comum que, em diversas páginas, o botão "Adicionar aos Favoritos" foi substituído por ligações que permitem publicar a página em questão, nestes serviços. Os favoritos ficam disponíveis para qualquer utilizador seguir e também guardar, se desejar. Este é um dos melhores exemplos do uso das já referidas *folksonomies*: etiquetando, partilhamos mais eficazmente os favoritos e, com novos e poderosos algoritmos de pesquisa, podemos obter resultados mais fiéis, de acordo com a relevância dada pelos utilizadores e das *tags* de cada favorito.

Tabela 1 : Aplicações de *social bookmarking*

Descrição	Endereço
Del.icio.us: agregador de favoritos	http://del.icio.us
Digg: agregador de notícias	http://www.digg.com
Reddit: agregador de favoritos	http://reddit.com

A Figura 10 mostra a interface simples do *site* del.icio.us e na tabela temos alguns exemplos de aplicações de *social bookmarking* bastante famosas na *World Wide Web*.

Apesar de parecer uma aplicação bastante complexa e ter um vasto leque de funcionalidades, a arquitectura do del.icio.us é relativamente simples e foi desenvolvida – em *part-time* – por uma pessoa apenas. O essencial é ter uma arquitectura de *tagging* eficiente e um algoritmo de pesquisa rápido, pois a estrutura da base de dados é relativamente simples.

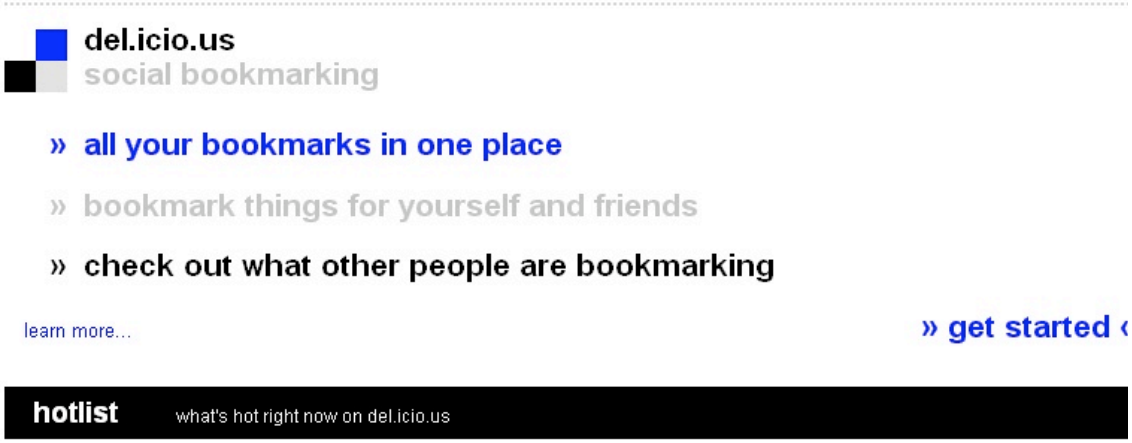


Figura 10 : http://del.icio.us

2.3.3 *Social Networking*

O *social networking* – redes sociais – teve também o seu grande *boom* com a Web2.0. Redes de amigos e conhecidos são cada vez mais frequentes e ocupam cada vez mais tempo e importância na vida e nas relações da juventude actual. Na generalidade, estas aplicações Web não são apenas redes de contactos, mas sim *sites* com muitas funcionalidades, desde a partilha de ficheiros até à criação e armazenamento de álbuns de fotos.

O funcionamento destas redes é bastante simples, começa-se pelo registo de uma conta e depois basta partilhar informação pessoal e adicionar contactos à rede de amigos.

Tabela 2 : Aplicações de *social networking*

Descrição	Endereço
Hi5: rede social bastante popular	http://www.hi5.com
MySpace: rede social de artistas	http://www.myspace.com
Orkut: rede social do Google	http://www.orkut.com

Ao nível técnico, no caso do Hi5, o SGBD escolhido foi o PostgreSQL [19] com recurso a bases de dados espalhadas por diversas zonas geográficas de forma a ter um melhor desempenho na disponibilização da informação ao cliente. O MySpace usa também várias bases de dados, dividindo a informação mais comum e mais acedida – informação pessoal – em bases de dados mais rápidas e a informação mais pesada e menos acedida – vídeos e músicas – em servidores distintos.

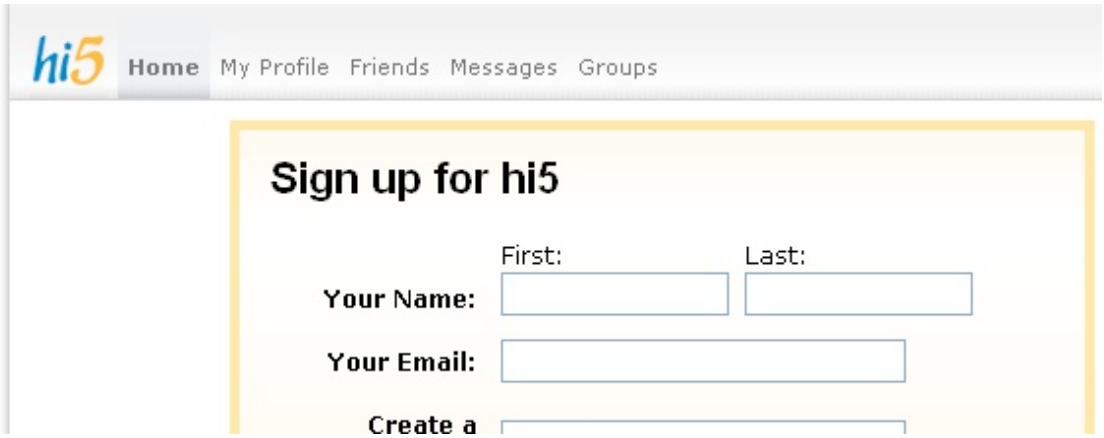


Figura 11 : http://www.hi5.com

2.3.4 *Wiki*

As *wikis* vêm substituir as convencionais enciclopédias em papel. Uma *wiki* consiste num aglomerado de páginas, com informação proveniente das mais diversas

áreas, que podem ser criadas, editadas e apagadas pelos utilizadores. O conceito surgiu em 1995 com a *WikiWikiWeb*.

A mais famosa *wiki* é a *Wikipedia*, uma verdadeira enciclopédia *online* com informação sobre quase todos os temas, mantida pelos utilizadores e disponível em mais de duas centenas de línguas. O consórcio que mantém esta *wiki* mantém também diversas outras relacionadas com outras áreas de influência como dicionários de línguas ou espécies biológicas existentes.

O alcance das *wikis* está também a mudar. Com o processo de criação de *wikis* cada vez mais simplificado, quem produz pequenas aplicações opta por criar uma *wiki* a servir de *Frequently Asked Questions* (FAQ), pois novos utilizadores podem adicionar problemas ou editar informação já existente, melhorando a funcionalidade do programa e a qualidade da sua documentação.

Como seria de esperar, todo este poder dado ao utilizador vai trazer problemas. Dados podem ser falsificados ou informação errada pode ser disponibilizada deliberadamente. Para combater este problema, a *Wikipedia* optou por bloquear o acesso a algumas das páginas mais importantes, utilizadores mal intencionados não possam fazer prevalecer os seus actos.

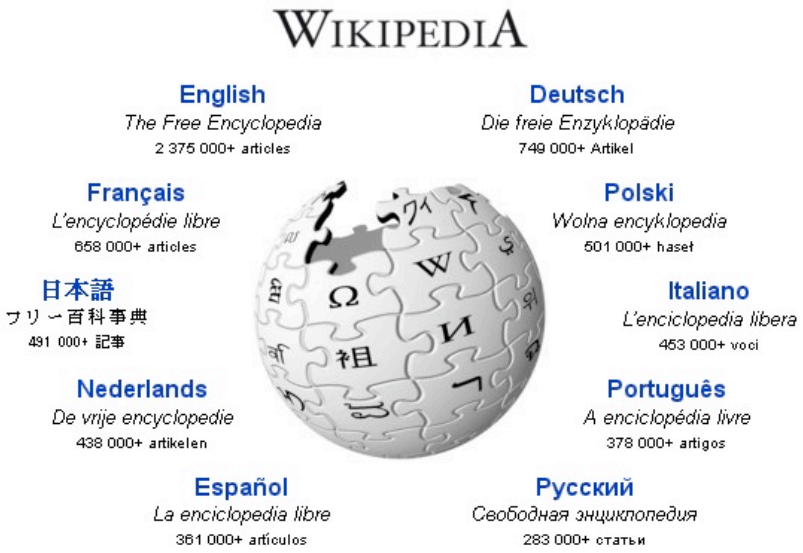


Figura 12 : http://www.wikipedia.org

Para manter a Wikipedia a funcionar, a Wikimedia tem um sistema complexo com uma arquitectura base Linux, Apache [20], MySQL [21], PHP [22] (LAMP) onde são adicionados diversos módulos: GeoDNS [23] para localização, Lucene [24] para pesquisa, Lighttpd [25] para alojar imagens e diversos mecanismos de *caching* para melhorar o desempenho da aplicação.

2.3.5 *Blogs*

Blog é sem dúvida sinónimo de Web2.0, apesar de ser um conceito que surgiu há mais de 30 anos, nos primórdios da Internet. O conceito de simples diário pessoal onde podemos partilhar experiências, histórias ou opiniões, quer sejam reais ou ficcionais, é, à partida, bastante apelativo.

Usados de início apenas para divulgação dentro de meios restritos, os *blogs* cresceram e a sua influência estendeu-se para os campos da política e entretenimento. Os *blogs* tornaram-se uma das formas mais simples de fazer passar uma mensagem em massa à juventude cada vez mais virada para as novas tecnologias.

Tabela 3 : Aplicações de *blogging*

Descrição	Endereço
Blogger: aplicação do Google	http://www.blogger.com
Wordpress: aplicação para jornalistas	http://www.wordpress.com
Movable Type: plataforma de publicação	http://www.movabletype.com

A arquitectura de armazenamento de informação de um *blog* é sem dúvida a mais simples que existe. É suficiente ter uma base de dados com uma tabela para podermos ter um sistema de *blogging* a funcionar correctamente. Claro que adicionando funcionalidades para aumentar as mais valias da aplicação a complexidade irá aumentar, mas nunca será um sistema de elevada complexidade.

Figura 13 : http://www.blogger.com

2.3.6 *Feeds*

Os mecanismos de *feeds* foram também muito impulsionados pela evolução da Web2.0. Estas novas normas para formatação de dados, como o *Really Simple Syndication* (RSS) [26], facilitam a partilha de informação, seja ela texto, áudio ou vídeo. O mecanismo é simples, o utilizador subscreve uma *feed*, de um *blog* por exemplo, e ela é actualizada sempre que o conteúdo do *blog* sofre alterações. Uma *feed* é assim uma sintetização das últimas entradas num arquivo de dados específico, como, por exemplo, um *blog* ou um arquivo áudio. Diversas aplicações, tanto *web* como *Desktop*, usam *feeds* para acederem e tratarem a informação. Exemplificando, as mais recentes aplicações Desktop, usam mecanismos de *feeds* para manter a versão instalada num computador sempre actualizada com a última versão disponível.

Figura 14 : http://multimedia.rtp.pt

A RTP disponibiliza grande parte da sua programação áudio e vídeo através de subscrição por *feeds*, o que permite que possamos acompanhar o nosso programa preferido mesmo que não o consigamos ouvir ou ver em directo.

A tecnologia de subscrição de *feeds* depende muito do formato XML (já mencionado nesta dissertação). O formato RSS e o seu principal concorrente mais recente, Atom [27], são ambos especificados num esquema XML para permitir a mais fácil partilha de informação.

2.3.7 *Media Sharing*

Aplicações de *media sharing* também apareceram com a Web2.0. De ente as diversas aplicações existentes há que destacar o YouTube.

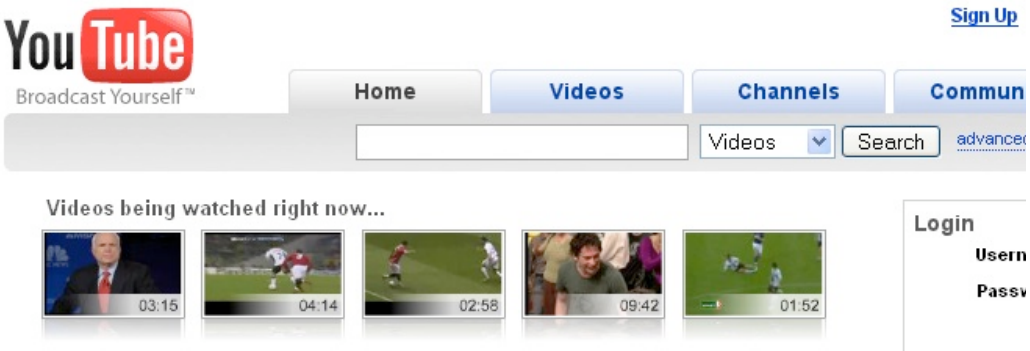


Figura 15 : http://www.youtube.com

Existem também aspectos nefastos na partilha de multimédia sendo o principal a partilha de ficheiros com direitos de autor. As aplicações de *media sharing* e as inovações que elas trouxeram em termos de partilha de multimédia – como os *web media players* – vieram aumentar a facilidade na partilha de ficheiros, favorecendo a pirataria.

Tabela 4 : Aplicações de *media sharing*

Descrição	Endereço
YouTube: o mais famoso site de partilha	http://www.youtube.com
Sapo Vídeos: ferramenta do SAPO	http://videos.sapo.pt
Twango: partilha de fotos, vídeos e áudio	http://www.twango.com

Antes da sua agregação à rede Google, o YouTube assentava numa plataforma LAMP (tal como a Wikimedia) à qual se juntava o interpretador de Python [28] e o psyco [29], para compilação dinâmica de Python para linguagem C.

2.3.8 *Web Desktops*

Outras das aplicações *web* mais interessantes que surgiram com a Web2.0 foram os *Web Desktops*. Nestas aplicações é possível configurar uma página *web* para que tenha uma interface em tudo idêntica à do nosso Sistema Operativo.

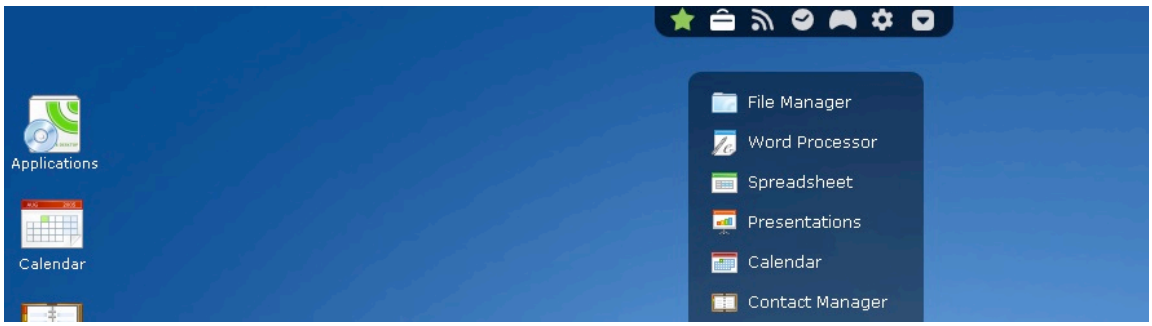


Figura 16 : http://www.eyeos.info

Estas aplicações contêm conjuntos de funcionalidades bastante completos: desde jogos a processadores de texto, isto para que consigam alcançar o objectivo de conjugar, na mesma página Web, tudo o que se pode fazer no *Desktop* de um Sistema Operativo. Devido à sua elevada compatibilidade – qualquer *browser*, qualquer Sistema Operativo – e de serem facilmente e amplamente configuráveis, estas aplicações estão numa crescente curva de popularidade.

Tabela 5 : Aplicações Web Desktop

Descrição	Endereço
EyeOS	http://www.eyeos.info
Desktop Two	https://desktoptwo.com
Netvibes	http://www.netvibes.com

2.4 Integração de dados e serviços usando fluxos de informação

A principal ideia subjacente ao trabalho descrito nesta dissertação prende-se com a criação de uma interface que proporcione um aumento da facilidade no acesso a informação e serviços fornecidos por diversas fontes. Perante este objectivo, foi necessário escolher a metáfora ideal para a integração das fontes de dados e funcionalidades dispersas.

A solução encontrada consiste em dar ao utilizador a possibilidade de construir um fluxo de informação – Workflow – que tem como entrada os dados iniciais submetidos pelo utilizador – Dataset – e devolve, no final do processamento, a informação pretendida. Como definido pela *Workflow Management Coalition*, um Workflow é a "*automatização ou simplificação computorizada do todo ou de uma só parte de um processo"* [30]. É precisamente esta definição essencial que se pretende cobrir usando um fluxo de informação em que diversos serviços são encadeados: "*Um aspecto chave no encadeamento de serviços é a possibilidade de usar directamente a saída de um serviço como entrada para outro*" [31]. Ou seja, o objectivo é proporcionar ao utilizador a oportunidade de aceder a diversos serviços fechados, heterogéneos e dispersos, de uma forma dinâmica, sequencial e personalizável. Cada um deste

serviços é representado por um *wrapper* (ou adaptador, designado na arquitectura de Módulo de Processamento), que é apenas identificado pela localização do serviço, o tipo de dados de entrada que aceita e o tipo de dados que devolve na saída. O utilizador tem a possibilidade de construir o percurso que pretende dar à informação. Esta aproximação diferencia-se dos métodos de integração de serviços estáticos, em que o *workflow* era construído programaticamente no desenvolvimento da aplicação, e não dava ao utilizador qualquer tipo de controlo sobre o mesmo.

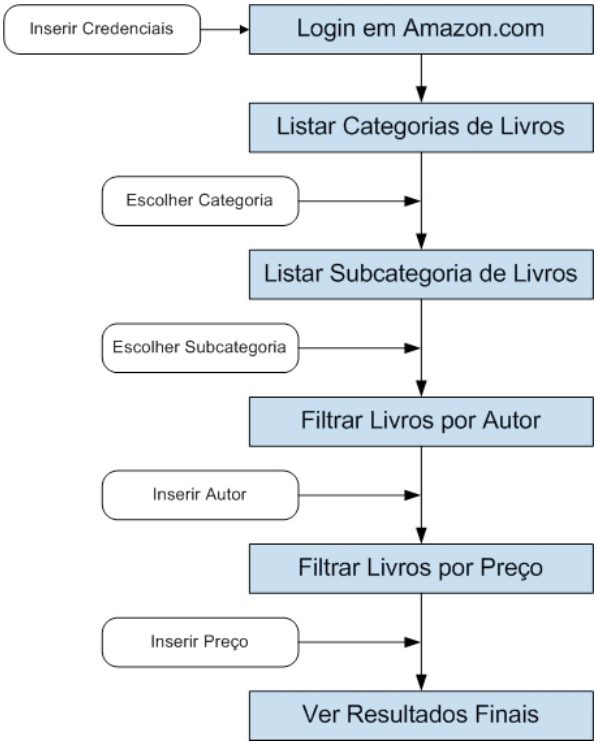


Figura 17 : Exemplo de fluxo de informação personalizado

O *workflow* pode ser composto por diversos adaptadores que representam o acesso a diferentes serviços e fontes de informação (Figura 17). Nestes módulos de processamento (Figura 18), os dados de entrada são provenientes dos dados de saída do módulo anterior. Cada um dos adaptadores corresponderá a uma funcionalidade ou fonte de informação disponibilizada por entidades externas à nossa aplicação e que estão disponíveis através de um endereço Web público.

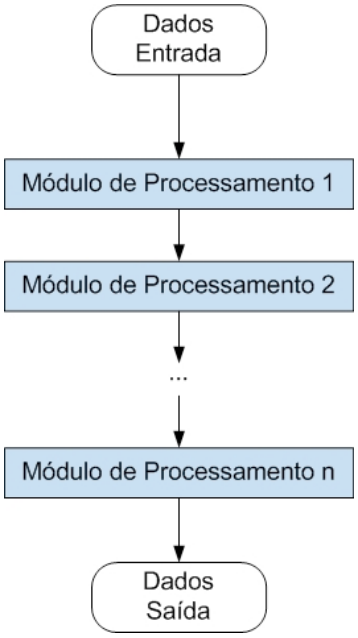


Figura 18 : Diagrama do fluxo de informação

A Tabela 6 tem uma explicação sucinta dos elementos mais importantes da aplicação. Foi sobre este método de processamento que foi desenhada e arquitectada a aplicação descrita nesta dissertação.

Tabela 6 : Descrição das componentes do sistema

Elemento	Descrição
Dataset	Dados iniciais fornecidos pelo utilizador para início do processamento do fluxo de informação. É a entrada do primeiro módulo de processamento do Workflow.
Módulo de Processamento	Função típica. Trabalha sobre os dados que recebe à entrada de modo a produzir dados de saída de acordo com o fornecido por uma entidade externa. Podem ser métodos de filtragem de informação, de procura de informação, de agregação de informação. Os tipos de dados de entrada e saída podem ser distintos. Completamente independentes da aplicação, são apenas fornecidas as configurações de entrada e saída, o que acontece dentro deles não interfere com o funcionamento da aplicação.
Workflow	Agrupamento ordenado de módulos de processamento que pode ser editado pelo utilizador de acordo de modo a obter a informação final que pretende.
Dataflow	Conjunto de trabalho: Dataset e Workflow.

Usando fluxos de informação é possível aceder a dados e funcionalidades mantidos por diversas fontes de uma forma transparente. Com este método consegue-se criar um sistema genérico: quer o utilizador, quer a aplicação em si, não

sabem o que acontece em cada passo do processamento, apenas necessitam de construir o fluxo de forma a que a coerência entre os dados seja mantida. Este aspecto é sem dúvida um ponto a favor da aplicação, pois permite que sejam tornadas públicas funcionalidades – através de *Web Services* por exemplo – que escondem tudo o que acontece do lado da entidade, salvaguardando, assim, trabalho e informação privados. Este método permite também que um problema geral de alguma complexidade, seja dividido em pequenos sub-problemas, sendo cada um deles resolvido num *wrapper* específico e através de uma estratégia de "dividir para reinar". Aliado a estes factores, surge ainda o facto de, através de um pequeno Dataset como informação de entrada, ser possível extrair e aumentar o conhecimento, com o processamento organizado dos diferentes adaptadores.

2.4.1 Aplicações Web 2.0 de Gestão de *Workflows*

Usando as novas potencialidades fomentadas pela Web2.0, surgiram online as primeiras aplicações de gestão de *workflows* que seguem princípios semelhantes ao que pretendemos alcançar.

Nesta área, é necessário dar o devido destaque a duas delas: Yahoo! Pipes e Microsoft Popfly. Ambas fornecem um ambiente gráfico *web-based* que permite construir fluxos de informação através de adaptadores pré-definidos e que suportam diversas funcionalidades.

Microsoft Popfly

A Popfly [32], criada nos laboratórios da Microsoft, fornece uma panóplia de *sites* e adaptadores pré-definidos sobre os se pode trabalhar.

Centrada numa perspectivas de trabalho sobre *feeds*, grande parte das funcionalidades oferecidas por esta aplicação lidam com o manuseamento das mesmas: existem sobretudo métodos para agregação e filtragem de informação. A maioria dos *wrappers* pré-definidos encontra-se associado a aplicações Web de diversas áreas e poucas alterações podem sofrer.

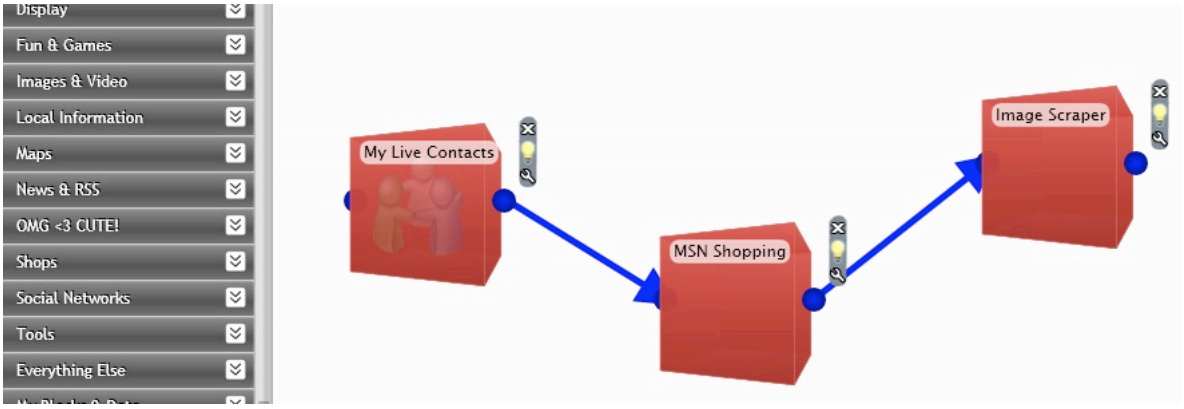


Figura 19 : http://www.popfly.com

A interface da aplicação baseia-se em cubos daptadores que se ligam através de pontos de contacto. Esta de interface muito pouco usável e para alcançá-la, a Popfly necessita que o utilizador tenha instalado o *plugin* Silverlight no seu *browser*, um ponto negativo na avaliação da aplicação.

Yahoo! Pipes

Criada pela Yahoo!, a Pipes [33] é uma aplicação bastante mais avançada que a Popfly. Os blocos de processamento são representados por caixas arrastáveis interligados através de pontos de contacto em vez dos cubos da Popfly e apresentando uma interface bastante mais limpa e intuitiva.

Ao contrário da Popfly, a Pipes destina-se a um público diferente. São necessários alguns fundamentos gerais de programação e funcionamento da Web, pois funciona aproximadamente como uma linguagem de programação: tem operadores, ciclos, estruturas condicionais e tipos de dados. Neste aspecto, também temos menos módulos de processamento já construídos e é dada ao utilizador uma total liberdade na personalização dos mesmos.

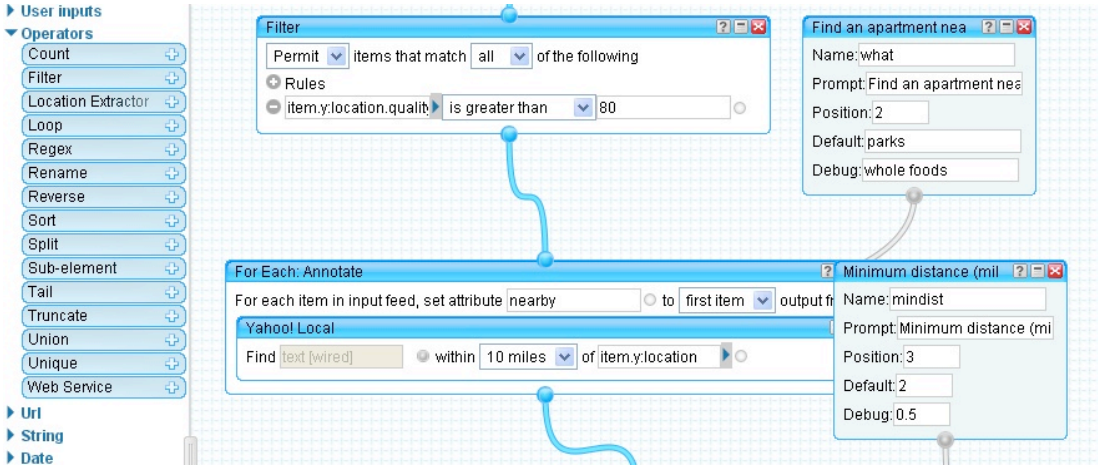


Figura 20 : http://pipes.yahoo.com

2.5 Sumário

Neste capítulo são apresentadas diversas tecnologias e formas de efectuar a integração de dados e serviços e de disponibilizar a informação obtida. Há ainda bastante trabalho no sentido de melhorar a interacção do utilizador com as bases de dados, principalmente para os utilizadores sem qualquer conhecimento técnico de como processar a informação. Um dos grandes problemas que surge nesta interacção é o facto de o utilizador estar dependente de *queries* pré-definidas para acesso aos dados. Ou seja, o conjunto de informação que pode ser extraída de um conjunto de bases de dados é, na generalidade, pré-definido e depende daquilo que os programadores e os donos das bases de dados fornecem. O utilizador final da aplicação não pode definir qual o fluxo que a informação deve levar por forma a conseguir obter os resultados pretendidos. O utilizador está sempre preso a funções, a motores de pesquisa e a *workflows* que já existem para obter a informação, fazendo com que nem sempre os resultados sejam os melhores e o utilizador tenha de proceder a uma segunda análise recorrendo a outras aplicações ou ferramentas.

3 Arquitectura Web 2.0 para integração de dados e serviços usando fluxos de informação

Como mencionado previamente, a integração de dados começa a ser cada vez mais comum em aplicações *Web*, principalmente devido às características e ideias da Web2.0.

Neste capítulo pretende-se definir uma arquitectura típica de aplicações Web2.0 que suporte a integração de dados e serviços com base em fluxos de informação personalizados.

O objectivo de permitir que o utilizador construa o seu próprio fluxo de informação e a particularidade da integração de serviços – não só de dados – levam à necessidade de efectuar um estudo rigoroso sobre a arquitectura. O sistema tem de ser escalável e flexível tanto a nível de componentes que usa como a nível de serviços e fontes de dados que irá integrar. A complexidade da arquitectura é ainda aumentada devido ao facto de se tratar de uma aplicação Web: não será instalada num computador próprio para uso particular e terá de estar sempre online, disponível a qualquer momento e a partir de qualquer local, funcionando com um desempenho aceitável.

A solução encontrada usando fluxos de informação levanta ainda diversos requisitos muito específicos e que contribuem também para o aumento da complexidade da arquitectura.

Na primeira parte temos uma descrição das diversas funcionalidades pretendidas. A esta descrição segue-se uma secção com diagramas ilustrativos da organização da aplicação em função dos requisitos existentes.

3.1 Funcionalidades

As várias funcionalidades pretendidas encontram-se organizadas em três grupos distintos: Gestão de Utilizador, Gestão do Dataset e Gestão do Workflow.

Na Tabela 7 encontra-se uma lista do conjunto de funcionalidades da aplicação.

Tabela 7 : Lista de funcionalidades

Nome	Descrição
Gestão de Utilizador	
Registar	Utilizador regista-se no sistema, para poder usufruir de todas as potencialidades do sistema. O registo envolve o fornecimento de alguns dados e a criação de um *username* com uma *password* associada.
Login	Utilizador entra no sistema, pode aceder a todas as funcionalidades da aplicação, nomeadamente às que envolvem um histórico.
Logout	Utilizador sai do sistema.
Alterar Password	Utilizador pode alterar a sua *password* de acesso ao sistema.
Gestão de Dataset	
Novo Dataset	Utilizador insere os dados necessários para a construção de um novo Dataset.
Ver Dataset	Utilizador visualiza a informação que tem num determinado Dataset.
Editar Dataset	Utilizador edita a informação de um determinado Dataset.
Apagar Dataset	Utilizador remove um Dataset previamente guardado. Apenas para utilizador registados.
Abrir Dataset	Utilizador abre um Dataset previamente guardado. Apenas para utilizadores registados.
Ver Todos Datasets	Utilizador vê todos os Datasets que possui. Apenas para utilizadores registados.
Gestão de Workflow	
Novo Workflow	Utilizador inicia a construção de um novo Workflow.
Editar Workflow	Utilizador altera os dados de um Workflow.
Apagar Workflow	Utilizador apaga um Workflow previamente guardado. Apenas para utilizadores registados.
Abrir Workflow	Utilizador abre um Workflow previamente guardado. Apenas para utilizadores registados.
Ver Todos Workflows	Utilizador vê todos os Workflows que possui. Apenas para utilizadores registados.
Adicionar Módulo de Processamento	Utilizador adiciona módulo de processamento à posição pretendida no Workflow.
Remover Módulo de Processamento	Utilizador remove o módulo de processamento que pretende do Workflow.
Alterar Ordem de Módulo de Pocessamento	Utilizador altera a ordem do módulo de processamento pretendido no Workflow.
Partilhar Workflow	Utilizador torna o seu Workflow público para que outros utilizadores possam acedê-lo.
Clonar Workflow	Utilizador clona o Workflow pretendido.

3.2 Requisitos Técnicos

Além das funcionalidades necessárias pelo utilizador, uma aplicação de integração de dados e serviços destinada a ter as características que desejamos, necessita de cumprir diversos requisitos de nível técnico. É a partir destes requisitos que será estruturado o sistema de suporte à aplicação e às funcionalidades da mesma.

3.2.1 Generalidade de serviços

Um dos principais requisitos da arquitectura da aplicação é a sua generalidade e, aliada a esta, a sua flexibilidade e escalabilidade.

Para alcançar a generalidade de serviços é necessário que os adaptadores sejam completamente independentes entre si e funcionem de acordo com uma norma pré-definida. Desde que haja respeito pela norma, o sistema terá de suportar um *workflow* com um qualquer número e ordem de *wrappers*.

Isto implica que a implementação de cada módulo tenha de ser independente da aplicação, ou seja, tudo o que acontece dentro do módulo não terá interferência da aplicação, fornecendo esta apenas os dados de entrada e recebendo os dados de saída.

3.2.2 Processamento do lado do cliente

Apesar dos *wrappers* necessitarem de efectuar o acesso a um servidor para obtenção da informação, o processamento do fluxo de informação – transferência dos dados da saída de um adaptador para a entrada do adaptador seguinte – terá de ser feito do lado do cliente.

Este requisito implica que o tratamento da informação e a sua canalização dentro do *workflow* seja tratado no *browser* do utilizador. Este método permite poupar carga desnecessária no servidor libertando-o para um melhor fornecimento das páginas HTML ao cliente.

3.2.3 Integração dos serviços de utilizador numa plataforma existente

O utilizador (e seus respectivos dados) terão de estar dependentes de um serviço centralizado de autenticação independente desta aplicação.

Este requisito implica que os métodos para tratar a área de Gestão de Utilizador terão de trabalhar com APIs fornecidas por um sistema externo. Este acesso ao serviço de autenticação pode ser efectuado de diversas formas: acesso directo à base de dados, API programada ou *Web Services*. No entanto, é necessário que a aplicação não seja dependente do tipo de acesso aos dados de utilizador escolhido, por forma a facilitar a migração para outro serviço de utilizadores.

3.3 Modelo e Arquitectura

3.3.1 Componentes

Numa aplicação Web, mais que nas tradicionais aplicações Desktop, os recursos necessários para o bom funcionamento da aplicação são variados. Tendo em conta que a aplicação precisa de ser escalável, flexível e estar online 24h por dia, a utilização de diversas componentes para que o serviço seja mais completo, torna-se essencial.

Servidor Web

O elemento mais importante da aplicação é sem dúvida o servidor Web. É este recurso que irá receber os pedidos do cliente e tratá-los de acordo com o programado, por forma a fornecer ao utilizador todas as funcionalidades da aplicação.

É também no servidor Web que estarão alojados os componentes programáticos da aplicação: a ligação entre as diversas APIs, as classes que mapeiam a base de dados e o código necessário para o pré-processamento das páginas.

Sistema de Gestão de Bases de Dados

A informação a armazenar por uma aplicação de integração de dados é sempre bastante elevada. Juntando a esta os dados de suporte necessários às diversas funcionalidades de gestão de Datasets e Workflows temos uma quantidade considerável de dados a armazenar.

Uma base de dados seguindo um tradicional modelo relacional é a melhor opção para guardar os diversos dados e relações entre eles.

Servidor de Utilizadores

Tal como foi referido anteriormente, a Gestão de Utilizadores faz parte de um sistema independente da aplicação onde esta terá de se integrar de forma transparente.

Não sendo da competência da nossa aplicação manter este componente, é necessário que esteja sempre operacional por forma a não comprometer o funcionamento global da nossa aplicação.

Módulos de Processamento

Tal como o Servidor de Utilizadores, o funcionamento dos *wrappers* também terá de ser independente da aplicação.

Estes módulos devem ter uma funcionalidade fechada e, de forma a serem inter-operáveis entre si, seguir uma norma pré-definida.

3.3.2 Modelo

O modelo de suporte à aplicação revela-se simples e objectivo. Mesmo tendo um extenso número de requisitos a cumprir e recursos a usar, o modelo traduz-se em 3 camadas distintas (Figura 21).

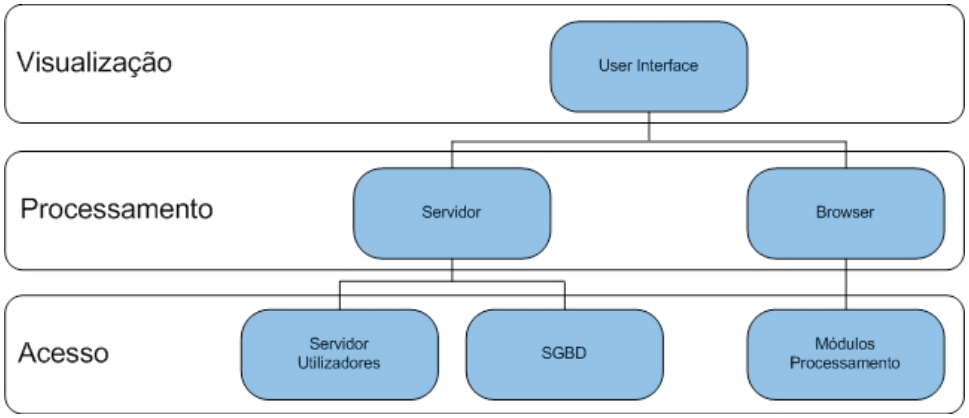


Figura 21 : Modelo da aplicação

Começando pelo nível inferior, temos a primeira camada, de acesso aos adaptadores, ao Serviço de Utilizadores e ao SGBD da aplicação.

Acima desta temos a camada de processamento. Nesta camada temos o processamento do lado do servidor, onde é tratado o conteúdo das páginas a mostrar ao utilizador e o processamento no cliente, onde é tratado o fluxo de informação no *workflow* construído pelo utilizador e a visualização da resposta a algumas operações.

No topo, temos a camada do cliente, onde se encontra o *browser* que o utilizador usa para aceder à aplicação.

3.3.3 Arquitectura

A arquitectura da aplicação é igualmente simples. A Figura 22 pretende mostrar uma visão da arquitectura do sistema.

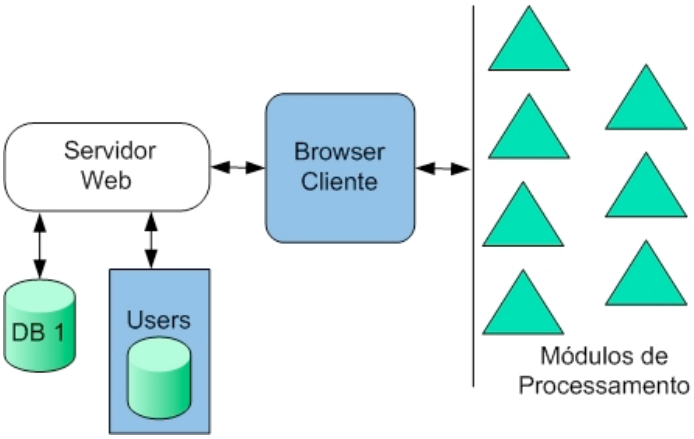


Figura 22 : Arquitectura da aplicação

Sucintamente, o funcionamento da aplicação subdivide-se em três categorias. Acessos do cliente ao Servidor Web: quando o utilizador abre o site e utiliza a aplicação, os pedidos são efectuados directamente ao Servidor Web e tratados por este. Acessos do Servidor Web aos serviços externos: caso seja necessário para a realização de uma determinada funcionalidade o Servidor Web tem de aceder ao SGBD (Gestão de Datasets e Workflows) ou ao Servidor de Utilizadores (Gestão de Utilizadores). A última categoria, inclui as interacções entre o *browser* do cliente e os módulos de processamento, isto durante o processamento do fluxo de informação.

3.4 Sumário

Este capítulo apresenta uma primeira análise do sistema. São mostradas as funcionalidades da aplicação, o recursos que terá de usar e os requisitos que terá de cumprir.

No final é apresentada uma proposta de arquitectura para cobrir da melhor forma todas as áreas apresentadas.

4 DynamicFlow

Actualmente, os grandes problemas da informação disponível na área da biologia são: a grande quantidade de dados e a heterogeneidade das ferramentas que tratam e disponibilizam os mesmos [34]. A heterogeneidade – de dados e ferramentas – é a palavra chave na contextualização aqui apresentada.

A heterogeneidade de dados, distribuídos por diversas fontes e armazenados em múltiplos formatos, fomentam a integração de dados de modo a conseguir obter mais e melhor informação, da forma mais simples e directa possível.

A heterogeneidade de serviços, ou seja, funcionalidades distribuídas por diversas aplicações e usando dados e métodos de execução completamente distintos, fomentam o uso de fluxos de informação organizados em módulos de processamento, cada um deles com um serviço autónomo e não relacionado com os outros.

Para melhor perceber o problema partimos de uma questão biológica específica: "*Um geneticista localizou um factor de obesidade numa região até 5-Mb do cromossoma humano. Existem genes nesta região que tenham homólogos que estejam envolvidos na regulação do metabolismo de lípidos em qualquer sistema modelo?*" [35].

Responder a esta questão não é tão simples quanto possa parecer à primeira vista. Uma primeira solução passaria por realizar todo o trabalho de pesquisa "à mão", percorrendo as imensas bases de dados existentes e realizando as operações necessárias até obtermos a nossa resposta. Alternativamente, poder-se-ia programar uma aplicação que respondesse a esta pergunta, fornecendo-lhe tudo o que seria necessário para chegar a uma resposta. Esta alternativa parece válida, pelo menos até aparecer outra questão, tornando necessário programar outra aplicação.

Analisando o processo de resolução do problema, concluímos que é necessário dividir o problema em sub-problemas mais pequenos e mais simples de resolver, sendo que, para cada um destes sub-problemas temos de pesquisar numa fonte de dados ou usar um serviço disponibilizado online por alguma entidade. Tudo isto, seguindo uma determinada ordem, como se de uma equação matemática se tratasse.

O ideal seria ter uma aplicação que permitisse responder a todas as perguntas semelhantes à citada, em que, introduzida a pergunta, o processo fosse completamente autónomo até à obtenção da resposta. No entanto, a tecnologia disponível ainda não é tão avançada para permitir uma aplicação deste género. O que é exequível, e este é o principal desafio lançado para a construção da aplicação DynamicFlow, é uma aplicação que permita que o utilizador resolva os seus problemas de forma modular, construindo um fluxo de informação personalizado para cada questão e que possibilite, dentro do mesmo fluxo de informação, o acesso a serviços distintos e a fontes de dados distintas.

Assim, no desenvolvimento da aplicação tentou-se proporcionar ao utilizador uma estratégia de resolução de problemas "*dividir para conquistar*", com integração de informação e serviços para expansão de conhecimento, seguindo linhas Web2.0.

4.1 Estratégias de Desenvolvimento

De forma a conseguir satisfazer todos os requisitos foi necessário ser bastante criterioso nas escolhas dos componentes a utilizar. Tal como já foi referido, é necessário ter em conta que a aplicação tem de estar disponível 24 horas por dia e com um desempenho aceitável para o máximo número de utilizadores possível. Atingir esta meta implica escolher as melhores tecnologias disponíveis. Não existiu uma preocupação pelo uso de *software* livre apesar do código final da aplicação ser disponibilizado sem qualquer restrição.

Mesmo antes de começar a programar, a aplicação teve de ser planificada, tendo-se escolhido as tecnologias que seriam usadas para responder aos objectivos propostos, cumprindo os requisitos traçados. As escolhas foram feitas para que a aplicação se integrasse da melhor forma possível com o conjunto de aplicações do grupo já existentes, simplificando a selecção de alguns dos componentes.

Servidor Web

Ao considerar que Servidor Web deveria ser escolhido, as opções são: um servidor livre – Apache (dentro de qualquer sistema operativo) ou um servidor comercial – Windows Server [36].

Atendendo ao facto de as aplicações realizadas dentro do grupo correrem de momento num servidor Windows Server 2003 e a linguagem de pré-processamento usada ser uma combinação de ASP.NET [37] com C# - linguagem orientada a objectos, um conceito que se tentou utilizar desde o início da planificação do sistema – optou-se por manter a coerência e seguir a estratégia de implementação de aplicações existente no grupo. Com esta escolha, o suporte ao trabalho dentro do próprio grupo encontra-se facilitado pois tratam-se de áreas onde existe experiência.

Outra combinação possível seria usar um servidor Apache (ou qualquer uma das suas ramificações) e apostar numa arquitectura baseada na linguagem Java. Em termos globais, os dois caminhos que podiam ter sido seguidos são semelhantes e dariam os mesmo resultados para o utilizador.

Para a componente final de contacto com o cliente temos o formato típico das páginas Web, o HTML, que pode ser pós-processado através de Javascript.

Sistema de Gestão de Base de Dados

Na escolha do SGBD a usar, as opções são variadas. Optando pela aproximação relacional no SGBD, foram consideradas as seguintes hipóteses: MySQL, PostgreSQL, Oracle [38] e Microsoft SQL Server [39].

A escolha recaiu mais uma vez sobre a opção mais coerente com as práticas do grupo: o Microsoft SQL Server. Este SGBD tem um grande desempenho e escalabilidade. Como já é o usado pelas restantes aplicações do grupo, o processo de configuração foi simples.

Não entrando em discussão retórica sobre qual o melhor SGBD, o desempenho do Microsoft SQL Server é público e pesou na decisão. A opção Oracle foi descartada à partida devido ao facto de envolver uma licença de utilização que o grupo não tem. As outras opções, MySQL e PostgreSQL foram descartadas devido ao facto de não serem usadas nos servidores do grupo.

No entanto, devido à arquitectura da aplicação, o SGBD é completamente independente da restante aplicação. Isto significa que pode ser alterado em qualquer altura, bastando uma simples reconfiguração da camada de acesso ao SGBD, para a aplicação continuar completamente funcional.

Servidor de Utilizadores

Sabendo de antemão que o grupo pretende centralizar num serviço único as áreas de utilizadores das suas aplicações e que usa nestas a *Framework* .NET [40] da Microsoft, a escolha para o serviço de utilizadores recaiu sobre o serviço de autenticação fornecido por esta arquitectura (Figura 22).

Este serviço providencia grande parte das funcionalidades necessárias para o manutenção de utilizadores e, usando-o, cumpre-se um dos requisitos traçados à partida, relacionado com o facto de o serviço de utilizadores ter de ser independente de aplicação e este poder ser trocado com relativa facilidade.

Uma descrição completa da arquitectura de autenticação (Figura 23) usada não é do âmbito desta dissertação, importa no entanto referir os principais pontos positivos da arquitectura escolhida.

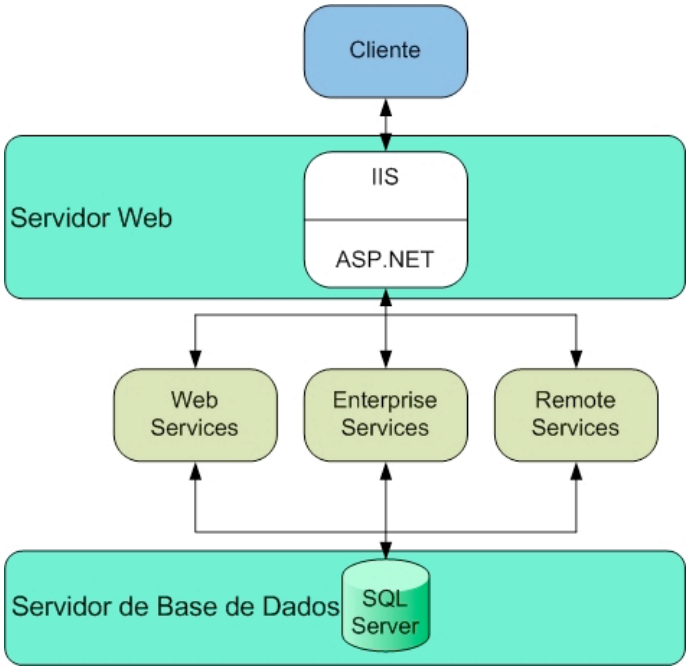


Figura 23 : Arquitectura de autenticação da *Framework* .NET

Esta arquitectura implementa diversos serviços: além do tradicional serviço de *Membership* de utilizadores existe um serviço de *Roles* para definição de autorizações dentro das aplicações e podemos ter também distinções entre as diversas aplicações do grupo. Esta arquitectura é bastante flexível e traz a grande vantagem de ser de fácil utilização dentro do esquema de aplicações que estamos a usar: usando ASP.NET e Microsoft SQL Server, todos os métodos necessários para a Gestão de Utilizadores, de *Memberships* e de *Roles*, encontram-se implementados e bem documentados simplificando a tarefa de construir uma arquitectura de autenticação centralizada de larga escala.

Mais uma vez, o acesso a este serviço de autenticação centralizado também se encontra dependente de apenas uma camada da aplicação, o que facilita o processo de migração para outro esquema de autenticação em caso de necessidade.

Módulos de Processamento

Como previsto no início do projecto, os componentes mais difíceis de obter foram os adaptadores. Esta dificuldade prende-se sobretudo com o facto de terem de respeitar uma especificação criada de raiz de propósito para esta aplicação. Não obstante esta dificuldade, uma aplicação de integração de dados do grupo de Bioinformática – BioPortal – foi modificada por forma a providenciar *wrappers* com tarefas individuais.

O BioPortal é uma plataforma de integração de dados biológicos heterogéneos, tanto a nível de formato dos dados como a nível das classes de dados armazenados. No BioPortal tenta-se aproveitar as melhores características dos métodos de integração de dados existentes. Assim, são combinadas as vantagens do *data warehousing*, do uso de mediadores e da integração de ligações. O processo de inserção de novas fontes de dados, usando de qualquer um dos métodos, foi simplificado e normalizado. Esta aproximação híbrida visa melhorar o desempenho e sobretudo a escalabilidade, no acesso a dados biológicos dispersos. O BioPortal já é usado numa aplicação bioinformática, o Genebrowser [3], e que serviu de base ao presente trabalho.

Usando o BioPortal cada módulo de processamento corresponde a um pedido a um endereço Web específico, fornecendo na *query string* do pedido os argumentos de entrada.

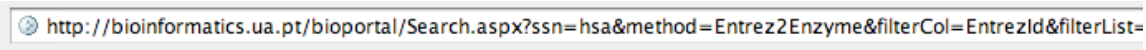


Figura 24 : Exemplo de *query string* de acesso ao BioPortal

A *query string* possui argumentos bastantes específicos através dos quais se diferenciam os diversos adaptadores. A Tabela 8 mostra os argumentos disponíveis e a sua funcionalidade.

Tabela 8 : Especificação dos argumentos de entrada do BioPortal

Argumento	Descrição
ssn	Espécie. Define a que espécie pertence a informação genética contida no Dataset para que os dados possam ser correctamente processados.
retType	Formato de dados de retorno. Define qual o formato em que os dados serão devolvidos, podem ser devolvidos em formato XML ou JSON.
objId	Identificador do objecto. Usado quando o formato de retorno dos dados é JSON para identificar o objecto devolvido contendo a informação.
method	Método. O método do BioPortal que servirá para efectuar a operação que pretendemos.
filterList	Lista de elementos. Informação de entrada no módulo de processamento.
filterCol	Tipo de elementos. Define qual o tipo de elementos de entrada contidos na filterList.
limit	Limite de saída. Define o número máximo de elementos de saída que são devolvidos pelo adaptador.
offset	Define o offset aplicado ao conjunto de dados de saída.
comp	Compressão de dados. Define se a compressão de dados na saída está activa ou inactiva.

Sabendo esta informação, o passo seguinte passou por fornecer à aplicação as características de cada módulo de processamento. Sabendo que elementos fixos compõem cada um destes *wrappers*, podemos torná-los parte do fluxo de informação ajustando as variáveis – os dados de entrada (filterList) por exemplo – de acordo com a informação do *workflow*.

O requisito essencial dos adaptadores terem de ser independentes entre si e fornecer a sua funcionalidade de forma fechada restringiu as implementações possíveis. Acrescentando à lista de requisitos existentes o facto do processamento do *workflow* ter de ser feito no *browser* do cliente, as escolhas reduzem-se a métodos Web. Nestes métodos temos a solução escolhida, um pedido HTTP composto por diversos argumentos ou então efectuar pedidos através de *Web Services*. A utilização destes a partir de Javascript revelou-se um processo complexo, pouco fiável e dependente de bibliotecas externas. Por seu lado, os pedidos HTTP são directos e dão aos programadores a liberdade de programar os módulos das mais variadas maneiras, quer em termos de linguagem de programação, quer em termos de argumentos de entrada e construção da *query string*.

4.2 Interface de utilizador

A interface da aplicação é o principal ponto de contacto entre o utilizador e a aplicação. É o principal elemento na complexa arquitectura pois é o que os utilizadores vêm e onde os utilizadores trabalham.

O processo de criação de uma interface Web2.0 que possibilite o respeito de todos os requisitos analisados, e alie a funcionalidade à usabilidade, foi um processo moroso. Diversas opções foram tomadas de modo a proporcionar a melhor experiência de utilização possível. A principal destas opções envolvia a escolha da tecnologia a usar na interface principal que tinha de ter em conta este ideal Web2.0, o conjunto de funcionalidades que iriam ser oferecidas ao utilizador e a dinâmica final da aplicação. Foram consideradas três tecnologias: AJAX, Flash e Silverlight.

Nas palavras da empresa Adobe, o Flash é "*o* plugin *de cliente de alto desempenho, leve e fortemente expressivo que fornece poderosas e consistentes experiências de utilizador em qualquer sistema operativo,* browser*, telemóvel ou outro dispositivo*" [41]. De facto, a tecnologia Flash revela-se muito apetecível para a criação de interfaces dinâmicas e interactivas para aplicações Web. No entanto, a criação das interfaces é complexa e a integração com a arquitectura em utilização pouco natural.

"*Microsoft Silverlight é um* plugin *para qualquer* browser*, qualquer plataforma e qualquer dispositivo para fornecer a próxima geração de experiências multimédia e aplicações para a Web interactivamente ricas, baseadas em .NET*" [42]. Tal como a Adobe, a Microsoft publicita as mesmas características na sua tecnologia concorrente ao Flash. Contudo, a tecnologia Silverlight da Microsoft é ainda muito recente e dá de momento os primeiros passos como alternativa na construção de interfaces dinâmicas. Apesar do grande potencial e da fácil integração com a arquitectura .NET usada na aplicação, o facto de ainda estar instável e em desenvolvimento são factores contra bastante pesados na análise desta tecnologia.

As tecnologias Flash e Silverlight têm um grande potencial e são, sem sombra de dúvida, as mais usadas actualmente nas novas aplicações Web que envolvem um grande nível de interacção com o utilizador. Têm no entanto de ser descartadas para a interface do DynamicFlow devido ao facto de necessitarem que o utilizador final instale um *plugin* no seu computador para que funcionem. Este facto é sem dúvida o mais importante na altura de escolher que tecnologia usar na interface. Isto porque a aplicação é desenvolvida num ambiente científico e tem como utilizadores finais elementos de instituições ou empresas que geralmente não podem instalar *software* no computador que usam para efectuar o seu trabalho.

Resta o já mencionado "aglomerado" de tecnologias AJAX, que serve de motor a quase todas as aplicações Web2.0 e que, apesar da sua complexidade, ultrapassa os principais problemas existentes na utilização de Silverlight ou Flash.

É importante também mencionar que a interface encontra-se sempre num constante processo de melhoramentos, quer através de sugestões dos utilizadores, quer através do melhoramento da interacção com o utilizador na execução de algumas funcionalidades.

De seguida são apresentadas as componentes mais importantes da interface. São consideradas como sendo as mais importantes, pois são aquelas que serão usadas mais vezes e terão mais influência durante o trabalho do utilizador na aplicação.

Área de trabalho

Como se trata de uma aplicação Web completa e não uma simples *website*, há que dar a devida importância à principal área de trabalho do utilizador final.

É na área de trabalho que é construído o Workflow, o principal elemento da aplicação. A construção do Workflow é um conjunto de operações que se pretende dinâmico, rápido e fácil. No entanto, é também necessário mostrar ao utilizador diversas áreas: caso o adaptador seja um filtro, é preciso reservar uma área para o utilizador inserir as características do filtro, e é necessário reservar uma área para uma pequena descrição do *wrapper*, caso o utilizador tenha dúvidas quanto à sua funcionalidade.

A área de trabalho encontra-se dividida em duas zonas principais: a zona dos módulos de processamento e a zona do Workflow.

A zona dos módulos de processamento contém uma lista de todos os adaptadores disponíveis, organizados segundo uma terminologia coerente com o âmbito da bioinformática, onde a aplicação se inclui.

Na zona do Workflow, existem 3 áreas distintas. Do lado esquerdo temos a área de entradas de utilizador, onde são mostradas as opções que envolvem uma escolha do utilizador para continuar o fluxo de informação. Na zona central temos o Workflow propriamente dito, uma lista de blocos, cada um deles correspondente a um adaptador e com informação visual para realização de diversas operações. No lado direito temos a área de descrição do módulo de processamento, sector informativo onde o utilizador pode saber tudo o que necessita sobre determinado adaptador.

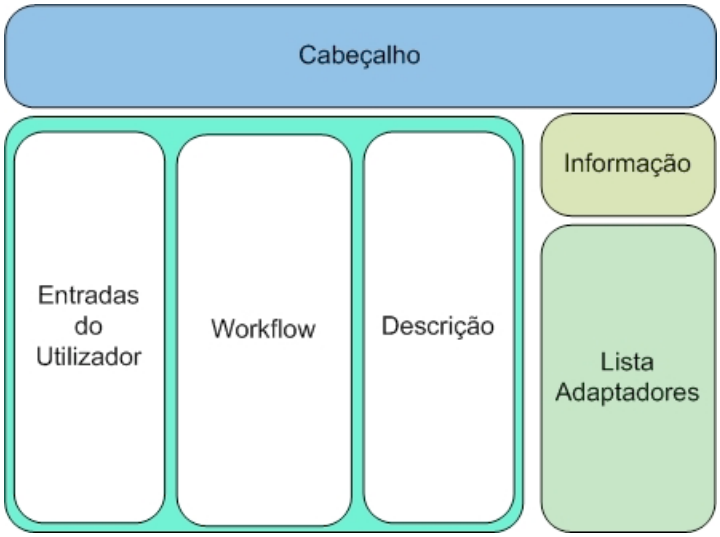


Figura 25 : Esquema da área de trabalho

Menus

Sendo a aproximação das aplicações Web às aplicações Desktop tradicionais, uma das ideias subjacentes à Web2.0, o conceito de menu foi adoptado e ajustado às necessidades da aplicação.

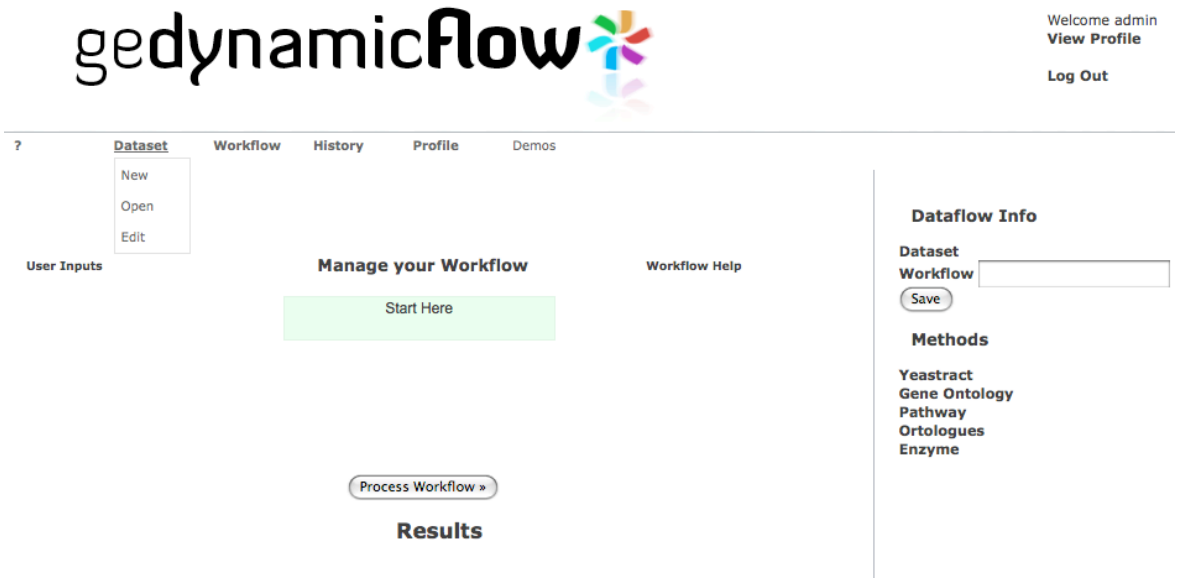


Figura 26 : Interface do Menu da aplicação

O utilizador pode, na sua área de trabalho, aceder às funcionalidades referentes ao Dataset e ao Workflow navegando para o respectivo menu, tal como faria numa aplicação Desktop. A figura 26 exemplifica a navegação para o opção Workflow do menu, onde são mostradas as várias funcionalidades disponíveis relacionadas com a Gestão do Workflow. A composição do menu varia de acordo com a página em visita e de acordo com o tipo de utilizador a navegar na página.

O recurso aos menus, permite que o utilizador se sinta mais familiarizado com a aplicação e que a aprendizagem das capacidades da aplicação se realize mais depressa.

Call-outs

Durante a utilização da aplicação DynamicFlow existe, naturalmente, uma grande quantidade de informação que terá de ser apresentada ao utilizador.

A realização de operações típicas como Abrir Dataset, Editar Dataset ou Abrir Workflow necessita de chamar a atenção do utilizador, mas isto sem que ele perca noção do local onde se encontra na aplicação, do trabalho que está a fazer e sem quebrar a experiência de utilização com elevados tempos de espera e páginas em branco. Para conseguir alcançar estes objectivos são usados *call-outs*: painéis que se sobrepõem suavemente sobre a página mostrada no *browser* e que podem conter toda a informação necessária à realização de alguma funcionalidade.

A Figura 27 mostra o funcionamento de um *call-out* na nossa aplicação. Ao seguir uma ligação que executa uma funcionalidade que necessite de mostrar informação ao utilizador, a área de trabalho é sombreada e surge do topo da página uma nova área focada com uma nova página a mostrar ao utilizador. Ao realizar alguma operação que feche o *call-out*, o utilizador regressa à página onde se encontrava previamente, com a mesma transição suave de aparecimento do *call-out*.

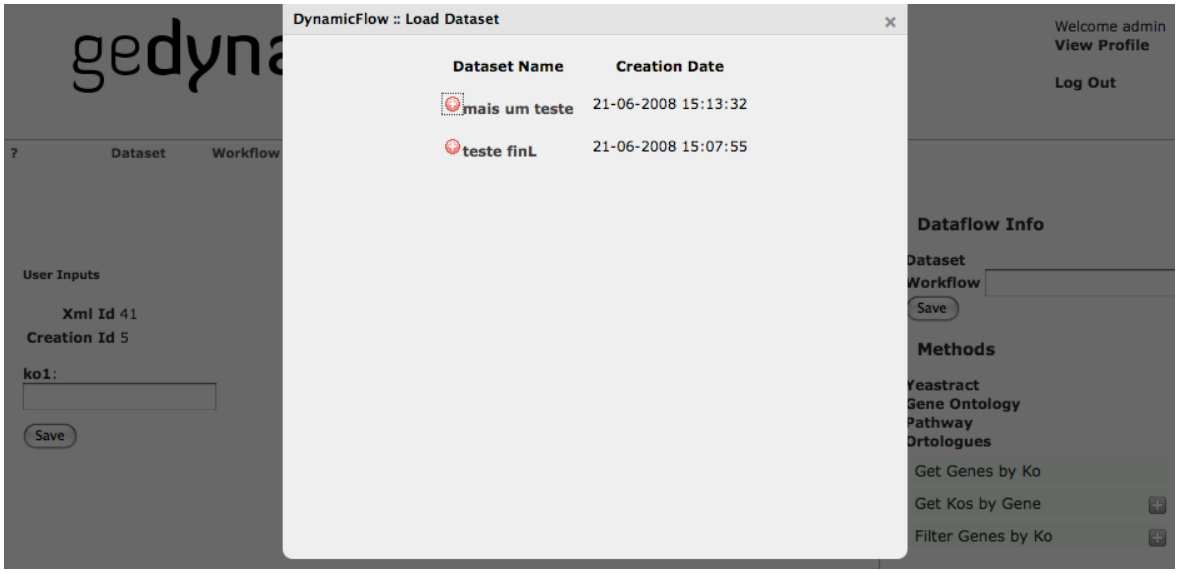


Figura 27 : Exemplo de um *call-out* na aplicação

Lista de módulos de processamento

Uma das principais zonas da área de trabalho é a que contem a listagem dos adaptadores disponíveis.

Esta lista é carregada dinamicamente a partir do ficheiro XML que guarda a configuração dos módulos de processamento. Para criar esta componente da interface, é utilizada uma folha de transformação XSL (Figura 28), que irá ler o ficheiro de configuração XML e criar a interface da listagem completa automaticamente.

```
<xsl:output method="xml" indent="yes"/>

<xsl:template match="@* | node()">
  <xsl:for-each select="//DataType">
    <a href="javascript:Slide.toggleSlide('{@id}');"><xsl:value
    <div id="{@id}" style="display:none; overflow:hidden;opacit
      <ul id="{TypeName}" class="sortable boxier">
        <xsl:for-each select="Elements/Element">
          <xsl:apply-templates select="."></xsl:apply-templates
        </xsl:for-each>
      </ul>
    </div>
    <br />
    <script type="text/javascript">
      <![CDATA[

        var list = document.getElementById("]]><xsl:value-of se
        DragDrop.makeListContainer( list, false );
        list.onDragOver = function() { this.style["border"] = "
        list.onDragOut = function() {this.style["border"] = "1p
      ]]>
    </script>
  </xsl:for-each>
</xsl:template>

<xsl:template match="Element">
  <li xmlid="{@xmlid}" urlstring="{UrlString}" objectstring="{Obj
    <xsl:value-of select="DisplayName"/>
```

Figura 28 : Extracto do ficheiro XSL para transformação dos módulos de processamento

Como já foi mencionado, um dos aspectos a ter em conta na criação da interface, foi a elevada quantidade de informação que é mostrada ao utilizador. Por forma a diminuir esta quantidade, e sabendo que os diversos *wrappers* se encontram agrupados por categorias, optou-se por usar um mecanismo que, inicialmente, apenas mostra o nome das categorias existentes. O utilizador pode depois, expandir com um clique cada uma das categorias, para que os módulos de processamento pertencentes à categoria seleccionada sejam mostrados. Se desejar, o utilizador pode depois voltar a esconder a lista de adaptadores, clicando de novo na categoria (Figura 29).

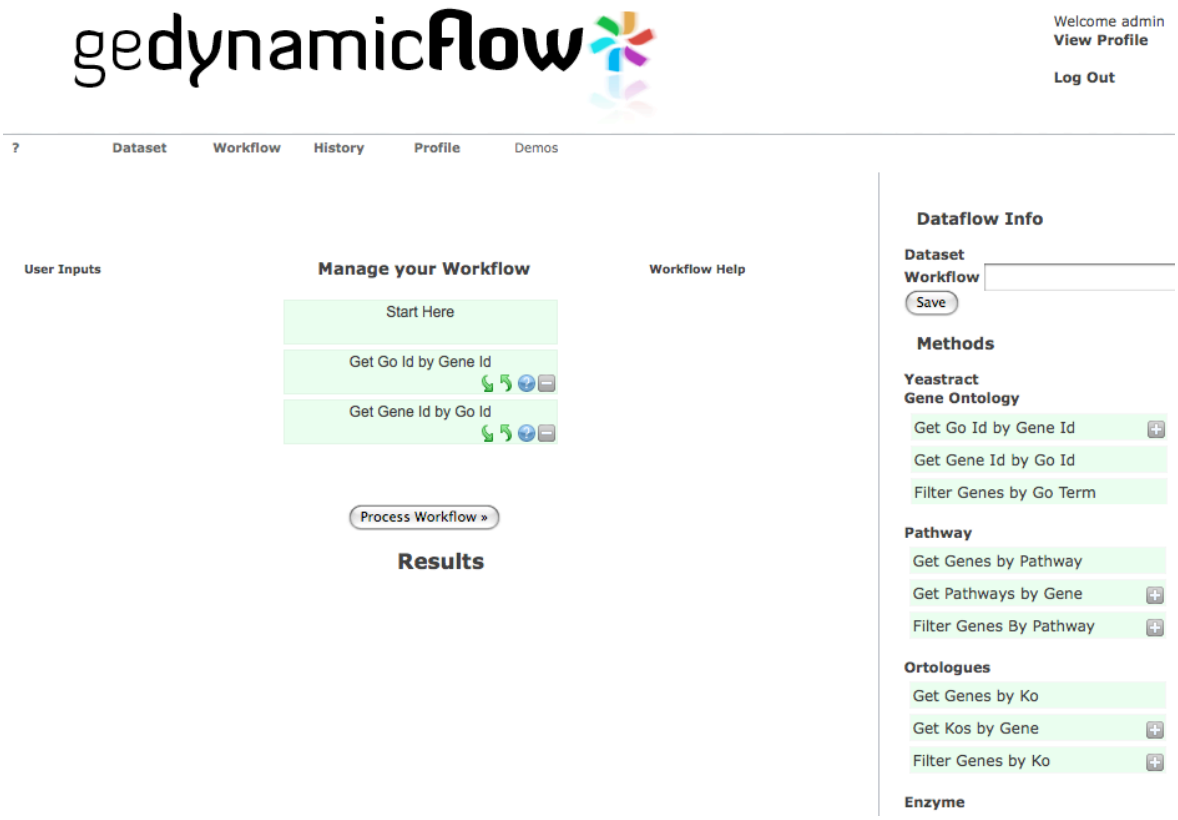


Figura 29: Interface da lista de módulos expandida

Figura 30 : Interface de lista de módulos por defeito

Operações sobre o Workflow

A principal característica desta aplicação é, sem dúvida, a construção de um fluxo de informação. Tendo em mente este aspecto, a interface para manuseamento do Workflow teve de ser bastante trabalhada.

Para construir um Workflow é necessário Adicionar Módulos de Processamento, Remover Módulos de Processamento ou Alterar a Ordem dos Módulos de Processamento. Por forma a melhorar a usabilidade da aplicação, foram criadas duas formas de realizar estas operações: através de um clique ou através de *drag-n-drop*.

A operação de *drag-n-drop* traduz-se numa melhor metáfora de utilização, igual à usada nos Sistemas Operativos e aplicações Desktop. A sequência clicar, arrastar, largar é familiar para o utilizador e, apesar de menos intuitiva nas aplicações Web, revela-se muito mais funcional pois permite colocar o adaptador na posição exacta.

Contudo, o clique é a operação mais efectuada nas páginas Web. Assim, através de símbolos representativos das acções a efectuar, é possível efectuar todas as operações possíveis sobre os *wrappers*.

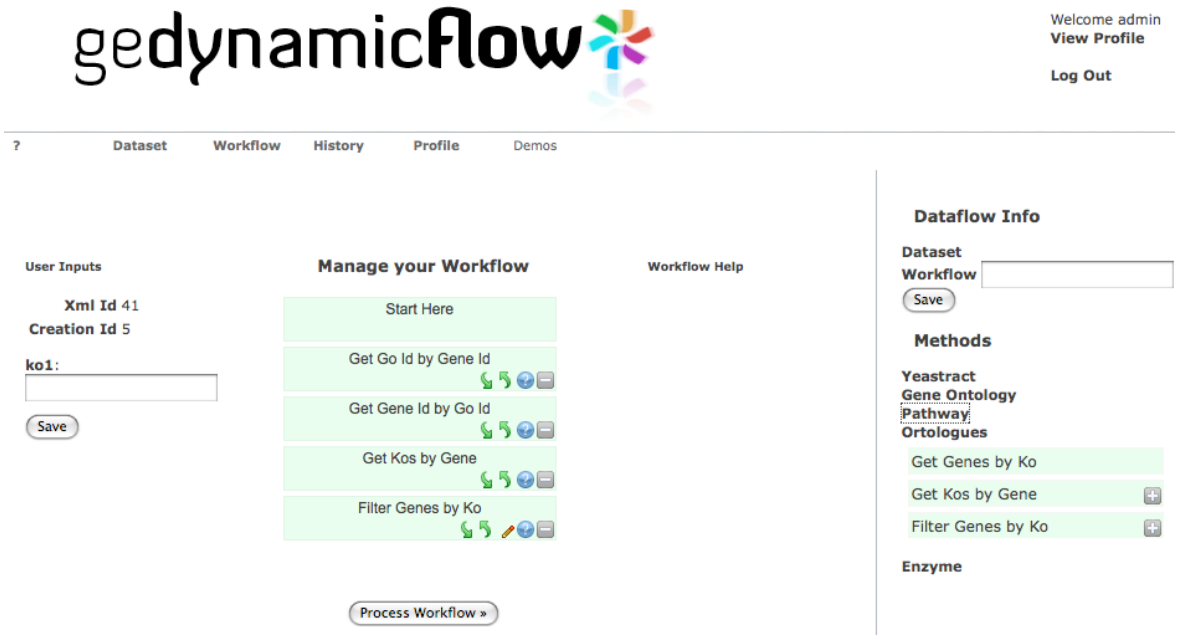


Figura 31 : Exemplo da interface dos módulo de processamento na aplicação no *workflow*

A Figura 31 mostra um exemplo de módulos de processamento, dentro do Workflow, na interface da área de trabalho. Cada bloco é composto pelo nome do adaptador, botões para passar para a posição anterior ou seguinte, botão para ver a descrição do adaptador, botão para remover o adaptador do Workflow e, caso seja um adaptador que necessite de entradas do utilizador, um botão para visualizar e editar estas entradas.

Informação da área de trabalho

Cada utilizador pode guardar uma multiplicidade de Datasets, Workflows e combinações entre Datasets e Workflows. Para que o utilizador saiba sempre com que Workflow e Dataset está a trabalhar, foi incluída na área de trabalho uma zona com informação relativa ao conjunto actual.

Esta área de informação contém o nome do Dataset e uma ligação para editar os dados do Dataset, o nome do Workflow e opção para alterá-lo, informação sobre a última data de acesso ao Workflow e ainda uma ligação para a funcionalidade Guardar Workflow.

Acessibilidade Web

"*Acessibilidade Web significa que pessoas com deficiências podem usar a Web*" [43]. As aplicações Web têm um conjunto de características específicas que aumentam a complexidade da sua utilização por parte de pessoas com algum tipo de *handicap*. É mais complicado para este tipo de pessoas compreender, navegar e interagir com a experiência Web natural para os restantes utilizadores.

Aplicando as boas práticas recomendadas pelo World Wide Web Consortium [44], a aplicação DynamicFlow torna-se não só acessível para pessoas com deficiências, mas também muito mais acessível e utilizável pelo utilizador comum da aplicação.

Se apenas pudéssemos construir o Workflow através da operação de *drag-n-drop* estaríamos a reduzir a acessibilidade do *site* a utilizadores que não tivessem possibilidade de usar um qualquer dispositivo apontador. Para colmatar esta falha, foi

criada a possibilidade de efectuar o clique numa hiperligação ou usar as teclas de atalho disponíveis. A existência de teclas de atalho revela-se também bastante útil, especialmente no acesso ao menu da aplicação. O utilizador pode usar apenas o teclado para efectuar a totalidade das funcionalidades que a aplicação oferece.

4.3 Arquitectura

4.3.1 Diagrama da base de dados da aplicação

Para suportar o armazenamento da informação de Dataset e Workflows pertencentes a cada utilizador, foi necessário elaborar um esquema de base de dados seguindo o modelo relacional, que guardasse estes dados e permitisse um desempenho óptimo na execução das funcionalidades da aplicação.

A Figura 32 esquematiza a estrutura da base de dados, mostrando as diversas tabelas criadas e os tipos de dados de cada campo.

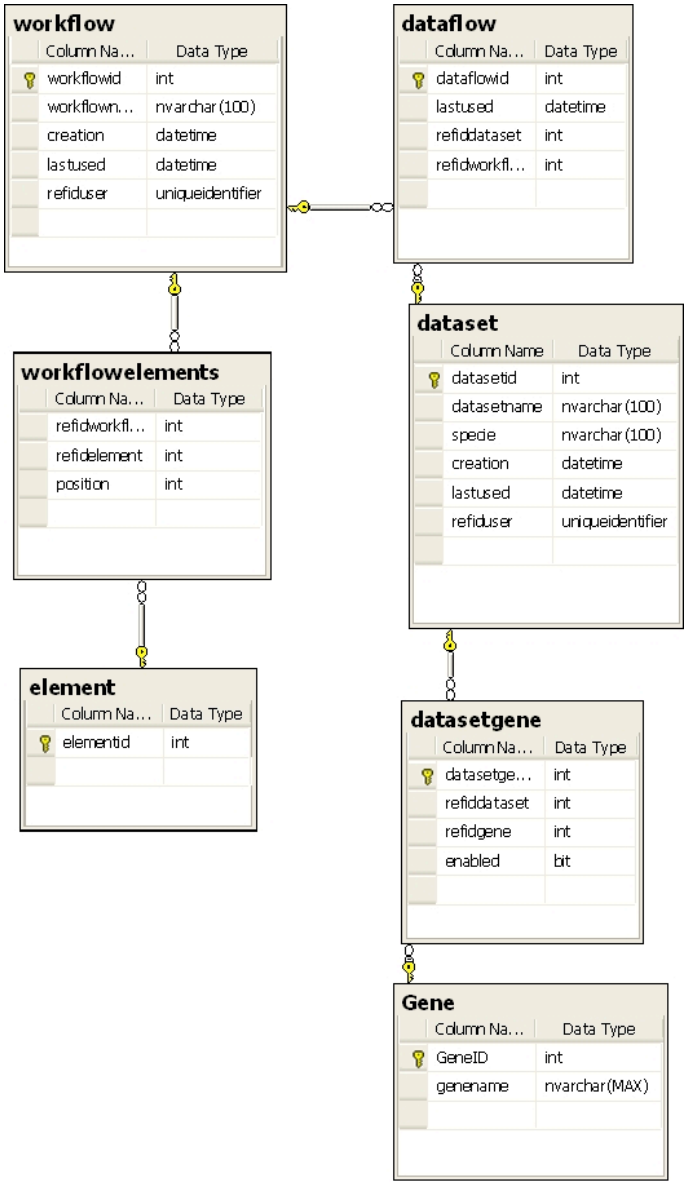


Figura 32 : Esquema da Base de Dados

Como se pode comprovar, o esquema da base de dados é relativamente simples. A Tabela 9 mostra uma pequena descrição dos elementos pertencentes a cada tabela.

Tabela 9 : Descrição das relações da Base de Dados

Tabela	Elemento	Descrição
Workflow	Workflow ID	Identificador único de cada Workflow
	Workflow Name	Nome dado pelo utilizador ao Workflow
	Creation	Data de criação do Workflow
	Last Used	Data da última modificação do Workflow
	Utilizador	Referência para o utilizador a que o Workflow pertence
Dataset	Dataset ID	Identificador único de cada Dataset
	Dataset Name	Nome dado pelo utilizador ao Dataset
	Specie	Espécie a que pertencem os genes do Dataset
	Creation	Data de criação do Dataset
	Last Used	Data da última modificação do Dataset
	Utilizador	Referência para o utilizador a que o Dataset pertence
Dataflow	Dataflow ID	Identificador único de cada conjunto Dataset e Workflow
	Last Used	Data do último acesso ao Dataflow
	Dataset	Referência para o Dataset
	Workflow	Referência para o Workflow
Gene	Gene ID	Identificador único de cada Gene, normalizado para o formato Entrez
	Gene Name	Nome de cada Gene
Dataset Gene	Dataset Gene ID	Identificador único de cada relação entre Dataset e Gene
	Dataset	Referência para o Dataset em que o Gene é utilizado
	Gene	Referência para o Gene
	Enabled	Marcador activo/inactivo sobre o uso do Gene no Dataset
Element	Element ID	Identificador único de cada elemento do Workflow, igual ao XML ID de cada módulo de processamento
Workflow Elements	Workflow	Referência para o Workflow
	Element	Referência para o módulo de processamento
	Position	Posição do módulo de processamento dentro do Workflow

4.3.2 Diagrama de Classes da aplicação

A organização por classes da aplicação encontra-se esquematizada na Figura 33.

De acordo com o modelado, o acesso à base de dados realiza-se numa camada única sintetizada na classe DBAccess. É nesta classe que são implementados todos os métodos de comunicação com as bases de dados. As restantes classes de mapeamento da base de dados – Gene, Workflow, Datas(et) e Element – implementam apenas métodos de ligação às implementações da classe DBAccess, que é instanciada dentro de cada uma das classes. Esta estrutura tenta pôr em prática, da forma mais coerente possível, o modelo proposto na secção 3.3.2. O pré-processamento das páginas não

usa a classe DBAccess directamente, mas sim uma instanciação da mesma presente em cada uma das classes que mapeiam o conteúdo da base de dados.

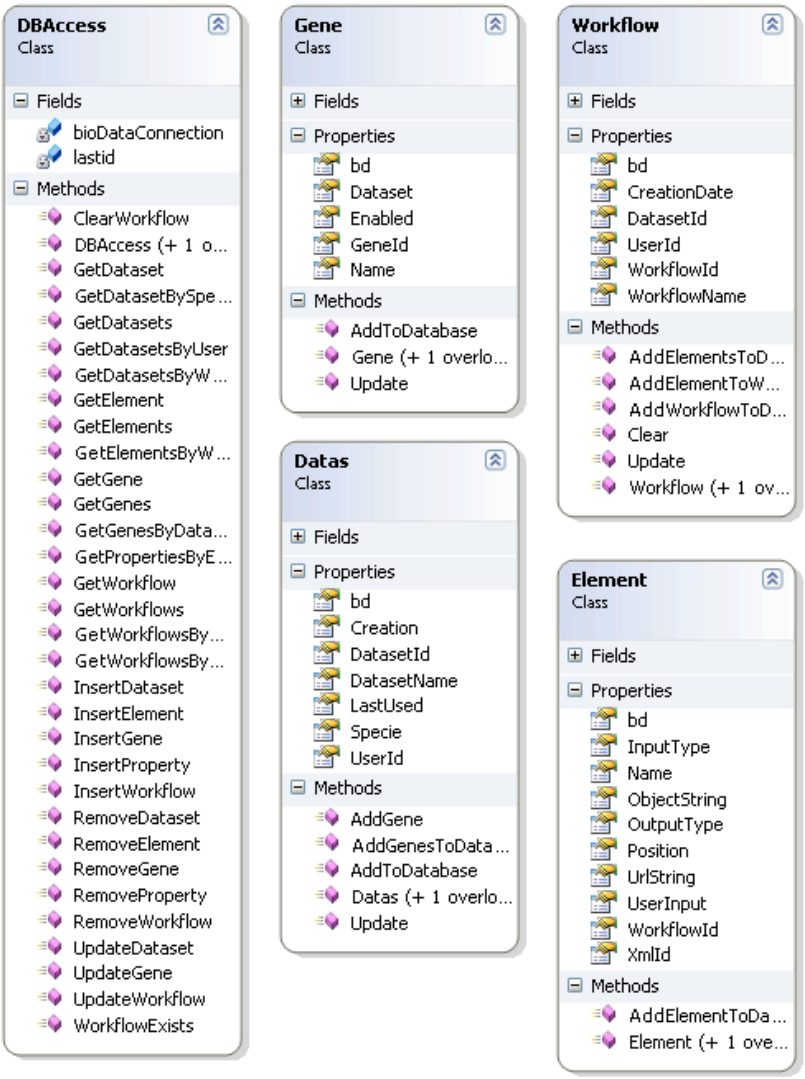


Figura 33 : Diagrama de Classes da aplicação

4.3.3 Fluxo de execução

Devido à arquitectura específica desta aplicação é necessário realçar o fluxo de execução de alguns métodos.

O método mais importante da aplicação é o processamento do Workflow. Este método envolve a comunicação entre o *browser* e diversos módulos de processamento dispersos.

O processamento do Workflow é iniciado após acção do utilizador (Figura 34 – 1). Depois do processamento inicial no *browser*, é estabelecido contacto com o servidor de adaptadores iterativamente (Figura 34 – 2). Este processo é descrito em mais detalhe no capítulo 4.4.4. No final, após obtida a informação do último adaptador, o Javascript despoleta alterações no DOM da página, mudando a interface de utilizador para mostrar os resultados (Figura 34 – 3).

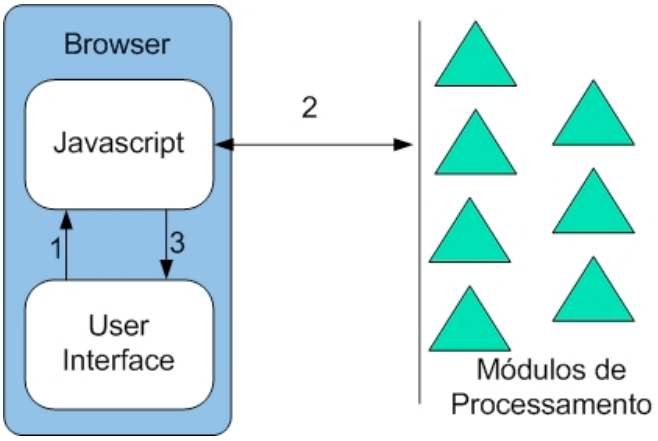


Figura 34 : Fluxo de funcionamento do processamento do Workflow

O outro fluxo de execução típico é seguido pelas restantes funcionalidades da aplicação. As funcionalidades de Gestão de Dataset e de Workflow usam este mecanismo para execução.

O contacto é iniciado pelo utilizador ao realizar uma acção, geralmente chamando uma funcionalidade através do menu da aplicação (Figura 35 – 1). De seguida, é executado um método Javascript que entra em contacto com o Servidor Web da aplicação; esta comunicação pode ser um pedido Http simples ou um pedido de execução de uma função específica (Figura 35 – 2). A aplicação, de acordo com o pedido recebido, devolve os dados pretendidos (Figura 35 – 3). Após processamento pelo Javascript, o resultado da funcionalidade pretendida é mostrado na interface de utilizador (Figura 35 – 4).

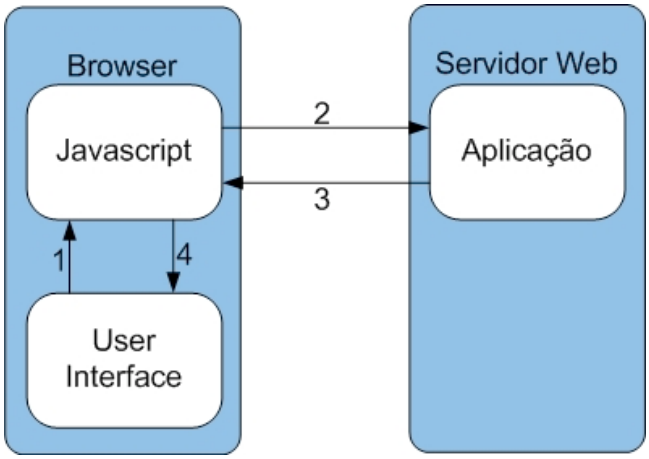


Figura 35 : Fluxo de funcionamento de funcionalidades gerais da aplicação

4.4 Problemas

Como seria de esperar foram encontrados diversos problemas ao colocar em prática toda a arquitectura que tinha sido planeada. Os principais problemas encontrados e as soluções adoptadas bem como as alternativas existentes encontram-se detalhados de seguida nesta secção.

4.4.1 *Same Origin Policy*

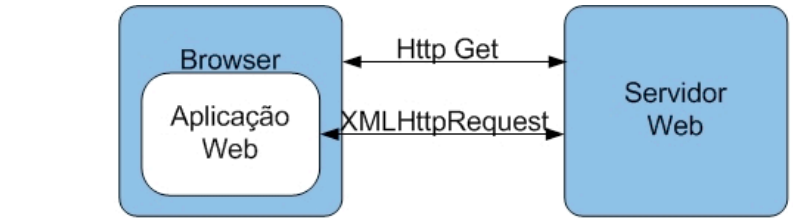
A *Same Origin Policy* diz-nos que apenas os objectos provenientes da mesma origem que a nossa página podem interagir com a nossa aplicação: *"objectos provenientes da mesma origem (isto é, do mesmo host, usando o mesmo protocolo e porto) podem interagir entre si."* [45] (Figura 36 – Modelo 1).

Esta política é usada como método de protecção de ataques a aplicações Web por todos os *browsers* da actualidade.

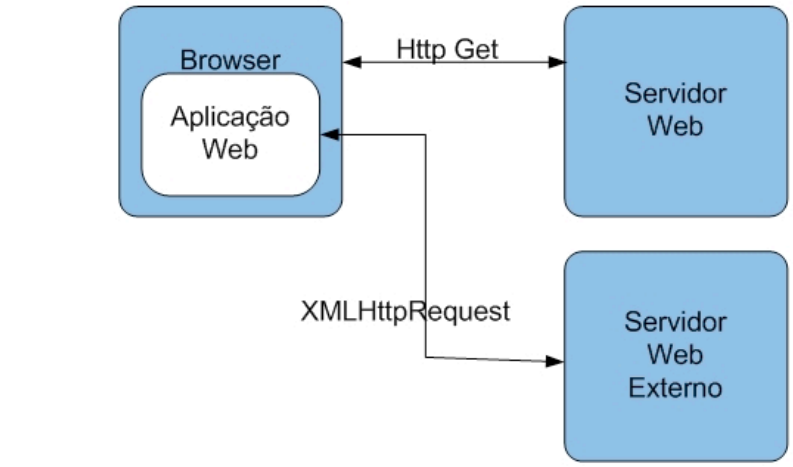
Ao projectar o funcionamento dos diversos adaptadores, partimos do princípio que a interacção apenas se daria entre o *browser* do cliente e o servidor que implementa o adaptador. Esta interacção, sempre que o servidor do adaptador não coincide com o da aplicação, causa uma violação à *Same Origin Policy*, levando a que

não possa ocorrer através dos métodos tradicionais de comunicação entre *browser* e servidor (Figura 36 – Modelo 2).

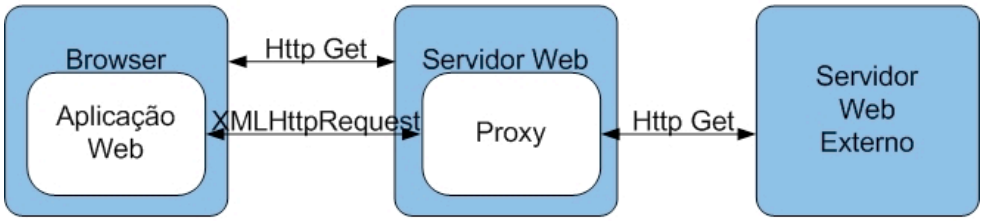
Uma das primeiras soluções testadas consistia em utilizar o nosso servidor da aplicação como *Proxy* do servidor que implementa os módulos de processamento [46] (Figura 36 – Modelo 3).



Modelo 1 - Pedido XMLHttpRequest ao servidor da Aplicação Web (Funcional)



Modelo 2 - Pedido XMLHttpRequest a um servidor externo (Não Permitido)



Modelo 3 - Pedido XMLHttpRequest a um Proxy do servidor externo (Funcional)

Figura 36 : Modelos de funcionamento de pedidos *XMLHttpRequest*

Usar esta solução iria implicar mudanças na arquitectura modelada e causar a inserção de mais um intermediário entre o cliente e o módulo de processamento, diminuindo o desempenho da aplicação e alterando a arquitectura projectada originalmente.

Outra alternativa estudada consistia em usar um pequeno ficheiro com tecnologia Flash para fazer o acesso aos servidores dos adaptadores. Além do já mencionado problema em usar a tecnologia Flash, este método tem também o problema de necessitar de um ficheiro de configuração específico no servidor onde se encontra o módulo de processamento a autorizar explicitamente que aceita os pedidos da nossa aplicação.

A solução usada foi a última estudada: *On Demand Javascript*. Este método implementa uma solução sem intermediários e com processamento no *browser* do utilizador. "*O carregamento inicial da página inclui código Javascript que, além de outras coisas, contém o código necessário para descarregar mais Javascript*" [47]. Explicando, a nossa página pode conter pedaços de código Javascript implementando funções que carregam dinamicamente outros pedaços de código Javascript.

Partindo do princípio que os módulos de processamento devolvem código Javascript, no nosso caso, objectos codificados através de JSON – requisito suportado pelo BioPortal, podemos realizar os pedidos dinamicamente, carregando a resposta JSON dos módulos de processamento no *browser* e acedendo ao objecto que ela contém.

```
addRequestObject : function(reqUrl)
{

    var script = document.createElement("script");
    script.setAttribute("type", "text/javascript");
    script.setAttribute("src", reqUrl);
    document.body.appendChild(script);
},
```

Figura 37 : Extracto de código Javascript para criação de uma tag "script"

O carregamento de código Javascript em *runtime* necessita que seja executado um método de criação dinâmica de etiquetas "*script*". Nas páginas HTML, a *tag* "*script*" - *<script>* - serve para englobar o código Javscript que será carregado na memória do *browser* e estará disponível para execução enquanto a página se encontrar aberta. Esta marca pode englobar pedaços de código ou simplesmente apontar para um ficheiro que contenha o código que será carregado. O endereço do ficheiro que contem o código Javascript que será carregado é indiferente para o

browser, não sendo afectado pela *Same Origin Policy*. Tendo em conta estas características, basta adicionar no processamento de cada elemento do fluxo de informação, uma nova *tag* "*script*" que aponte para o endereço construído a partir da descrição do adaptador, para que o valor de retorno do referido módulo seja carregado dinamicamente em memória. A saída de um determinado módulo de processamento fica, assim, disponível como entrada ao módulo de processamento seguinte.

4.4.2 Definição da norma dos módulos de processamento

Para garantir a independência e a interoperabilidade entre os diversos adaptadores foi necessário definir uma norma que todos respeitassem, que suportasse as diversas definições de configuração de cada módulo e que fosse o mais completa possível.

Para tal, foi escolhido o formato XML para definição do esquema dos diversos adaptadores. Os módulos encontram-se definidos num ficheiro XML que passará depois por uma folha de transformação XSL de forma a obter a lista de módulos a mostrar na área dos módulos de processamento da área de trabalho.

A Tabela 10 mostra os principais elementos presentes na norma XML.

Tabela 10 : Descrição da especificação dos módulos de processamento

Elemento	Descrição
Display Name	Nome do módulo de processamento a mostrar na interface da aplicação.
Description	Pequena descrição do módulo de processamento. O conteúdo deste elemento será mostrado na zona da área de trabalho referente à descrição de cada módulo de processamento.
Input	Descrição do tipo de dados de entrada do módulo de processamento.
Output	Descrição do tipo de dados de saída do módulo de processamento.
UrlString	Endereço a chamar para executar o módulo de processamento.
ObjectString	Descrição do objecto Javascript devolvido pelo módulo de processamento.
UserInput	Opcional. Apenas presente quando o módulo de processamento envolve a inserção de dados pelo utilizador. Um exemplo é um Módulo de filtragem que necessita da inserção do filtro pelo utilizador. No XML, são descritas as características das entradas: uma simples *TextBox* ou o conjunto de valores de uma *Drop Down List*.

Nesta aplicação os diversos módulos de processamento encontram-se agrupados em categorias de forma a simplificar o acesso aos mesmos, ou seja, cada módulo de processamento vai pertencer a uma categoria pré-definida de acordo com a sua área de aplicação na bioinformática.

A Figura 38 mostra um exemplo de um módulo de processamento descrito em XML.

```
<DataType id="1">
  <TypeName>Yeastract</TypeName>
  <Elements>
    <Element xmlid="11">
      <DisplayName>Filter By Transcript Factor</DisplayName>
      <Description>This is a description</Description>
      <Input>
        <Type>Gene</Type>
        <Value>TfGeneId</Value>
      </Input>
      <Output>
        <Type>Gene</Type>
        <Value>TfGeneId</Value>
      </Output>
      <Specie>sce</Specie>
      <BaseUrl></BaseUrl>
      <UrlString>http://bioinformatics.ua.pt/bioportal/Search.aspx?rettype##equal##jsonf
      <ObjectString>obj##id##.note.nameserver_##specie##_Yeastract[##counter##].TfGeneIc
      <XmlString>/note/nameserver_##specie##_Yeastract/TfGeneId</XmlString>
    </Element>
```

Figura 38 : Extracto do ficheiro XML de descrição dos módulos de processamento

Devido às vantagens do XML mencionadas no segundo capítulo desta dissertação, esta opção foi a única considerada para resolver o problema existente na definição da norma que os módulos de processamento teriam de seguir.

4.4.3 Passagem de informação dos módulos de processamento

O requisito de atribuir a carga de processamento do *workflow* ao cliente levanta mais um problema: o armazenamento da informação relativa a cada módulo de processamento.

Como já foi referido, os *wrappers* são modelados através de um ficheiro de configuração escrito em XML. Os dados contidos neste ficheiro são essenciais para o correcto funcionamento do fluxo de informação. No entanto, quando decorre o processamento do Workflow, não temos acesso directo ao ficheiro XML de configuração através do Javascript.

Para conseguir aceder a estes dados seria necessário carregar o ficheiro do servidor, envolvendo mais transferências de dados. Esta solução implicava também um acesso ao ficheiro XML através de Javascript, funcionalidade que não se encontra normalizada, variando de *browser* para *browser*, o que implica a criação de código que suporte os mais variados tipos de implementações de leitura e escrita de ficheiros XML. Deste modo, além de ser mais uma transferência de dados (o que é só por si indesejável) e de termos um ficheiro de configuração bastante grande, temos também a dificuldade de acesso ao ficheiro. Pesando estes pontos negativos, a opção de transferir o ficheiro XML para o cliente foi descartada.

Para ultrapassar os problemas da transferência do ficheiro, optou-se por usar a escalabilidade do formato HTML para armazenar dentro dos seus elementos a informação que necessitamos. As diversas etiquetas HTML podem possuir diversos atributos – pares chave valor; estes atributos são usados pelo *browser* para distinguir elementos, aplicar folhas de estilo ou definir propriedades características de cada *tag*. A vantagem da utilização destes atributos é que o *browser* apenas interpreta aqueles que conhece. Face a esta situação, é possível criar os atributos necessários sem trazer qualquer problema ao *browser* no processamento da página HTML.

```
<li creationid="1"
    style="top: 0px; left: 0px;" xmlid="22"
    urlstring="http://bioinformatics.ua.pt/bioportal/Search.aspx?rettype##equal##json#
    objectstring="obj##id##.note.nameserver_##specie##_Entrez2Go[##counter##].GoId"
    outputtype="Go"
    inputtype="Gene"
    userinput=""
    specie="sce|hsa|dre">Get Go Id by Gene Id
</li>
```

Figura 39 : Extracto do HTML de um módulo de processamento

É possível preencher uma *tag* "*list item*", filha de uma "*unordered list*", com atributos que possibilitam o armazenamento da informação necessária ao processamento do nosso fluxo de informação (Figura 39).

Este método, permite a definição de qualquer número e tipo de atributos – desde que estes não entrem em conflito com os interpretados pelo *browser*. Os atributos podem ser facilmente acedidos ou modificados pelo Javascript, tal como

qualquer outro atributo da norma HTML, o que torna este método a opção ideal para o armazenamento de informação.

4.4.4 Processamento do Workflow do lado do Cliente

Estando ultrapassadas as dificuldades levantadas pela *Same Origin Policy* e pela normalização dos módulos de processamento, o passo seguinte passou por tratar o processamento encadeado destes.

O primeiro requisito a verificar é a consistência do Workflow: é necessário garantir que as entradas de cada adaptador são compatíveis com as saídas do adaptador anterior. Para a prevenção deste erro de consistência, os módulos de processamento que não poderem ser adicionados ao último módulo de processamento do *workflow* são diferenciados visualmente dos restantes e a opção de adição ao Workflow através do clique passa a estar indisponível. Caso o utilizador use o método de *drag-n-drop* para adicionar o adaptador ao *workflow* e este fique inconsistente, a aplicação informa o erro devidamente.

De seguida é necessário carregar em memória toda a informação *wrappers* e a ordem por que serão executados.

Através do método de *On Demand Javascript* descrito anteriormente é feito o primeiro pedido. O pedido é construído de acordo com o especificado no ficheiro XML dos módulos de processamento. Na resposta, acede-se ao objecto devolvido de acordo com a especificação do módulo de processamento contida no XML e é extraída a informação necessária para construir o pedido seguinte. O processo repete-se iterativamente até estar completado o fluxo de informação.

O utilizador é informado do trajecto da informação através da disponibilização dos dados de saída obtidos em cada módulo.

Ao terminar o fluxo de informação – temos a informação dada pelo último elemento do Workflow – o utilizador pode iniciar o processo de análise da informação através dos diversos métodos de visualização disponíveis. Estes métodos são diferenciados de acordo com o tipo de dados a analisar. Apesar da visualização não fazer parte deste trabalho, é importante referir que os diversos métodos de visualização encontram-se disponíveis de acordo com o tipo de dados a visualizar e o

utilizador pode não só analisar os dados de saída finais, mas também os dados de saída de cada módulo de processamento executado no Workflow.

4.5 Sumário

Neste capítulo foram apresentadas as soluções encontradas para satisfazer a arquitectura planeada inicialmente e também alguns problemas encontrados ao longo do desenvolvimento da aplicação.

Grande parte das escolhas feitas para componentes do sistema tiveram em conta as tecnologias usadas nas restantes aplicações em desenvolvimento pelo grupo, isto de forma a melhorar a interligação para partilha de dados entre as aplicações existentes e ainda em desenvolvimento. As tecnologias escolhidas, além de serem as mais amplamente usadas pelas aplicações Web2.0, são também as que fornecem mais garantias de escalabilidade e funcionalidade final da aplicação.

Os problemas encontrados no processo de desenvolvimento foram proveitosos para verificar que existem sempre variadas soluções e que a escolha final nem sempre é fácil e envolve a análise de diversos factores.

A interface da aplicação prototipada, um dos principais objectos de estudo desta dissertação, foi planeada criteriosamente para que, a aproximação de integração de serviços através de fluxos de informação, resultasse e fosse o mais usável possível para o utilizador.

Uma primeira versão experimental da aplicação encontra-se disponível em http://bioinformatics.ua.pt/geDynamicFlow.

5 Conclusões e trabalho futuro

A evolução da Internet levou ao aparecimento de cada vez mais e melhores aplicações Web. O fenómeno da Web2.0 e as potencialidades por ele despoletadas, tornaram possível a existência de aplicações Web tão boas ou melhores que as tradicionais aplicações Desktop. No entanto, a arquitectura destas novas aplicações é bastante mais complexa que as suas predecessoras.

O que se pretende apresentar nesta dissertação é uma proposta de arquitectura (e respectivas escolhas tecnológicas) para integração de serviços que tenha em conta as características da Web2.0 e os requisitos trazidos por uma nova e ágil aproximação com fluxos de informação controlados pelo utilizador.

O desenho da arquitectura e a construção da aplicação levaram a um estudo aprofundado de aplicações Web2.0 existentes e, dentro destas, as que usam uma aproximação com *workflows* semelhante à que se pretendia implementar. Levou também ao estudo das mais variadas tecnologias - HTML, DOM, XML, Javascript, CSS, XSL, Xpath, C#, ASP.NET, *Web Services* – e aplicações / *frameworks* de trabalho – Visual Studio, ASP.NET AJAX, Microsoft SQL Server, Microsoft Windows Server.

Apesar da aplicação criada para testar a arquitectura estar ligada à área da bioinformática, o modelo proposto pretende abranger qualquer área de trabalho e servir de base à construção de novas aplicações e interfaces. Deste modo, a integração em diferentes áreas de trabalho encontra-se facilitada e o processo de desenvolvimento de novas aplicações sobre a arquitectura criada não deve ser moroso.

O futuro da arquitectura está dependente do estudo de áreas onde possa ser inserida e utilizada com sucesso. Áreas como a medicina ou a farmacologia partilham características com a bioinformática, o que possibilita a criação de novas aplicações idênticas ao DynamicFlow.

Dentro do contexto da bioinformática, a aplicação DynamicFlow necessita de algum trabalho até que possa ser considerada como finalizada. É necessário pesquisar e estudar mais fontes de dados e serviços que possam ser usados como adaptadores

na aplicação; isto para que o número de problemas resolvidos pela aplicação cresça (de acordo com o crescimento dos adaptadores disponíveis). É também necessário promover a divulgação da aplicação junto dos utilizadores, para que possam ser conhecidas quais as funcionalidades gostariam de ver implementadas na aplicação e que soluções seriam mais úteis para resolver possíveis problemas. Posteriormente, resta melhorar a interface: não a nível de formas de interacção com o utilizador, mas ao nível de aparência das páginas.

Num quadro mais global, a integração da aplicação DynamicFlow num pacote de aplicações de análise de expressão genética, fomentaria o desenvolvimento de diversas funcionalidades relacionadas com a partilha de informação entre utilizadores e entre aplicações, aproximando mais a aplicação do conceito Web2.0.

6 Referências

1. Bertino, E. and E. Ferrari, *XML and Data Integration.* IEEE Internet Computing, 2001(November-December 2001): p. 2.
2. Dias, G., et al. *Integration of Genetic and Medical Information Through a Web Crawler System*. in *Biological and Medical Data Analysis (ISBMDA' 2005)*. 2005. Aveiro, Portugal.
3. Arrais, J., et al., *GeneBrowser: an approach for integration and functional classification of genomic data.* Journal of Integrative Bioinformatics, 2007.
4. Whalin, D. and M. Gibss, *Professional ASP.NET 2.0 AJAX*. 2007: Wrox. 336.
5. O'Reilly, T. *What Is Web 2.0: Design Patterns and Business Models for the Next Generation of Software*. 2005 [citado em 2008 31-05].
6. ECMA. *Standard ECMA-357*. 2008 [citado em 2008 31-05]; Disponível em: http://www.ecma-international.org/publications/standards/Ecma-357.htm.
7. W3C, W.W.W.C. *W3C Document Object Model*. 2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/DOM.
8. W3C, W.W.W.C. *Extensible Markup Language (XML)*. 2008 22-05-2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/XML.
9. W3C, W.W.W.C. *Extensible Stylesheet Language (XSL)*. 2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/TR/xsl.
10. W3C, W.W.W.C. *XML Path Language (XPath)*. 2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/TR/xpath.
11. JSON. *JSON*. 2008 [citado em 2008 31-05]; Disponível em: http://www.json.org/json-pt.html.
12. Rubio, D. *An Introduction to JSON*. 2007 [citado em 2008 31-05]; Disponível em: http://dev2dev.bea.com/pub/a/2007/02/introduction-json.html.
13. W3C, W.W.W.C. *Web Services @ W3C*. 2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/2002/ws.

14. W3C, W.W.W.C. *SOAP Specifications*. 2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/TR/soap.
15. W3C, W.W.W.C. *Web Service Definition Language (WSDL)*. 2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/TR/wsdl.
16. OASIS. *UDDI Version 3.0.2*. 2008 [citado em 2008 31-05]; Disponível em: http://www.uddi.org/pubs/uddi_v3.htm.
17. ECMA. *Standar ECMA-334*. 2006 [citado em 2008 31-05]; Disponível em: http://www.ecma-international.org/publications/standards/Ecma-334.htm.
18. W3C, W.W.W.C. *Cascading Style Sheets*. 2008 [citado em 2008 31-05]; Disponível em: http://www.w3.org/Style/CSS.
19. PostgreSQL. *PostegreSQL: The world's most advanced open source database*. 2008 [citado em 2008 31-05]; Disponível em: http://www.postgresql.org.
20. Apache, T.A.S.F. *The Apache Softtware Foundation*. 2008 [citado em 2008 31-05]; Disponível em: http://www.apache.org.
21. Sun, S.m. *MySQL:: The world's most popular open source database*. 2008 [citado em 2008 31-05]; Disponível em: http://www.mysql.com.
22. PHP. *PHP: Hypertext Preprocessor*. 2008 [citado em 2008 31-05]; Disponível em: http://www.php.net.
23. Carayatech. *GeoDNS: BIND patch to add geographical filters to views*. 2008 [citado em 2008 31-05]; Disponível em: http://www.caraytech.com/geodns.
24. Apache, T.A.S.F. *Apache Lucene*. 2008 [citado em 2008 31-05]; Disponível em: http://lucene.apache.org/java/docs/index.html.
25. Lighttpd. *lighttpd fly flight*. 2008 [citado em 2008 31-05]; Disponível em: http://www.lighttpd.net.
26. W3C, W.W.W.C. *RSS 2.0 specification*. 2008 [citado em 2008 31-05]; Disponível em: http://validator.w3.org/feed/docs/rss2.html.
27. IETF. *RFC 4287 - The Atom Syndication Format*. 2008 [cited; Disponível em: http://tools.ietf.org/html/rfc4287.
28. Python. *Python Programming Language*. 2008 [citado em 2008 31-05]; Disponível em: http://www.python.org.
29. Psyco. *Psyco*. 2008 16-01-2008 [citado em 2008 31-05]; Disponível em: http://psyco.sourceforge.net.

30. Hollingsworth, D., *The Workflow Reference Model*, W.M. Coalition, Editor. 1995.
31. Gordon, P.M. and Sensen, *Seahawk: moving beyond HTML in Web-based bioinformatics analysis.* BMC Bioinformatics, 2007.
32. Microsoft. *Microsoft Popfly*. 2008 [citado em 2008 31-05]; Disponível em: http://www.popfly.com.
33. Yahoo! *Pipes: Rewire the web*. 2008 [citado em 2008 31-05]; Disponível em: http://pipes.yahoo.com/pipes.
34. Romano, P., D. Marra, and L. Milanesi, *Web services and workflow management for biological resources.* BMC Bioinformatics, 2005.
35. Stein, L.D., *Integrating Biological Databases.* Nature Reviews - Genetics, 2003. **4**(Maio 2003): p. 337-345.
36. Microsoft. *Windows Server 2008*. 2008 [citado em 2008 31-05]; Disponível em: http://www.microsoft.com/windowsserver2008/en/us/default.aspx.
37. Microsoft. *The Official Microsoft ASP.NET Site*. 2008 [citado em 2008 31-05]; Disponível em: http://www.asp.net.
38. ORACLE. *Oracle 11g, Siebel, PeopleSoft | Oracle, Thw World's Largest Enterprise Software Company*. 2008 [citado em 2008 31-05]; Disponível em: http://www.oracle.com/index.html.
39. Microsoft. *Microsoft SQL Server 2005 Home*. 2008 [citado em 2008 31-03]; Disponível em: http://www.microsoft.com/SQL/default.mspx.
40. Microsoft. *.NET Framework Developer Center*. 2008 [citado em 2008 31-05]; Disponível em: http://msdn.microsoft.com/en-us/netframework/default.aspx.
41. Adobe. *Adobe Flash*. 2008 [citado em 2008 31-05]; Disponível em: http://www.adobe.com/products/flashplayer.
42. Microsoft. *Silverlight*. 2008 [citado em 2008 31-05]; Disponível em: http://silverlight.net.
43. W3C, W.W.W.C. *Web Accessibility Initiative (WAI)*. 2006 [citado em 2008 31-05]; Disponível em: http://www.w3.org/WAI.
44. W3C, W.W.W.C. *WAI Guidelines and Techniques*. 2006 [citado em 2008 31-05]; Disponível em: http://www.w3.org/WAI/guid-tech.html.

45. Holz, T., S. Marechal, and F. Raynal, *New Threats and Attacks on the World Wide Web.* IEEE Security & Privacy, 2006: p. 72-75.
46. Yahoo! *Use a Web Proxy for Cross-Domain XMLHttpRequest Calls*. 2008 [citado em 2008 31-05]; Disponível em: http://developer.yahoo.com/javascript/howto-proxy.html.
47. Mahemoff, M., *On-Demand JavaScript*, in *Ajax Design Patterns*. 2006, O'Reilly.